Trim 3D Gaussian Splatting for Accurate Geometry Representation

Lue Fan[1,2*] **Yuxue Yang**[1*] **Minxing Li**[1] **Hongsheng Li**[2,3✉] **Zhaoxiang Zhang**[1✉]

[1]CASIA [2]MMLab, CUHK [3]Shanghai AI Lab

https://trimgs.github.io

Abstract

In this paper, we introduce Trim 3D Gaussian Splatting (TrimGS) to reconstruct accurate 3D geometry from images. Previous arts for geometry reconstruction from 3D Gaussians mainly focus on exploring strong geometry regularization. Instead, from a fresh perspective, we propose to obtain accurate 3D geometry of a scene by Gaussian trimming, which selectively removes the inaccurate geometry while preserving accurate structures. To achieve this, we analyze the contributions of individual 3D Gaussians and propose a contribution-based trimming strategy to remove the redundant or inaccurate Gaussians. Furthermore, our experimental and theoretical analyses reveal that a relatively small Gaussian scale is a non-negligible factor in representing and optimizing the intricate details. Therefore the proposed TrimGS maintains relatively small Gaussian scales. In addition, TrimGS is also compatible with the effective geometry regularization strategies in previous arts. When combined with the original 3DGS and the state-of-the-art 2DGS, TrimGS consistently yields more accurate geometry and higher perceptual quality.

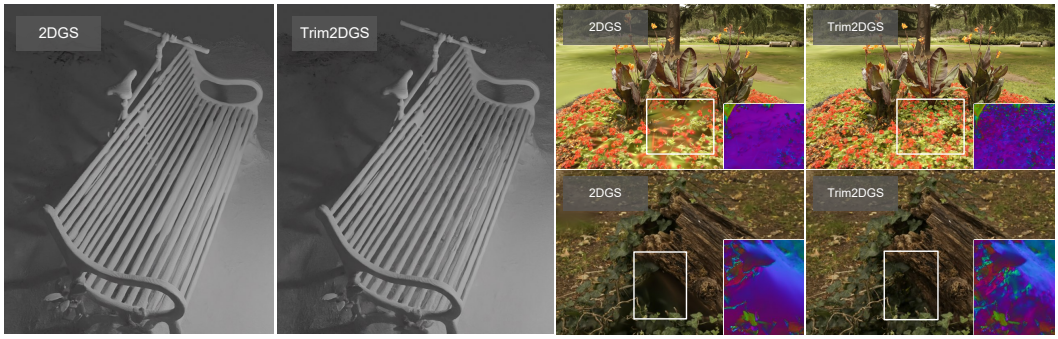


Figure 1: **TrimGS exhibits better geometric details and perceptual quality.** In (a), TrimGS separates the slender crossbars of a bench. (b) showcases that TrimGS excels in rendering intricate details in both color and normals.

1 Introduction

In novel view synthesis, it is a common phenomenon that messy geometry might be hidden behind the high rendering quality, as highlighted by NeRF++ [35]. This issue becomes more prominent with the explicit representation of 3D Gaussian Splatting (3DGS) [16]. To derive accurate explicit geometry from 3D Gaussian representation, recent methods [8, 11, 34, 4, 30] have made attempts to

*Equal contribution.

Preprint. Under review.

reconstruct accurate surfaces from 3D Gaussians. Despite differing methodological focuses, these approaches share the strategy of applying strong geometric regularization to the original 3DGS.

In this paper, we propose to ***trim*** 3D Gaussians, which is a new strategy for accurate geometry reconstruction that complements the previous geometry regularization strategies. Unlike the term "pruning" used in 3DGS, our proposed gradual "trimming" strategy is employed to highlight our method's goal of progressively refining messy Gaussian fields into geometrically accurate forms, akin to lawn trimming. To this end, we first introduce a metric to determine which parts of the Gaussians should be trimmed. In the original 3DGS, opacity serves as this metric, with Gaussians exhibiting very low opacities being removed. However, many Gaussians with low contributions or inaccurate geometric structures often have relatively high opacities, making the default strategy insufficient for maintaining precise geometry. To address this issue, we develop a faithful evaluation metric for measuring Gaussians' contributions, inspired by the weights used in the alpha-blending process. We then design an intuitive yet effective contribution-based trimming procedure and integrate it into the 3DGS training process.

In evaluating the contributions, the scale of Gaussians is a non-negligible factor. Although large Gaussians often have higher contributions to the rendering process, their large sizes limit their ability to represent intricate geometry. They tend to produce blurred patterns in high-frequency regions, significantly diminishing perceptual quality. Beyond their limited capacity, our experimental and theoretical analyses, detailed in § 3 and § A.1, reveal that the optimization of large Gaussians is hampered by noisy gradients. To address these drawbacks, we propose modifying the densification strategy in 3DGS to control and maintain relatively small Gaussian scales.

Although we follow a new perspective, the proposed TrimGS remains compatible with existing geometry regularization strategies. Therefore, we also incorporate a normal consistency regularization to ensure better and more stable geometry optimization. Furthermore, the compatibility makes TrimGS seamlessly integrated with the emerging 2DGS.

Our contributions are summarized as follows.

- We propose TrimGS to learn 3D Gaussians with accurate geometry, utilizing a contribution-based Gaussian trimming strategy. It offers a fresh perspective on geometry reconstruction from 3D Gaussians, complementing conventional geometry regularization techniques.
- Through experimental and theoretical analysis, we highlight Gaussian scale is a non-negligible factor in trimming and optimization. Thus, we advocate for maintaining relatively small Gaussian scales, with the hope that our analysis provides insights to the community.
- The proposed TrimGS is a general technique and compatible with the original 3DGS and the recent 2DGS methods. Based on them, TrimGS consistently produces more accurate geometry and higher perceptual quality.

