NoiseGS: Boosting 3D Gaussian Splatting with Positional Noise for Large-Scale Scene Rendering

Minseong Kweon[1], Kai Cheng[2], Xuejin Chen[†2], and Jinsun Park[†1]

[1]Pusan National University, South Korea
[2]University of Science and Technology of China, China

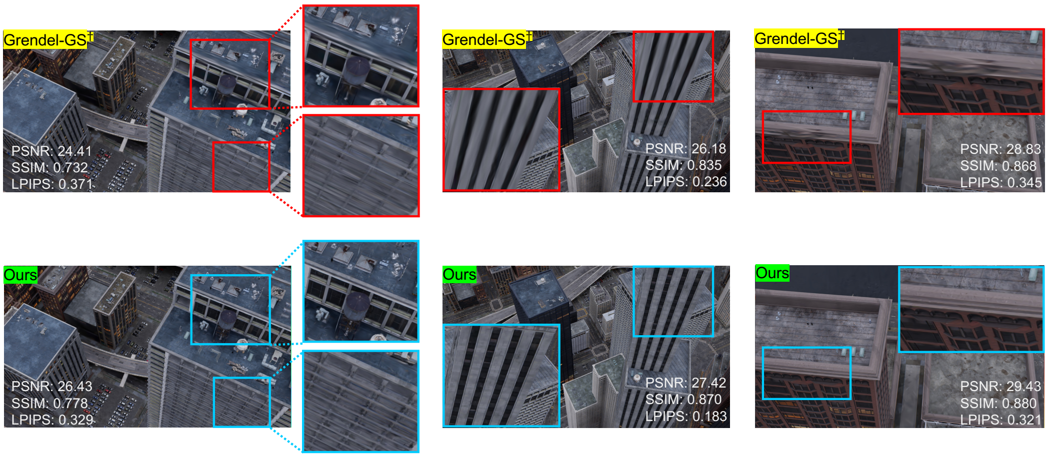


Figure 1: *The proposed NoiseGS facilitates improved scene reconstruction by enhancing the optimization of 3D Gaussians.*

Abstract

3D Gaussian Splatting (3DGS) efficiently renders 3D spaces by adaptively densifying anisotropic Gaussians from initial points. However, in complex scenes such as city-scale environments, large Gaussians often overlap with high-frequency regions rich in edges and fine details. In these areas, conflicting per-pixel gradient directions cause gradient cancellation, reducing the overall gradient magnitude and potentially causing Gaussians to remain trapped in suboptimal positions even after densification. To address this, we propose NoiseGS, a novel approach that integrates randomized noise injection into 3DGS, guiding suboptimal Gaussians selected for densification toward more optimal positions. In addition, to mitigate the instability caused by oversized Gaussians, we introduce an ℓ_p-penalization on the scale of Gaussians. Our method integrates seamlessly with existing heuristic-based optimization and demonstrates strong generalization in reconstructing complex scenes such as MatrixCity and Building.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Point-based models; Computer vision;*** *Machine learning approaches;*

1. Introduction

3D scene reconstruction is a critical domain for understanding and modeling the real world within computational geometry. Among the various approaches, 3D Gaussian Splatting (3DGS) [KKLD23] has made significant advancements in this domain by effectively modeling scenes using an explicit representation. 3DGS encodes scenes as sets of anisotropic 3D Gaussians, defined by covariance, opacity, and center position, enabling efficient rasterization by projecting 3D Gaussians onto the 2D screen space for accurate scene representation. However, scaling 3DGS to large and complex scenes remains challenging due to sparse data distribution and intricate scene structures. Recently, Hierarchy-GS [KMK*24] and CityGaussian [LLF*25] have proposed effective approaches for handling large-scale scenes using a divide-and-conquer strategy. Grendel-GS [ZWL*24] introduces a novel framework that

