Learning Differentiable Hierarchies in 3D Gaussian Splatting

Supplementary Material

6. Gradient Derivation for the Gaussian Hierarchy

In this section, we present a detailed derivation of the gradient of the loss function with respect to the Gaussian hierarchy value H_i^{exp} in DDSF rendering, i.e., the backpropagation procedure described in Sec. 3.3.

Starting from Eq. 4, the gradient can be expressed as

$$\frac{\partial L(x)}{\partial H_i^{\text{exp}}} = \frac{\partial L(x)}{\partial C(x)} \cdot \frac{\partial C(x)}{\partial m_i} \cdot \frac{\partial m_i}{\partial H_i^{\text{exp}}}, \tag{12}$$

where $C(x)$ denotes the rendered pixel color, and $m_i = \sigma(H_i^{\text{exp}}; \mu)$ is the DDSF modulation coefficient of the i-th Gaussian.

Eq. 12 clearly illustrates the chain of dependencies in the gradient:

1. $\partial L(x)/\partial C(x)$: the gradient of the loss with respect to the pixel color,
2. $\partial C(x)/\partial m_i$: the gradient of the pixel color with respect to the rendering modulation coefficient,
3. $\partial m_i/\partial H_i^{\text{exp}}$: the gradient of the modulation coefficient with respect to the Gaussian hierarchy value.

The computation of the last two terms is tightly coupled with the rendering process, as they involve the DDSF modulation of Gaussian opacities and the accumulation of colors along the ray.

6.1. Derivative of color w.r.t. modulation coefficient.

We rewrite the modulation rendering equation from Eq. 2 as follows:

$$C(x) = \sum_i T_i \left(m_i \alpha_i\right) c_i, \quad T_i = \prod_{j<i} (1 - m_j \alpha_j), \tag{13}$$

where c_i is the color, α_i the opacity, and m_i the DDSF modulation of the i-th Gaussian.

The derivative $\partial C(x)/\partial m_i$ consists of the *direct* dependence of $C(x)$ on $w_i = m_i \alpha_i$, and the *indirect* dependence through all later transmittances T_k for $k > i$.

(1) Direct contribution. The term corresponding to the i-th Gaussian contributes directly to $C(x)$ as

$$T_i (m_i \alpha_i) c_i. \tag{14}$$

Differentiating with respect to m_i gives

$$\frac{\partial}{\partial m_i} \left(T_i (m_i \alpha_i) c_i\right) = T_i \alpha_i c_i. \tag{15}$$

(2) Indirect contribution. For Gaussians $k > i$, the transmittance depends on m_i:

$$T_k = \prod_{j<k} (1 - m_j \alpha_j), \tag{16}$$

and only the factor $(1 - m_i \alpha_i)$ depends on m_i. Hence, for each $k > i$,

$$\frac{\partial}{\partial m_i} \left(T_k (m_k \alpha_k) c_k\right) = \frac{\partial T_k}{\partial m_i} (m_k \alpha_k) c_k = -\frac{\alpha_i \, T_k (m_k \alpha_k) c_k}{1 - m_i \alpha_i}. \tag{17}$$

Summing over all $k > i$ gives the total indirect contribution.

(3) Combine direct and indirect terms. Adding the direct and indirect contributions, we obtain

$$\frac{\partial C(x)}{\partial m_i} = \alpha_i \left(T_i c_i - \sum_{k>i} \frac{T_k (m_k \alpha_k) c_k}{1 - m_i \alpha_i} \right), \tag{18}$$

which is exactly the expression used in Eq. 5.

6.2. Derivative of modulation coefficient w.r.t. hierarchy value.

Given that the DDSF is implemented as a decreasing sigmoid applied to the Gaussian hierarchy, the modulation coefficient of the i-th Gaussian is

$$m_i = \sigma_{\text{dec}}(H_i^{\text{exp}}; \mu) = \frac{1}{1 + \exp\left[\beta (H_i^{\text{exp}} - \mu)\right]}, \tag{19}$$

where μ is the activation center and $\beta > 0$ controls the sharpness of the transition.

The derivative of m_i with respect to H_i^{exp} is

$$\frac{\partial m_i}{\partial H_i^{\text{exp}}} = -\beta \, m_i (1 - m_i), \tag{20}$$

which is negative due to the decreasing nature of the sigmoid.

Importantly, this gradient attains its largest magnitude near $H_i^{\text{exp}} \approx \mu$, and quickly decays toward zero as H_i^{exp} moves away from μ, effectively focusing the hierarchy updates on Gaussians within the active region.