2 Related work

2.1 Novel View Synthesis

Methods for Novel View Synthesis (NVS) can be partitioned into implicit methods and explicit methods based on the used representation. NeRF [20] was the first to employ implicit radiance fields for representing 3D scenes for NVS. This pioneering work uses a multi-layer perceptron (MLP) to encode geometric and photometric information at every spatial position. The directional encoding enables view-dependent appearances. Volume rendering techniques [14, 5] are leveraged to render the implicit neural field into images. The most significant issue for the original NeRF is the low training and rendering efficiency. Thus a line of methods aims for improving its efficiency. Grid-based feature representation is proven effective in accelerating convergence. Representative methods [22, 17, 7, 26] propose to optimize the grid-based feature instead of neural parameters in MLP. For the rendering efficiency, baking strategy [24, 9, 32, 25] and sparse hash-based grid [22] are leveraged for performance boosting. Another line of works [1, 2, 10] focuses on improving the rendering quality of NeRF, especially addressing the anti-aliasing issue. On the other hand, explicit radiance fields, such as Plenoxels [7], and more recently, 3D Gaussian Splatting [16], offer alternative approaches and are getting popular. The 3DGS employs 3D Gaussians to depict complex scenes, achieving impressive rendering quality and efficiency. However, the reconstructed geometry of the

original 3DGS is quite messy. In the following paragraph, we briefly introduce the recent advances in accurate geometry extraction from 3DGS.

2.2 Surface Reconstruction with Gaussians

The explicit representation of 3D Gaussians makes extracting accurate geometry an intriguing topic. Almost all previous methods [8, 28, 11, 3, 33], focus on applying geometric regularization, which is proven effective. Notably, SuGaR [8] leverages signed distance function and density to supervise Gaussian distribution, forcing them to align with object surfaces. Then it extracts meshes with Poisson surface reconstruction [15]. Finally, it proposes a novel binding strategy to bind 3D Gaussians with coarse meshes and conduct further refinement. SuGaR enables flexible editing of 3D Gaussians by editing the coupled meshes. Recently, NeuSG [3] and GSDF [33] combine 3D Gaussians with implicit surfaces. They optimize 3D Gaussian fields together with signed distance functions to obtain high-quality surfaces. GS2Mesh [30] utilizes stereo depth estimation to enhance the depth quality for fusion. The most recent 2DGS [11] introduces 2D disk representation to replace the 3D representation, yielding more smooth object surfaces. Meanwhile, in addition to geometry regularization, 2DGS employs Truncated Signed Distance Function (TSDF) fusion and Marching Cubes [19] for mesh extraction, which shows impressive robustness to floaters and noisy depth. A similar idea of combining Gaussians and surfels is concurrently proposed by GaussianSurfels [4]. GOF [34], a concurrent work of 2DGS, mainly focuses on unbounded scenes. It leverages ray-tracing-based volume rendering of 3D Gaussians, enabling directly extracting geometry 3D Gaussians utilizing levelset, without resorting to Poisson reconstruction or TSDF fusion as in SuGaR and 2DGS. Although these methods produce smooth surfaces with strong geometric regularization, they face challenges in capturing detailed geometry and textures. In this paper, in addition to solely focusing on regularization, our method learns geometrically accurate 3D Gaussian by trimming strategy and enhancing the detailed quality by scale control.

2.3 Pruning Strategies for Gaussians

Pruning is an important technique in 3DGS. Recently, some methods prune Gaussians during training to control the total number of Gaussian primitives, whether incorporating additional modules [18, 13] or designing a certain criterion [6, 23]. [18] develops a learnable mask strategy that eliminates Gaussian based on its volume and transparency while [13] applies a Virtual Camera View Pruning method to mark and eliminate outliers. [23] calculates a redundancy score by an intersection test according to which Gaussians are filtered and [6] designs a significant score that measures the contribution of a Gaussian to the rendered image and prunes those with low scores.

While these methods all focus on the pruning strategy, they mainly treat it as an approach to the reduction of memory cost [23, 18, 6] instead of more accurate geometry. Therefore, they tend to keep larger Gaussians while pruning the smaller ones, which reduces the capacity to represent geometry details.

3 Preliminaries

In this section, we present the preliminaries of 3D Gaussian Splatting (3DGS) and our analysis of its properties.

3.1 3D Gaussian Splatting

3DGS creates a set of 3D Gaussians to explicitly represent the target 3D scene. Each Gaussian is defined by a position μ, covariance matrix Σ, opacity σ, and SH coefficients that determine its view-dependent color. To render an image, 3DGS utilizes an alpha-blending process to accumulate the color of each pixel. Let p denote the target pixel, its color I_p is given by

$$I_p = \sum_{i \in \Lambda} \mathbf{c}_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{1}$$

where Λ is the set of Gaussians that affects p, with their depth in ascending order. $\mathbf{c}_i$ denotes the color of the i-th Gaussian generated by SH coefficients, depending on the observing point. The blending

weight α_i is acquired by querying from the corresponding Gaussian distribution

$$\alpha_i = \sigma_i \cdot \exp\left(-\frac{1}{2}(\boldsymbol{p}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\boldsymbol{p}-\boldsymbol{\mu}_i)\right), \tag{2}$$

with $\boldsymbol{p}$ denoting the coordinate of pixel image space and σ_i being the opacity parameters of the i-th Gaussian. It is noticeable that the normalization coefficient of Gaussian distribution is ignored in Eq. (2), with the standard version given as follows

$$G(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{3/2}(\det\boldsymbol{\Sigma})^{1/2}}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right). \tag{3}$$

Thus, the Gaussian distribution adopted in 3DGS implementation is actually an ***unnormalized*** formulation. This property is considered in the following gradient analysis and the contribution metric design in § 4.1.

3.2 Gradient Analysis

As we mentioned in § 1, large Gaussians usually have more noisy gradients. Here we conduct a simple analysis. During training, we accumulate the gradient of the Gaussian positions for hundreds of iterations. Due to the ignored normalization factor in Eq. (2), we manually normalize the gradient by the determinant of the covariance Σ for fair gradient comparison between Gaussians of different sizes. The relationship between Gaussian sizes and gradients is shown in Table 1. As can be seen, large Gaussians have much smaller normalized gradients.

Table 1: **The relationship between Gaussian size and gradients.** The size of a 3D Gaussian is measured by the determinant of its covariance matrix Σ.