† Corresponding authors

DOI: 10.2312/egs.20251042

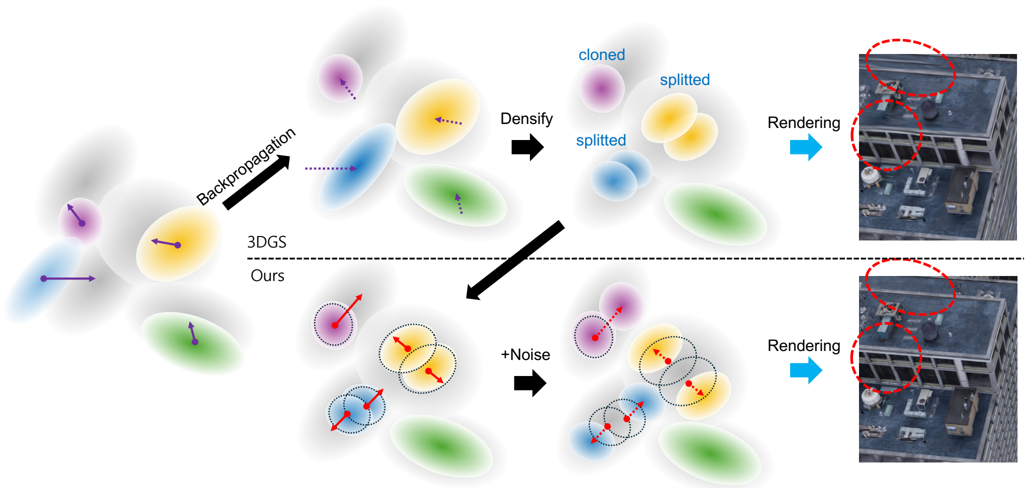


Figure 2: *Our density control strategy with positional noises. Purple represents the Gaussian to be cloned, while blue and yellow represent the Gaussians to be split. The red and purple arrows indicate the noise direction and the inverse direction of the gradient, respectively. The gray region represents the ground truth surface. Note that the cloned one stays in place, while the split ones are slightly randomly offset.*

enables training 3DGS on multi-GPU systems. Additionally, researchers have focused on the impact of suboptimal Gaussians in complex views, which often obscure fine details and cause blurred renderings. AbsGS [YLL*24] introduced the sum of absolute gradients to prevent bidirectional cancellation caused by different optimization directions across pixels within a Gaussian. Trim-GS [FYL*24] further demonstrated that large Gaussian sizes contribute to noisy gradients and emphasized the importance of maintaining smaller Gaussian scales for better representation of complex scenes. While these approaches contribute to more adaptive density control during optimization, there remains room for improvement in handling Gaussian distributions after densification. Drawing inspiration from advanced machine learning algorithms [STK*17, CWS*20], which enhance generalization performance through noise injection techniques, we introduce random noise to adjust the Gaussian center positions, enabling suboptimal Gaussians to escape noisy regions. While MCMC-GS [KRS*24] demonstrated the potential of noise in this context, it did not incorporate this concept into adaptive density control. Similar to how Gaussian Noise Injections (GNIs) [CWS*20] leverages noise to regulate high-frequency components and noise sensitivity, we propose adding noise to the center positions of Gaussians that cover high-frequency regions with many edges. Moreover, to address the instability caused by oversized Gaussians, we design an ℓ_p-penalization approach that adaptively constrains the scale of Gaussians during densification and fine-tuning. Our method seamlessly integrates into the 3DGS densification pipeline, enabling effective optimization of suboptimal and oversized Gaussians in complex scenes via randomized noise injection and scale regularization.

2. Method

To demonstrate the effectiveness of positional noise in scenes with complex structures, we utilized large-scale scenes. Accordingly, to efficiently train such data, we modified and employed the distributed 3DGS approach, Grendel-GS [ZWL*24].

2.1. Randomized positional noise

Figure 2 illustrates the process of adding positional noise to 3D Gaussians during the densification. During the training of complex scenes, suboptimal large Gaussians with small gradients may remain trapped in high-frequency regions, as densification fails to induce meaningful movement. When large Gaussians fail to converge to their optimal positions and remain stuck, the affected regions appear blurred, leading to a decline in rendering quality. To address this, we introduce random directional noise to the centers of Gaussians selected for densification. This approach allows unstable Gaussians trapped in high-frequency areas to escape their static state, thereby facilitating position optimization. At each densification step, a noise vector $\mathbf{d} = (x, y, z)$ is randomly sampled as:

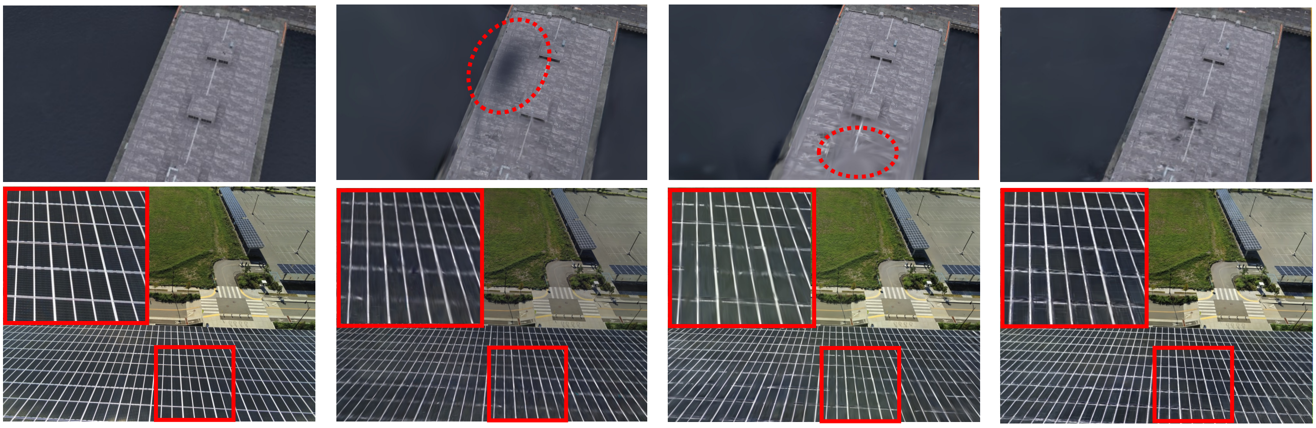
$$x = r\cos\theta\cos\phi, \quad y = r\sin\theta\cos\phi, \quad z = r\sin\phi. \tag{1}$$

Notably, the angles and radius are sampled according to $\theta \sim \mathcal{U}(0, 2\pi)$, $\phi \sim \mathcal{U}(-\pi/2, \pi/2)$, and $r \sim \mathcal{N}(0, 1)$. After that, we assign weights to the generated noise such that its magnitude increases when the sum of screen point gradients and opacity are small, and decreases otherwise. The weights are computed as:

$$w_{grad} = 1/\big(1 + \exp(a(\|\mathbf{g}\| - 2\tau_{grad}))\big), \tag{2}$$

$$w_{op} = 1/(1 + \exp(a(o - \tau_{op}))), \tag{3}$$

where o, $\mathbf{g}$, and a represent the Gaussians' opacity, positional gradient, and slope parameters of the sigmoid function, respectively. Note that τ_{grad} and τ_{op} are the same as the densification and pruning thresholds of gradient and opacity. The reason for decreasing the influence of noise when the sum of gradients is large is that it indicates well-aligned optimization directions, allowing training alone to optimize the Gaussian's position, making noise unnecessary in such cases. If the sum of gradient values is sufficiently below the densification threshold, the Gaussian is not selected for densification, and noise is therefore not applied. Our design ensures that



Ground Truth | CityGS | Grendel-GS† | Ours

Figure 3: *Qualitative comparison on Block_All (1st row) and Building (2nd row) datasets.*

unnecessary noise is not introduced to well-optimized Gaussians. Following that, the weighted noise vector ϵ' is then calculated as:

$$\epsilon' = w_{grad} \cdot w_{op} \cdot \eta_{\text{noise}} \cdot \eta_{\mu} \cdot \mathbf{d}, \tag{4}$$

where η_{noise}, η_{μ}, and $\mathbf{d}$ represent the learning rate for noise, the learning rate for position and the randomly sampled direction, respectively. After that, the noise is transformed by the covariance matrix to reflect the shape of the current Gaussian:

$$\epsilon = \Sigma \cdot \epsilon', \tag{5}$$

where Σ is the covariance matrix.

2.2. ℓ_p-Penalization for scale

Building on the insights from Trim-GS [FYL*24], which emphasized the benefits of small Gaussians, and MCMC-GS [KRS*24], which applied ℓ_1-regularization to the covariance matrix, we extend these ideas by introducing a ℓ_p-penalization for Gaussians' scales. During the densification process, we keep the p value close to 1, leveraging ℓ_1-regularization to promote sparsity in 3D Gaussians and eliminate unnecessary values. As fine-tuning progresses, the p value is gradually increased to 2, enhancing the regularization effect and guiding the learning process toward minimizing all scales. Our ℓ_p-penalization is incorporated into the standard L1 and SSIM loss [KKLD23] functions, denoted by $\mathcal{L}_1$ and $\mathcal{L}_{D-SSIM}$, as follows:

$$\mathcal{L} = \lambda_{\text{D-SSIM}} \cdot \mathcal{L}_{D-SSIM} + (1 - \lambda_{\text{D-SSIM}})\mathcal{L}_1 + \lambda_s \frac{1}{N} \sum_{i=1}^{N} |s_i|^p, \tag{6}$$

where λ_{D-SSIM}, λ_s, N, and s_i are the weight of D-SSIM loss, the weight of scale regularization, the total number of Gaussians, and the scale value of the i^{th} Gaussian, respectively.