7. Additional Experiments

7.1. Comparison with PRoGS-based Gaussian Hierarchy Ranking

PRoGS [32] is a method that manually defines Gaussian contributions and ranks Gaussians accordingly. Specifically, it computes the contribution of each Gaussian as the

Table 5. Comparison of LoD Rendering Across Different Splat Percentages Using PRoGS and Default Gaussian Hierarchy. PRoGS can improve the hierarchy-based rendering of classical 3DGS, but it actually degrades the rendering quality for our model. Our approach achieves the best performance by utilizing the learned hierarchy.

splats %	**Method**	Mip-NeRF360			Tanks&Temples			Deep Blending		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
25	MCMC	18.18	0.61	0.36	15.57	0.57	0.37	19.60	0.69	0.44
	MCMC-PRoGS	25.28	0.79	0.19	22.19	0.79	0.19	25.68	0.83	0.21
	Δ	+7.10	+0.18	-0.17	+6.62	+0.22	-0.18	+6.08	+0.14	-0.23
	Ours	28.35	0.86	0.13	23.64	0.84	0.14	26.79	0.85	0.21
	Ours-PRoGS	25.75	0.81	0.18	22.35	0.80	0.18	25.80	0.82	0.22
	Δ	-2.60	-0.05	+0.05	-1.29	-0.01	+0.04	-0.99	-0.03	+0.01
50	MCMC	23.45	0.77	0.21	19.96	0.74	0.21	24.46	0.79	0.28
	MCMC-PRoGS	28.57	0.86	0.13	23.93	0.84	0.13	26.99	0.84	0.19
	Δ	+5.12	+0.09	-0.08	+3.97	+0.10	-0.08	+2.53	+0.05	-0.09
	Ours	29.47	0.89	0.11	24.20	0.86	0.12	26.92	0.85	0.19
	Ours-PRoGS	28.79	0.87	0.14	24.02	0.85	0.13	26.47	0.84	0.20
	Δ	-0.68	-0.02	+0.03	-0.18	-0.01	+0.01	-0.45	-0.01	+0.01
75	MCMC	27.26	0.85	0.14	22.72	0.82	0.14	26.33	0.83	0.21
	MCMC-PRoGS	29.65	0.88	0.11	24.17	0.86	0.12	27.01	0.85	0.19
	Δ	+2.39	+0.03	-0.03	+1.45	+0.04	-0.02	+0.68	+0.02	-0.02
	Ours	29.68	0.90	0.11	24.26	0.87	0.12	26.94	0.86	0.19
	Ours-PRoGS	29.62	0.90	0.11	24.25	0.86	0.12	26.90	0.85	0.19
	Δ	-0.06	0.00	0.00	-0.01	-0.01	0.00	-0.04	-0.01	0.00
100	MCMC	29.73	0.89	0.11	24.21	0.86	0.12	27.07	0.85	0.19
	MCMC-PRoGS	29.73	0.89	0.11	24.21	0.86	0.12	27.07	0.85	0.19
	Δ	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Ours	29.74	0.90	0.10	24.27	0.87	0.12	27.02	0.86	0.18
	Ours-PRoGS	29.74	0.90	0.10	24.27	0.87	0.12	27.02	0.86	0.18
	Δ	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

sum over all pixels in the training set of the product of its opacity and accumulated transmittance:

$$B_i = \sum_{v \in V} \sum_{p \in P} T_i \, \alpha_i, \tag{21}$$

where v denotes a viewpoint in the training set and p denotes a pixel in that viewpoint. The Gaussians are then sorted from high to low to obtain a hierarchy. Essentially, PRoGS is a post-processing method based on hand-crafted rules. Given a trained Gaussian model and the training dataset, it computes the Gaussian hierarchy without altering the original model parameters.

Since the original PRoGS paper does not provide quantitative results or open-source code, we re-implemented the method and applied it to both 3DGS-MCMC [10] and our approach for comparison. All experimental settings follow those reported in the [10] paper.

We evaluate on 3 datasets by sampling Gaussians at different percentages according to both default hierarchy and the PRoGS-based hierarchy, and report the averaged results across each dataset. The results are summarized in Tab. 5. Our approach achieves the best performance by utilizing the learned hierarchy.