Size	$[10^{-6}, 10^{-5})$	$[10^{-5}, 10^{-4})$	$[10^{-4}, 10^{-3})$	$[10^{-3}, \infty)$
Normalized gradient norm	9.85	2.81	0.60	0.08

We attribute this to the conflict of pixel-wise gradient. For each Gaussian, its gradient is computed by summing the gradients of pixel-wise photometric loss. Inconsistent gradients of these pixels result in a relatively small summation. Large Gaussians, which cover a wide range of pixels, are more likely to be trapped in this dilemma, especially when the target 3D scene is complex and rich in detail. A theoretical demonstration of this phenomenon is given in Appendix A.1.

4 Method

Our methodology has three components. The major one is contribution-based trimming presented in § 4.1. We further introduce a scale control strategy for better contribution evaluation and rendering details in § 4.2. Finally, we propose a normal regularization to enhance the geometry in § 4.3.

4.1 Gaussian Trimming

The most important factor in Gaussian Trimming is to properly evaluate the contribution of each individual Gaussian. The original 3DGS directly adopts Gaussian opacity as the contribution and removes the low-opacity ones. However, as discussed in § 1, such an opacity-based strategy might mistakenly removes important Gaussians but preserves high-opacity floaters. Inspired by the alpha-blending process, we propose a more faithful metric to evaluate the Gaussian contribution.

Single-view Contribution. We first consider the contribution of a Gaussian to a single view. In the alpha blending formulated by Eq. (1), the blending weight $\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j)$ can be used to indicate the Gaussian's contribution to a pixel. In this way, the overall contribution $\mathbf{C}_k$ of a Gaussian to the k-th rendered image can be the sum of alpha-blending weights of all related pixels, formulated as

$$\mathbf{C}_k = \sum_{p\in\mathbb{P}_k} \alpha_{i(p)} \prod_{j=1}^{i(p)-1}(1-\alpha_j). \tag{4}$$

In Eq. (4), $\mathbb{P}_k$ stands for the 2D projected region of the Gaussian in k-th view. Here we use the notation $i(p)$ instead of i used in Eq. (1) because the Gaussian has different depth orders in different pixel rays.

Eq. (4) is the theoretically accurate evaluation of Gaussian contribution. However, according to our analysis in § 3, the Gaussian distribution adopted in 3DGS is not normalized. Directly applying Eq. (4) outputs quite significant contributions for large Gaussians and very low contributions for small Gaussians. Since large Gaussians have limited capacities to represent the details and are hard to optimize as demonstrated in § 3.2, we further manually normalize Eq. (4) into

$$\mathbf{C}_k = \frac{1}{|\mathbb{P}_k|} \sum_{p \in \mathbb{P}_k} \left(\alpha_{i(p)}\right)^{\gamma} \left(\prod_{j=1}^{i(p)-1} (1-\alpha_j) \right)^{(1-\gamma)}. \tag{5}$$

Eq. (5) has two differences from Eq. (4): (1) The contribution is normalized by the number of pixels in the projected area; (2) We introduce a hyper-parameter γ to control the contribution components. When γ becomes larger, Eq. (5) has less weight for the transmittance $\prod_{j=1}^{i(p)-1}(1-\alpha_j)$ and degrades to the default opacity-based pruning in 3DGS.

Another reason for introducing γ is to control a contribution bias. Intuitively, points deviating from the surface usually lead to inaccurate geometry. Among these inaccurate Gaussians, those near outside surfaces usually have large transmittance while inside ones usually have smaller transmittance. Therefore, γ controls the bias toward the Gaussians deviating to inner sides or outer sides.

Multi-view Contribution. Then we consider the overall contribution of a Gaussian to all views. A Gaussian usually has large contributions only in a limited number of views. Thus, we use the average contribution value of a small number of high-contribution views as the overall contribution, formally denoted as

$$\mathbf{C} = \frac{1}{|\mathbb{V}|} \sum_{k \in \mathbb{V}} \mathbf{C}_k. \tag{6}$$

$\mathbb{V}$ is the view set where the Gaussian has top contributions and typically we choose top-5 views. In practice, the contribution evaluation method can be efficiently implemented with some simple modifications in the CUDA kernel.

Trimming. During training, we perform the proposed trimming at pre-defined intervals of iterations, where the interval is typically 1,000 iterations. At each time, we traverse all views in the training set to evaluate the multi-view contribution of each Gaussian. A certain percentage of Gaussians with the lowest contributions are removed.

4.2 Scale-driven Densification

According to our analysis in § 3.2, large Gaussian is a sub-optimal representation for geometry details and high-frequency regions. To maintain small Gaussian sizes, if the maximum scale of a Gaussian is larger than a scene-dependent scale threshold τ_s, we split it into K smaller Gaussians and shrink the scale by a predefined factor, similar to the splitting implementation in the original 3DGS.

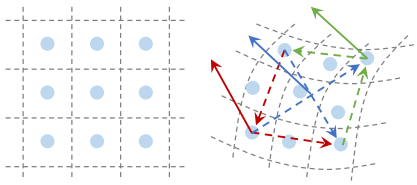
This strategy essentially abides by gradient-driven densification in the original 3DGS. If a Gaussian gradually grows to a large Gaussian, it is likely to receive significant **accumulated** gradients for scales. Thus, splitting those large Gaussians has a similar effect to splitting the Gaussians with large gradients while being more straightforward and intuitive.

4.3 Normal Regularization

In addition to Gaussian trimming, TrimGS can also be seamlessly combined with geometric regularization in previous methods. We therefore further propose a normal regularization loss for better geometry learning. The basic idea is forcing the consistency between the normals of Gaussians and the normals derived from rendered depth maps. The normal of a Gaussian is defined as the orientation of its shortest axis. The challenge here is that both Gaussian normals and depth-derived normals are quite noisy at the beginning, leading to noisy supervision. Thus we propose a robust normal calculation method from rendered depth maps.