2.3. Implementation details

We conducted our primary experiments on large-scale scenes to explore the effectiveness of random noises for optimization in complex environments. We utilize the MatrixCity's *Block_All* [LJX*23] and Mill19's *Building* [TRS22] scene datasets. The *Block_All* scene consists of 5,620/741 train/test images, covering an area of 2.7 km^2, while the *Building* scene contains 1,920/20 train/test images, covering an area of 0.125 km^2. We

Scenes	Block_All				Building			
	PSNR ↑	SSIM ↑	LPIPS ↓	training	PSNR ↑	SSIM ↑	LPIPS ↓	training
3DGS [KKLD23]	25.95	0.798	0.378	1h 39m	20.98	0.718	0.375	33m
AbsGS [YLL*24]	27.10	0.844	0.233	1h 56m	21.27	0.760	0.263	1h 6m
CityGS [LLF*25]	27.40	0.864	0.205	9h 37m	22.13	0.783	0.241	6h 34m
Grendel-GS† [ZWL*24]	27.59	0.861	0.266	2h 48m	22.20	0.770	0.314	57m
Ours	27.73	0.867	0.251	2h 53m	22.76	0.788	0.287	1h 1m

Table 1: *Benchmarking on Block_All and Building scenes, with a comparison of 1st, 2nd, and 3rd best performances. † indicates our modified version.*

follow MegaNeRF [TRS22] to evaluate the rendering performance using PSNR, SSIM, and LPIPS as metrics.

Our baseline follows the Grendel-GS [ZWL*24] approach, which redistributes Gaussians across multiple GPUs for multi-GPU training. In our modified version, we evenly divide the entire scene, set the batch size to 1, do not perform pixel-wise splitting, and accelerate training using fused SSIM [MGK*24]. During fine-tuning, it incorporates pruning as in LightGaussian [FWW*23]. The positional learning rate η_μ is configured to 1.6×10^{-5}, as suggested for large-scale scenes [KKLD23]. Regarding our hyperparameters, η_{noise} is set to 5×10^5, and λ_s was set to 0.01. For training, eight and four RTX A6000 GPUs are utilized for the MatrixCity and Mill19 datasets, respectively. Considering GPU capacity, the densification interval is set to 200 for single-GPU and 100 for multi-GPU methods, with densification stopping in single-GPU settings near the out-of-memory limit. Training spans 30K iterations, split into 15K for densification and 15K for fine-tuning. All other hyperparameters follow those of 3DGS [KKLD23].

2.4. Performance Analysis

Table 1 presents the benchmarking results for large-scale 3D reconstruction. Compared to the baseline, modified Grendel-GS [ZWL*24], our method achieves overall performance improvements. Notably, even with only 30K training iterations, our method outperforms CityGS [LLF*25] in PSNR and SSIM. While our method shows slightly worse LPIPS scores compared to CityGS, CityGS requires significantly increased training time to improve the perceptual quality through fine-grained block-wise tuning. In contrast, while our method can be integrated with CityGS, we adopt a multi-GPU scene-level optimization framework in this work for a

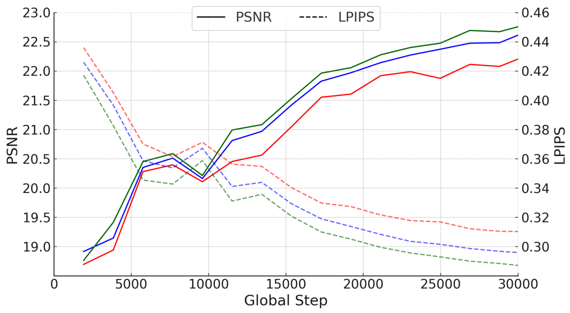


Figure 4: *Performance trends illustrating improvements with + positional noise and + ℓ_p-penalization, compared to the baseline.*

more practical balance between efficiency and reconstruction quality. Compared to AbsGS [YLL*24], our method shows slightly lower performance in LPIPS but achieves better results in PSNR and SSIM. This indicates that adding noises to the Gaussians' center positions can be more effective than using absolute gradients. As shown in Fig. 3, our method demonstrates superior rendering quality compared to previous state-of-the-art approaches, particularly in regions with complex textures. This suggests that the incorporation of noise and scale constraints holds promise for large-scale 3D reconstruction.