It can be observed that the PRoGS-based ordering significantly improves the rendering quality of Gaussian subsets for 3DGS-MCMC. However, for our model, the PRoGS-based ordering actually decreases the rendering performance of Gaussian subsets. This suggests that defining the hierarchy according to Gaussian opacity over the training set, as done in PRoGS, is somewhat reasonable and can effectively enhance progressive rendering compared to the default time-ordered arrangement in classical 3DGS.

Nevertheless, the PRoGS hierarchy is computed in a one-shot manner based on individual Gaussian contributions. It does not guarantee that subsets of Gaussians with higher contributions achieve optimal rendering performance, nor can it correct the hierarchy relations. In contrast, our approach employs a learning-based automatic hierarchy optimization mechanism, which integrates a differentiable hierarchy into the rendering equation through modulation of the Gaussian opacity, enabling a globally optimized rendering effect across all Gaussians.

7.2. Ablation for Different Decreasing Step Function.

The Differentiable Decreasing Step Function (DDSF) is a monotonic and continuous function that transitions from 1 to 0 with a narrow bandwidth. It assigns large gradients to Gaussians within the transition (active) region, while producing near-zero gradients elsewhere. In our method, we adopt the standard Sigmoid function as the DDSF, though in principle any function with similar characteristics can support effective hierarchy learning. We therefore conduct an ablation study comparing different DDSF designs.

Specifically, we evaluate the *Hard Step* (i.e., the standard step-down function), the *Piecewise Linear Step*, and the *Tanh* function on the MipNeRF360 [1] and Tank & Temple [11] datasets. To ensure fair comparison, we adjust all functions to have the same activation bandwidth (defined as $m = 0.05$–0.95), set to 0.60. The experimental results are shown in Tab. 6.

From the results, we observe that all step functions except the *Hard Step* (which has zero transition bandwidth and relies solely on the balance loss to maintain a stable hierarchy) achieve similarly strong performance. This indicates that DDSF is highly versatile: it enables training to both preserve high rendering quality and effectively learn the Gaussian hierarchy.

Table 6. Ablation of step-function choices on different datasets. All results are averaged over splatting ratios of 25%, 50%, 75%, and 100% within each dataset.

Method	Mip-NeRF360			Tanks&Temples		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Sigmoid($\beta = 10$)	29.31	0.88	0.11	24.09	0.86	0.13
Hard Step	21.82	0.72	0.28	18.67	0.71	0.27
Linear Step	29.25	0.88	0.12	24.06	0.85	0.13
Tanh($\beta = 5.0$)	29.29	0.88	0.12	23.91	0.85	0.13

8. Discussion and Future Work

Our method aims to provide a general framework for learning Gaussian hierarchies in 3DGS. By introducing a hierarchy feature into the classical 3DGS representation and incorporating hierarchy-aware training, it enables the structured organization of Gaussians.

Compared to other approaches, our method imposes minimal assumptions: the hierarchy is treated as a bounded continuous real-valued variable with a specified range, without any additional constraints. This design grants our approach strong generality. We outline the following directions for future improvements.

Online Scene Reconstruction. For online reconstruction tasks, such as SLAM, the frontend devices used for image capture and pose estimation typically have limited storage and computational resources. This necessitates that the backend-generated scene representations support multi-resolution expressivity, allowing frontend devices to access real-time, customized models and perform budget-based pruning or rendering.

Existing multi-stage LoD approaches, such as CLOD and LapisGS, are generally suitable only for offline tasks, since their hierarchy construction is decoupled from model training, preventing real-time multi-resolution reconstruction. In contrast, our method integrates hierarchy learning directly into the scene construction process within a unified training framework, introducing minimal additional computational cost, and is therefore applicable to online reconstruction scenarios.

Alternative Hierarchy Structures. Our method essentially represents the Gaussian hierarchy along a bounded, differentiable one-dimensional axis. However, hierarchy can be defined using a variety of data structures, such as binary trees [9] or octrees [24]. As long as the hierarchy representation is continuous and differentiable, and a hierarchy-based rendering rule is specified according to the data structure, we can design a framework that can optimize the Gaussian hierarchy under these structures, enabling a learning-based hierarchical 3DGS scene representation.

Furthermore, our hierarchy learning framework can be extended to large-scale scenes by incorporating scene partitioning or similar techniques. In large scenes, the number of Gaussians increases dramatically, which makes hierarchy training and hierarchy-based rendering computationally inefficient. Effective scene partitioning can be used to preselect relevant Gaussians for hierarchy learning, reducing unnecessary training and computational overhead.