Considering a local $k \times k$ window in a rendered depth map, there are k^2 depth points $p \in \mathbb{R}^3$, which can be calculated from the depth value and the camera pose. Any pair of the k^2 depth points represents a 3D vector in the tangent plane of the surface. Thus a normal can be derived from any two pairs of depth points by cross-product. We formally define the normal of the local window as

$$\mathbf{n} = \frac{1}{|\mathbb{T}|} \sum_{\mathbf{t}_i, \mathbf{t}_j \in \mathbb{T}} \mathbf{t}_i \times \mathbf{t}_j, \tag{7}$$



(a) Rendered depth map (b) 3D depth points

Figure 2: **Illustration of robust normal calculation from rendered depth map.**

where $\mathbb{T}$ is the set of all tangent vectors defined by pairs of depth points in the local window. In practice, we do not traverse all the tangent vectors in $\mathbb{T}$ but sample some of them as Figure 2 (b) shows. This normal map derived from a depth map is denoted as $\mathbf{N}_D$.

On the other hand, we render Gaussian normals into another normal map $\mathbf{N}_G$. L1 loss between $\mathbf{N}_D$ and $\mathbf{N}_G$ is adopted to ensure the normal consistency. In this way, the overall loss function is

$$\mathcal{L} = \alpha_1 \mathcal{L}_c + \alpha_2 |\mathbf{N}_D - \mathbf{N}_G|_1, \tag{8}$$

where the $\mathcal{L}_c$ is the default photometric loss in 3DGS. For a fair comparison, when combined with 2DGS, it utilizes the depth distortion loss and the default normal regularization in 2DGS.

5 Experiments

5.1 Setup

Model variants. We apply TrimGS to both original 3DGS [16] and emerging 2DGS [11]. For clarity, we name 3DGS-based version ***Trim3DGS*** and 2DGS-based version ***Trim2DGS***. For Trim2DGS, we also leverage the depth distortion loss [1, 27] in 2DGS.

Implementation details. We implement TrimGS based on the framework of 3DGS [16] and extend the CUDA kernel to calculate the contribution of each Gaussian for trimming as discussed in § 4.1. To be compatible with the original 3DGS and 2DGS, we perform a fast 7K-iteration optimization based on their pretrained models. We conduct trimming every 1,000 iterations and we remove 10% Gaussian with the lowest contribution each time. Other training hyperparameters are the same with 3DGS and 2DGS. For mesh extraction, we utilize the truncated signed distance fusion (TSDF) to fuse rendering median depth maps following 2DGS. All experiments are conducted on NVIDIA 3090 GPUs.

Table 2: **Quantitative comparison of reconstructed meshes on the DTU Dataset**.

		24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean ↓
implicit	NeRF [21]	1.90	1.60	1.85	0.58	2.28	1.27	1.47	1.67	2.05	1.07	0.88	2.53	1.06	1.15	0.96	1.49
	VolSDF [31]	1.14	1.26	0.81	0.49	1.25	0.70	0.72	1.29	1.18	0.70	0.66	1.08	0.42	0.61	0.55	0.86
	NeuS [29]	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
explicit	3DGS [16]	2.14	1.53	2.08	1.68	3.49	2.21	1.43	2.07	2.22	1.75	1.79	2.55	1.53	1.52	1.50	1.96
	SuGaR [8]	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
	GaussianSurfels [4]	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
	2DGS [11]	0.48	0.91	0.39	0.39	1.01	0.83	0.81	1.36	1.27	0.76	0.70	1.40	0.40	0.76	0.52	0.80
	GOF [34]	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
	Trim3DGS (ours)	0.52	0.84	0.58	0.41	1.07	1.02	0.82	1.26	1.48	0.75	0.82	1.23	0.50	0.61	0.52	0.83
	Trim2DGS (ours)	0.48	0.82	0.44	0.45	0.95	0.75	0.74	1.18	1.13	0.72	0.70	0.99	0.42	0.62	0.50	0.72

5.2 Mesh Evaluation

We first follow the standard evaluation protocol in previous methods [11] to evaluate the quality of reconstructed meshes on the DTU dataset [12], as shown in Table 2. Based on the original 3DGS, the proposed Trim3DGS achieves significant performance improvement by reducing Chamfer Distance error by 1.13. The proposed Trim2DGS also achieves further performance improvement based on the state-of-the-art 2DGS. Figure 3 shows the qualitative comparison on DTU and MipNeRF-360 [1] datasets.

Figure 3: **Qualitative comparison of meshes on the DTU and MipNeRF360 dataset.** Note SuGaR [8] uses a different mesh extraction strategy from others, so it includes some background. 3DGS and SuGaR have ***messy geometry*** and 2DGS exhibits slight ***over-smoothness***.

5.3 Point Evaluation

The TSDF-based mesh extraction adopted in this paper and previous arts can yield smooth meshes. However, the mesh quality is not the golden standard to evaluate the geometric quality of 3D Gaussian fields for the following two reasons: (1) The median depth map rendering (Appendix B.1) and TSDF generation are quite robust to floaters and noisy Gaussians around the surfaces; (2) Due to the limitation of quantization precision in TSDF, geometry details cannot be represented by explicit meshes.

To more properly evaluate the geometric quality of 3D Gaussian fields, we further propose to directly evaluate the quality of raw point clouds (i.e., Gaussian centers) by Chamfer Distance. We adopt a simple downsampling strategy to avoid unfairness caused by different local densities of different Gaussian fields. The details are presented in the Appendix B.2. The results in Table 3 show that the proposed method also achieves consistent improvement on the raw point evaluation.

Table 3: **Quantitative comparison of point clouds (i.e., Gaussians' centers) on the DTU dataset**.