2.5. Ablation studies

Figure 4 illustrates the performance improvement trends observed at intermediate stages of training. Compared to the baseline, significant enhancements can be seen when the proposed method is applied. Figure 5 shows the rendering results of the ablation study. When the proposed method is applied, progressive improvements can be observed in fine-grained details. For further results on smaller scenes, please refer to the additional material.

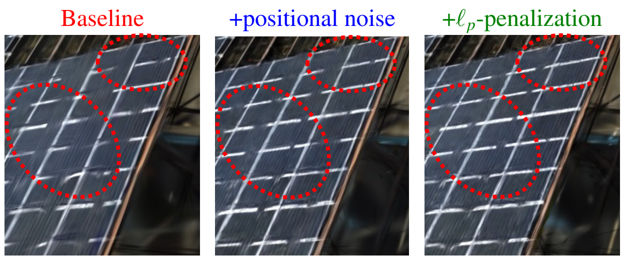


Figure 5: *Ablation study on the Building scene.*

3. Conclusion

We propose NoiseGS, a noise injection algorithm that enhances 3D rendering performance by adding randomized positional noise to 3D Gaussians and dynamically adjusting their positions based on screen space gradients and opacity. Additionally, we incorporate ℓ_p penalization on Gaussian's scale, where the sparsity level and regularization strength are adaptively modulated throughout the densification and fine-tuning stages to tackle the optimization challenges of large-scale Gaussians. Experiment results on large-scale scene reconstruction show that our method achieves competitive performance compared to previous approaches. This highlights its potential to effectively optimize the positions of suboptimal and oversized Gaussians in complex scenes through randomized noise injection and scale regularization.

4. Acknowledgments

This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (RS-2024-00344883, 50%), in part by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (RS-2024-00358935, 40%), and in part by Institute of Information & communications Technology Planning & Evaluation(IITP) under the Artificial Intelligence Convergence Innovation Human Resources Development grant funded by the Korea government(MSIT) (IITP-2025-RS-2023-00254177, 10%).

References

[CWS*20] Camuto A., Willetts M., Simsekli U., Roberts S. J., Holmes C. C.: Explicit regularisation in gaussian noise injections. *Advances in Neural Information Processing Systems 33* (2020), 16603–16614. 2

[FWW*23] Fan Z., Wang K., Wen K., Zhu Z., Xu D., Wang Z.: Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245* (2023). 3

[FYL*24] Fan L., Yang Y., Li M., Li H., Zhang Z.: Trim 3d gaussian splatting for accurate geometry representation. *arXiv preprint arXiv:2406.07499* (2024). 2, 3

[KKLD23] Kerbl B., Kopanas G., Leimkuehler T., Drettakis G.: 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG) 42*, 4 (2023), 1–14. 1, 3

[KMK*24] Kerbl B., Meuleman A., Kopanas G., Wimmer M., Lanvin A., Drettakis G.: A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics (TOG) 43*, 4 (2024), 1–15. 1

[KRS*24] Kheradmand S., Rebain D., Sharma G., Sun W., Tseng J., Isack H., Kar A., Tagliasacchi A., Yi K. M.: 3d gaussian splatting as markov chain monte carlo. *arXiv preprint arXiv:2404.09591* (2024). 2, 3

[LJX*23] Li Y., Jiang L., Xu L., Xiangli Y., Wang Z., Lin D., Dai B.: Matrixcity: A large-scale city dataset for city-scale neural rendering and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 3205–3215. 3

[LLF*25] Liu Y., Luo C., Fan L., Wang N., Peng J., Zhang Z.: Citygaussian: Real-time high-quality large-scale scene rendering with gaussians. In *European Conference on Computer Vision* (2025), Springer, pp. 265–282. 1, 3

[MGK*24] Mallick S. S., Goel R., Kerbl B., Steinberger M., Carrasco F. V., De La Torre F.: Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers* (2024), pp. 1–11. 3

[STK*17] Smilkov D., Thorat N., Kim B., Viégas F., Wattenberg M.: Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* (2017). 2

[TRS22] Turki H., Ramanan D., Satyanarayanan M.: Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 12922–12931. 3

[YLL*24] Ye Z., Li W., Liu S., Qiao P., Dou Y.: Absgs: Recovering fine details in 3d gaussian splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia* (2024), pp. 1053–1061. 2, 3, 4

[ZWL*24] Zhao H., Weng H., Lu D., Li A., Li J., Panda A., Xie S.: On scaling up 3d gaussian splatting training. *arXiv preprint arXiv:2406.18533* (2024). 1, 2, 3