	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean ↓
3DGS [16]	1.26	1.35	1.72	1.35	1.48	1.80	1.54	3.49	1.85	1.42	1.69	1.54	1.29	1.41	1.35	1.64
2DGS [11]	0.86	1.05	0.95	0.8	1.01	1.34	1.13	3.10	1.58	1.08	1.10	1.28	0.81	0.89	0.75	1.18
Trim3DGS	0.97	0.94	0.97	0.73	1.21	1.54	1.27	3.46	1.72	1.30	1.14	1.40	1.10	1.14	0.97	1.32
Trim2DGS	0.76	0.78	0.72	0.68	1.08	1.03	0.90	3.18	1.37	1.10	0.90	1.08	0.60	0.80	0.71	1.05

5.4 Rendering Evaluation

As we discussed in § 3 and § A.1, large Gaussians have limited capacity to represent intricate geometry details and high-frequency regions such as *leaves* and *lawn*, which greatly weakens the perceptual quality. In this sub-section, we demonstrate that our scale control strategy in § 4.2 results in better perceptual rendering quality, as shown in Table 4. We emphasize the improvement on the LPIPS metric, which focuses more on the human-vision-sensitive high-frequency and sharp regions as proven in LPIPS [36]. In contrast, PSNR is slightly biased towards blurred results due to its least square formulation.

Our visualization results in Figure 4 also verify this point, where our results show much better rendering quality in high-frequency regions. Since the very tiny geometry details cannot be represented by meshes due to the resolution limitation of TSDF, here we use the normal map to showcase the geometry details. As can be seen, the previous 2DGS tends to ***over-smooth*** these regions in both RGB images and normals.

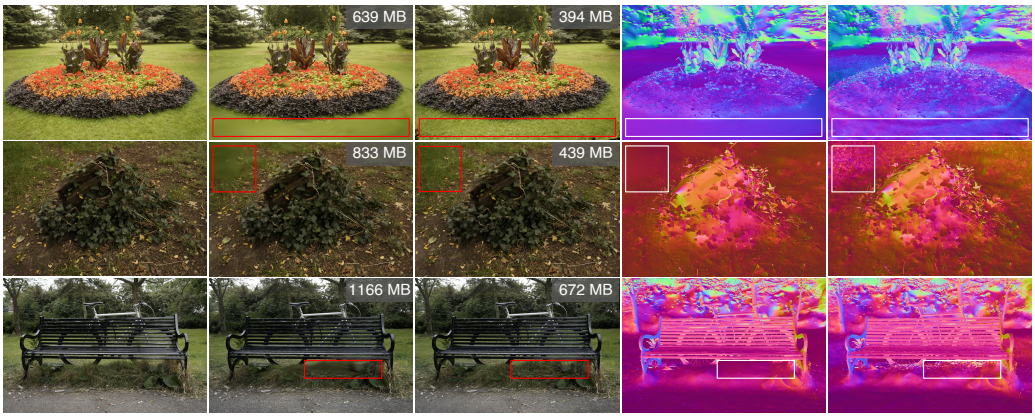


Figure 4: **Comparison of rendering quality (test-set view) between 2DGS and Trim2DGS in MipNeRF360 dataset.** Our Trim2DGS enhances perceptual quality in high-frequency regions, mitigating the over-smoothness in 2DGS. Notably, Trim2DGS substantially reduces the storage consumption, credited to our proposed contribution-based trimming technique.

Table 4: **Quantitative comparison of rendering quality for MipNeRF360 dataset.** Trim3DGS and Trim2DGS result in better perceptual quality, particularly in terms of the LPIPS metric. †: mean performance of outdoor / indoor scenes. *: our re-implementation.

	bicycle	flowers	garden	stump	treehill	out. mean†	room	counter	kitchen	bonsai	in. mean†
PSNR ↑											
3DGS*	25.18	21.33	27.39	26.53	22.48	24.58	31.40	28.96	31.22	32.16	30.93
SuGaR	23.12	19.25	25.43	24.69	21.33	22.76	30.12	27.57	29.48	30.59	29.44
2DGS*	24.75	20.97	26.58	26.18	22.31	24.16	30.56	28.01	30.13	31.21	29.98
Trim2DGS	24.95	20.80	26.53	26.28	22.01	24.11	30.29	27.91	30.03	31.12	29.84
Trim3DGS	25.16	21.29	27.28	26.59	22.36	24.54	31.05	28.89	31.07	32.03	30.76
SSIM ↑											
3DGS*	0.763	0.599	0.865	0.767	0.630	0.725	0.919	0.907	0.923	0.941	0.923
SuGaR	0.639	0.486	0.776	0.686	0.566	0.631	0.910	0.892	0.908	0.932	0.911
2DGS*	0.739	0.569	0.843	0.757	0.621	0.706	0.905	0.890	0.914	0.929	0.910
Trim2DGS	0.755	0.580	0.849	0.764	0.622	0.714	0.910	0.896	0.919	0.934	0.915
Trim3DGS	0.767	0.602	0.864	0.770	0.630	0.727	0.917	0.907	0.927	0.942	0.923
LPIPS ↓											
3DGS*	0.212	0.341	0.108	0.220	0.330	0.242	0.220	0.201	0.128	0.205	0.189
SuGaR	0.344	0.416	0.220	0.335	0.429	0.349	0.245	0.232	0.164	0.221	0.216
2DGS*	0.255	0.378	0.138	0.256	0.367	0.279	0.244	0.232	0.147	0.228	0.213
Trim2DGS	0.216	0.342	0.118	0.229	0.322	0.246	0.219	0.208	0.133	0.212	0.193
Trim3DGS	0.202	0.334	0.111	0.215	0.323	0.237	0.218	0.196	0.127	0.201	0.186

We further emphasize that TrimGS does not improve the quality of high-frequency regions by simply splitting more Gaussians, which is shown by the marked storage consumption in Figure 4. This is credited to the proposed contribution-based trimming technique, which accurately identifies and removes the redundant Gaussians.

5.5 Ablation Study

Effectiveness of major components. We first evaluate the effectiveness of major components in TrimGS, including contribution-based trimming, scale control, and normal regularization. Table 5 shows the performance roadmap from the original 3DGS to our full model, leading to the following three findings. (1) The contribution-based trimming effectively boosts the geometry quality of original 3D Gaussians, especially on point-based evaluation (Table 5). (2) Scale control leads to slight improvements. This is because the overall geometric metrics are not sensitive to detail changes and are dominated by smooth surfaces of large areas. (3) Normal regularization leads to significant improvement for the mesh-based metric but has no help for the point-based metric. This is because

(a) flowers (b) bicycle

Figure 5: **Visualization of Gaussian centers between 2DGS and Trim2DGS from *flowers, bicycle* scenes in MipNeRF360 dataset.** Trim2DGS exhibits a more uniform Gaussians while significantly reducing storage consumption. A better video illustration can be found in our project page.

normal regularization forces the surfaces to be smoother, therefore having a preference of the mesh quality.

Table 5: **Roadmap from 3DGS to Trim3DGS on the DTU Dataset**. All experiments are fine-tuned for 7K iterations for fair comparison.

		24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean CD ↓
mesh	3DGS finetune 7K	1.03	0.94	1.50	0.88	2.29	1.97	1.22	1.51	1.63	1.16	1.32	1.59	1.22	0.98	0.93	1.35
	+ Trimming	1.12	0.92	1.40	0.58	2.21	1.82	1.12	1.43	1.61	1.06	1.19	1.54	0.70	0.83	0.84	1.22
	+ Scale control	1.15	0.93	1.33	0.56	2.16	1.76	1.12	1.42	1.59	1.01	1.08	1.44	0.69	0.83	0.79	1.19
	+ Normal Reg.	0.52	0.84	0.58	0.41	1.07	1.02	0.82	1.26	1.48	0.75	0.82	1.23	0.50	0.61	0.52	0.83
point	3DGS finetune 7K	1.22	1.32	1.70	1.34	1.46	1.78	1.52	3.45	1.82	1.39	1.68	1.50	1.30	1.39	1.35	1.61
	+ Trimming	1.02	0.95	1.29	0.72	1.24	1.46	1.23	3.21	1.69	1.18	1.26	1.21	0.99	1.02	0.95	1.29
	+ Scale control	1.04	0.93	1.10	0.74	1.24	1.46	1.21	3.21	1.68	1.16	1.16	1.21	1.00	1.02	0.96	1.27
	+ Normal Reg.	0.97	0.94	0.97	0.73	1.21	1.54	1.27	3.46	1.72	1.3	1.14	1.40	1.10	1.14	0.97	1.32

Contribution designs. Here we adopt some potential alternatives to our default design. (1) The default opacity-based contribution in the original 3DGS. (2) The significance score in LightGaussian [6], based on projection area and Gaussian volume. (3) Unnormalized contribution in Eq. (4). Table 6 shows that our design outperforms these alternatives, especially on the point-based metric.

Table 7 shows the effectiveness of different γ in Eq. (5). The results demonstrate smaller γ has better performance. It reveals that the transmittance term with $(1-\gamma)$ as exponential is more important than the Gaussian opacity with γ as exponential. It again verifies that the proposed contribution is more reasonable than the default opacity-based contribution.

Table 6: **Comparisons with other strategies for contribution evaluation on the DTU dataset.**

		24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean CD ↓
mesh	Trim3DGS	0.52	0.84	0.58	0.41	1.07	1.02	0.82	1.26	1.48	0.75	0.82	1.23	0.50	0.61	0.52	0.83
	Opacity-based	0.75	0.92	0.70	0.68	1.71	1.15	0.99	1.49	1.59	1.01	0.86	1.49	1.50	0.66	0.56	1.07
	LightGaussian	0.51	0.86	0.63	0.53	1.16	1.03	0.84	1.25	1.50	0.75	0.83	1.39	0.71	0.63	0.53	0.88
	Unnormalized	0.50	0.85	0.66	0.42	1.05	1.02	0.81	1.24	1.49	0.73	0.81	1.28	0.54	0.62	0.54	0.84
point	Trim3DGS	0.97	0.94	0.97	0.73	1.21	1.54	1.27	3.46	1.72	1.3	1.14	1.40	1.10	1.14	0.97	1.32
	Opacity-based	1.36	1.44	1.52	1.32	2.26	2.26	1.91	4.84	2.16	2.01	1.58	1.70	2.72	1.49	1.42	2.00
	LightGaussian	1.28	1.41	1.47	1.58	1.93	2.18	1.71	3.85	1.95	1.70	1.62	2.01	2.06	1.61	1.42	1.85
	Unnormalized	1.05	1.04	1.10	0.86	1.39	1.69	1.38	3.57	1.76	1.40	1.31	1.52	1.36	1.24	1.09	1.45

Table 7: **Performance of Trim3DGS on the DTU Dataset with different γ as discussed in Eq. (5).**

		24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean CD ↓
mesh	$\gamma = 0.25$	0.51	0.81	0.61	0.44	1.12	1.01	0.81	1.27	1.47	0.74	0.81	1.18	0.48	0.60	0.51	0.82
	$\gamma = 0.50$	0.52	0.84	0.58	0.41	1.07	1.02	0.82	1.26	1.48	0.75	0.82	1.23	0.50	0.61	0.52	0.83
	$\gamma = 0.75$	0.53	0.85	0.65	0.46	1.11	1.01	0.84	1.25	1.49	0.74	0.82	1.26	0.67	0.64	0.54	0.86
point	$\gamma = 0.25$	0.90	0.92	1.03	0.75	1.17	1.54	1.21	3.40	1.67	1.27	1.44	1.40	1.06	1.13	0.95	1.30
	$\gamma = 0.50$	0.97	0.94	0.97	0.73	1.21	1.54	1.27	3.46	1.72	1.30	1.14	1.40	1.10	1.14	0.97	1.32
	$\gamma = 0.75$	1.10	1.11	1.15	0.80	1.32	1.68	1.44	3.55	1.79	1.45	1.35	1.52	1.38	1.24	1.15	1.47

Scale control. As mentioned above, geometric evaluation is not sensitive to detail changes. However, we find perceptual rendering quality is a more sensitive metric. In Table 8, Trim2DGS without scale control (§ 4.2) has significant performance drops in terms of the perceptual metric LPIPS. The qualitative results in Figure 4 confirm this. We further visualize Gaussian centers in MipNeRF360 dataset to demonstrate that our Trim2DGS can yield a more uniform distribution of Gaussians while significantly reducing the storage consumption, as can be seen in Figure 5.

Table 8: **Effectiveness of scale control on MipNeRF360.** Scale control improves performance, particularly in terms of the perceptual metric LPIPS, which is instrumental in enhancing the details.

	bicycle	flowers	garden	stump	treehill	mean	room	counter	kitchen	bonsai	mean
PSNR ↑											
2DGS	24.75	20.97	26.58	26.18	22.31	24.16	30.56	28.01	30.13	31.21	29.98
Trim2DGS	24.95	20.80	26.53	26.28	22.01	24.11	30.29	27.91	30.03	31.12	29.84
Trim2DGS (w/o. scale control)	24.82	20.91	26.59	26.18	22.14	24.13	30.62	27.99	30.02	31.03	29.92
SSIM ↑											
2DGS	0.739	0.569	0.843	0.757	0.621	0.706	0.905	0.890	0.914	0.929	0.910
Trim2DGS	0.755	0.580	0.849	0.764	0.622	0.714	0.910	0.896	0.919	0.934	0.915
Trim2DGS (w/o. scale control)	0.742	0.572	0.843	0.757	0.621	0.707	0.906	0.887	0.911	0.927	0.908
LPIPS ↓											
2DGS	0.255	0.378	0.138	0.256	0.367	0.279	0.244	0.232	0.147	0.228	0.213
Trim2DGS	0.216	0.342	0.118	0.229	0.322	0.246	0.219	0.208	0.133	0.212	0.193
Trim2DGS (w/o. scale control)	0.249	0.376	0.137	0.253	0.364	0.276	0.246	0.238	0.155	0.249	0.218

6 Conclusion and Limitations

Conclusion. In this paper, we propose TrimGS to gradually trim 3D Gaussian fields for accurate geometry representation. It features a new Gaussian contribution evaluation method and a contribution-based trimming strategy. We also conduct experimental and theoretical analyses and find that large Gaussians are hard to optimize and have limited capacities to represent geometry and appearance details. So we combine the Trimming strategy with scale control and normal regularization, achieving consistent improvement in geometry reconstruction and perceptual rendering quality.

Limitations. Although TrimGS emphasizes Gaussian trimming, it still needs geometry regularization terms such as normal consistency. The geometry regularization inevitably causes a slight drop in rendering quality compared with the original 3DGS, especially for PSNR metric in outdoor scenes as shown in Table 4. The rendering quality degradation is also observed in previous 3DGS-based reconstruction methods. This phenomenon reveals that it remains a challenge to maintain both high rendering and accurate geometry structures. We will explore this topic in our future work.

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.

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19697–19705, 2023.

[3] Hanlin Chen, Chen Li, and Gim Hee Lee. Neusg: Neural implicit surface reconstruction with 3d gaussian splatting guidance. *arXiv preprint arXiv:2312.00846*, 2023.

[4] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In *SIGGRAPH 2024 Conference Papers*. Association for Computing Machinery, 2024. doi: 10.1145/3641519.3657441.

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

[6] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023.

[7] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022.

[8] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*, 2023.

[9] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5875–5884, 2021.

[10] Wenbo Hu, Yuling Wang, Lin Ma, Bangbang Yang, Lin Gao, Xiao Liu, and Yuewen Ma. Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19774–19783, 2023.

[11] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. *arXiv preprint arXiv:2403.17888*, 2024.

[12] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 406–413, 2014.

[13] Yiming Ji, Yang Liu, Guanghu Xie, Boyu Ma, and Zongwu Xie. Neds-slam: A novel neural explicit dense semantic slam framework using 3d gaussian splatting. *arXiv preprint arXiv:2403.11679*, 2024.

[14] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH computer graphics*, 18(3):165–174, 1984.

[15] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013.

[16] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023.

[17] Jonas Kulhanek and Torsten Sattler. Tetra-nerf: Representing neural radiance fields using tetrahedra. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18458–18469, 2023.

[18] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. *arXiv preprint arXiv:2311.13681*, 2023.

[19] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.

[20] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[21] 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. *Communications of the ACM*, 65 (1):99–106, 2021.

[22] 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.

[23] Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. Reducing the memory footprint of 3d gaussian splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1):1–17, 2024.

[24] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 14335–14345, 2021.

[25] 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.

[26] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5459–5469, 2022.

[27] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Improved direct voxel grid optimization for radiance fields reconstruction. *arXiv preprint arXiv:2206.05085*, 2022.

[28] Matias Turkulainen, Xuqian Ren, Iaroslav Melekhov, Otto Seiskari, Esa Rahtu, and Juho Kannala. Dn-splatter: Depth and normal priors for gaussian splatting and meshing. *arXiv preprint arXiv:2403.17822*, 2024.

[29] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

[30] Yaniv Wolf, Amit Bracha, and Ron Kimmel. Surface reconstruction from gaussian splatting via novel stereo views. *arXiv preprint arXiv:2404.01810*, 2024.

[31] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. *Advances in Neural Information Processing Systems*, 34:4805–4815, 2021.

[32] 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*, pages 1–9, 2023.

[33] Mulin Yu, Tao Lu, Linning Xu, Lihan Jiang, Yuanbo Xiangli, and Bo Dai. Gsdf: 3dgs meets sdf for improved rendering and reconstruction. *arXiv preprint arXiv:2403.16964*, 2024.

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

[35] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[36] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

A Appendix / supplemental material

A.1 Theoretical Analysis for Gradients

In this section, the gradient dilemma discussed in § 3.2 is demonstrated in 1D space, where a common square wave function is approximated by 1D Gaussian distributions as shown in Figure 6. Let

$$f(x) = \begin{cases} 1 & x \in [2kT, (2k+1)T) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

be a square wave function with the period 2T, and $g_\sigma(x;\mu)$ be a 1D Gaussian distribution with changeable mean μ and fixed standard deviation σ. Considering $L_\sigma(\mu) = \int_{-\infty}^{+\infty} |g_\sigma(x;\mu) - f(x)|dx$, the L_1 loss between g_σ and f, we propose that $g_\sigma(x;\mu)$ with relatively smaller σ helps with faster convergence when solving the optimal μ that minimized $L_\sigma(\mu)$ by gradient descent. Specifically, with $\sigma_1 = T/4$ and $\sigma_2 = T/2$, the gradient relationship $|L'_{\sigma_1}(\mu_0)| > 2|L'_{\sigma_2}(\mu_0)|$ holds when the initial value $\mu_0 = 2kT$ is set at the start of any pulses of the square wave function.

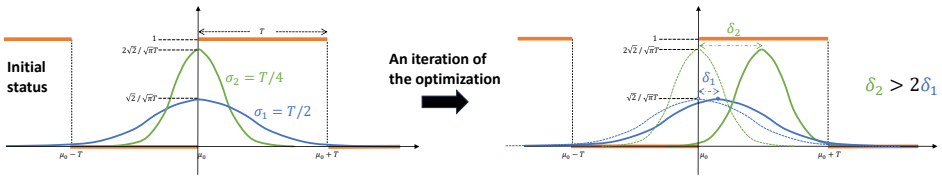


Figure 6: **A square wave function is locally approximated by Gaussian distributions**. The one with $\sigma = T/4$ gets a larger gradient than the one with $\sigma = T/2$, resulting in a faster move towards the optimal position.

We prove this by simplifying the expression of the gradient. For convenience, we only consider the finite interval $[\mu_0 - 3\sigma, \mu_0 + 3\sigma]$ due to the 3σ principle (which is also applied in the 3DGS kernel). Let I_+ and I_- denote intervals where f equals 1 and 0 respectively. The gradient can be written as

$$\begin{aligned} L'_\sigma(\mu) = \frac{\partial L_\sigma(\mu)}{\partial \mu} &= \frac{\partial}{\partial \mu} \int_{\mu_0-3\sigma}^{\mu_0+3\sigma} |g_\sigma(x;\mu) - f(x)|dx \\ &\xlongequal{\text{differentiability of } g} \int_{\mu_0-3\sigma}^{\mu_0+3\sigma} \frac{\partial |g_\sigma(x;\mu) - f(x)|}{\partial \mu} dx \\ &= \int_{I_-} \frac{\partial g_\sigma(x;\mu)}{\partial \mu} dx - \int_{I_+} \frac{\partial g_\sigma(x;\mu)}{\partial \mu} dx. \end{aligned} \tag{10}$$

Noticing the fact that $\frac{\partial g_\sigma(x;\mu)}{\partial \mu} = -\frac{\partial g_\sigma(x;\mu)}{\partial x}$ we further conclude that

$$\begin{aligned} \frac{\partial L_\sigma(\mu)}{\partial \mu} &= -\int_{I_-} \frac{\partial g_\sigma(x;\mu)}{\partial x} dx + \int_{I_+} \frac{\partial g_\sigma(x;\mu)}{\partial x} dx \\ &\xlongequal{\text{Newton-Leibniz}} g_\sigma(x;\mu)\Big|_{I_+} - g_\sigma(x;\mu)\Big|_{I_-}. \end{aligned} \tag{11}$$

Eq. (11) means that the gradient is obtained by adding and subtracting the values of Gaussian distribution at endpoints of intervals. When σ is small, these values are more likely to have the same effects on the final gradient (see Eq. (12)); when σ grows larger, more intervals are incorporated and their effects may be contradictory (Eq. (13)).

Specifically, setting $\mu = \mu_0$, when $\sigma = T/4$, we have

$$\begin{aligned} & g_{T/4}(x;\mu_0)\Big|_{I_+} - g_{T/4}(x;\mu_0)\Big|_{I_-} \\ =& (g_{T/4}(\mu_0;\mu_0 + 3T/4) - g_{T/4}(\mu_0;\mu_0)) - (g_{T/4}(\mu_0;\mu_0) - g_{T/4}(\mu_0;\mu_0 - 3T/4)) \\ =& 2(e^{-\frac{9}{2}} - 1)/(\sqrt{2\pi}T/4). \end{aligned} \tag{12}$$

When $\sigma = T/2$, we have

$$\begin{aligned}
&g_{T/2}(x;\mu_0)\Big|_{I_+} - g_{T/2}(x;\mu_0)\Big|_{I_-} \\
=&(g_{T/2}(\mu_0;\mu_0 - T) - g_{T/2}(\mu_0;\mu_0 - 3T/2) + g_{T/2}(\mu_0;\mu_0 + T) - g_{T/2}(\mu_0;\mu_0)) \\
&- (g_{T/2}(\mu_0;\mu_0) - g_{T/2}(\mu_0;\mu_0 - T) + g_{T/2}(\mu_0;\mu_0 + 3T/2) - g_{T/2}(\mu_0;\mu_0 + T)) \\
=&2(e^{-2} - 1)/(\sqrt{2\pi}T/2).
\end{aligned} \tag{13}$$

Therefore, $|L'_{T/4}(\mu_0)| > 2|L'_{T/2}(\mu_0)|$, which indicates that the Gaussian distribution with smaller σ gets larger gradient in this situation.

B Implementation

B.1 Median Depth Rendering

We follow 2DGS to render depth maps. Specifically, during volume rendering, we hypothesize that the furthest visible Gaussian is closest to the actual surface. Thus we record the depth of the Gaussian with the largest z-axis value in camera coordinates, whose transmittance T_i exceeds 0.5, as the median depth

$$d = \max_{i \in \{T_i > 0.5\}} z_i. \tag{14}$$

B.2 Point Cloud Evaluation

As we discussed in § 5.3, we propose an evaluation pattern for the quality of raw point clouds (Gaussian centers) through Chamfer Distance. Specifically, our evaluation starts with the voxelization of the raw point clouds. Within each voxel, we retain the point that is closest to the voxel's centroid, discarding the others. The method ensures a uniform distribution of point clouds, mitigating the bias that may arise from varying local densities across different Gaussian fields. Following voxelization, we leverage the ground truth point cloud provided by the DTU to compute the Chamfer Distance for the downsampled point cloud, which serves as a metric for evaluating the quality of the Gaussian centers point cloud.