

1 A Broader Impacts and Limitation

2 A.1 Broader Impacts

3 Our method focuses on exploring a novel representation for 3D reconstruction and improving the
4 rendering quality of 3D reconstruction. It does not have a direct social impact. However, it may trigger
5 a chain reaction in some downstream applications. For example, when using 3DGS to reconstruct
6 digital humans, if the rendering quality is high enough to make it difficult for people to distinguish
7 between real and fake, it may raise social and ethical concerns, as well as issues related to the forgery
8 of personal portrait assets. Our method can improve the quality of 3D rendering in certain scenarios,
9 which may exacerbate such risks. This is a concern that any technology should take into account, and
10 we urge users to carefully consider the potential consequences when applying our method.

11 A.2 Limitation

12 Our method is applicable to various kernel representations, not limited to Gaussian kernels. In
13 this paper, we only explore the Gaussian kernel and the generalized Gaussian kernel; however,
14 our approach is, in fact, generalizable to the vast majority of kernels. Many existing works have
15 investigated alternative kernel representations [1, 2], many of which are based on 2DGS [3] to
16 leverage its inherent 2D structure and the convenience of projection-based computation. Similarly,
17 our method adopts a 2D representation and is compatible with many of these approaches, which
18 could provide significant inspiration for future research in this field.

19 Our method focuses on improving the quality of 3D rendering without incorporating normals to
20 enhance the geometric consistency, such as [3, 4]. This may cause difficulties in applications like
21 surface reconstruction and mesh extraction.

22 Our method does not address lighting and physical reflection. Prior works [5] have shown that fitting
23 complex lighting conditions can significantly improve rendering quality. We consider this to be a
24 valuable direction for future research.

25 B More Technical Details

26 B.1 Overview

27 This appendix is organized as follows: (Sec. B.2) Experimental Setup; (Sec. B.3) Network Architec-
28 ture; (Sec. B.4) Pseudocode of Our Method; (Sec. B.5) Proof of Unbiased Projection; (Sec. B.6) More
29 Details of SAP-Ges; (Sec. B.7) Experiments on Computational Resources; (Sec. B.8) Discussion of
30 Densification Strategies; (Sec. B.9) View-dependent Gaussian Attribute Decoding; (Sec. B.10) More
31 Ablation on Kernels; (Sec. B.11) Additional Results;

32 B.2 Experimental Setup

33 **Experimental Setup on the Mip-NeRF360 Dataset.** It is worth noting that in Scaffold-GS [6],
34 images with resolutions larger than 1600 are downsampled to 1600. In contrast, some methods such
35 as 3DGS-MCMC [7] downsample outdoor scenes by a factor of 4 and indoor scenes by a factor of 2.
36 These differences in preprocessing can lead to slight variations in the results. In this paper, we follow
37 the experimental setup of Scaffold-GS.

38 **Hyperparameter Settings.** Most of our hyperparameter settings follow those of Scaffold-GS.
39 Specifically, we used 10 neural Gaussians for each anchor; We set the threshold for the average
40 gradient magnitude (τ_1) to 0.0002, and the threshold for the maximum gradient magnitude (τ_2) to
41 0.0015; The initial learning rate of our 3D-consistent decoder is set at 0.004 and gradually annealed
42 to 0.00004 during training; the learning rate for the features is set to 0.0025. All other hyperparameter
43 settings follow those of Scaffold-GS to ensure a fair comparison.

44 B.3 Network Architecture

45 Our 3D-Consistent Decoder architecture is illustrated in Fig. 1. We condition the network on the
46 viewing direction, which is first encoded and passed through a linear layer to generate modulation

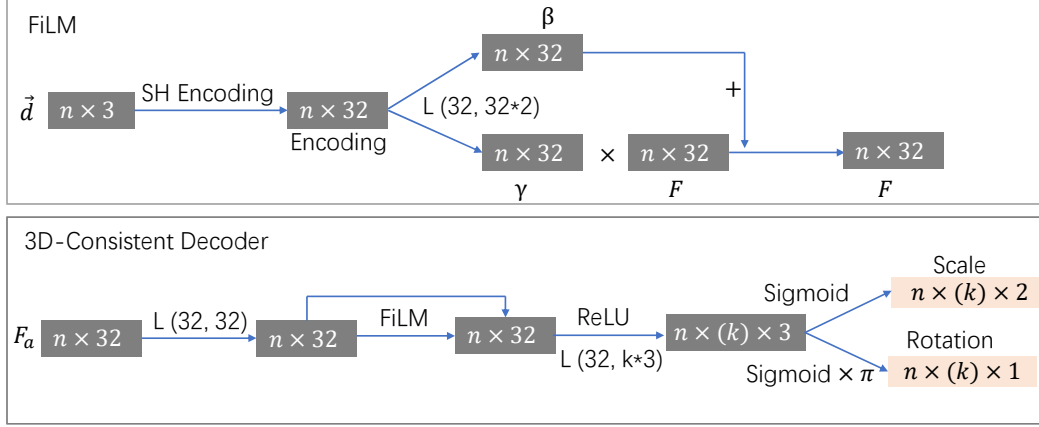


Figure 1: **Illustration of the 3D-Consistent Decoder.** The top shows the direction modulator, while the bottom depicts the full decoder architecture. Here, $L(\cdot)$ denotes a linear layer.

parameters. These parameters are then used to linearly modulate the features to produce the final output. The rest of our network architecture remains consistent with that of Scaffold-GS [6].

B.4 Pseudocode of Our Method

Algorithm 1 Screen-Aligned Primitives

Require: Decoder \mathcal{D} , Anchor Feature $\hat{\mathbf{f}}_a$, Anchor Position x , direction \mathbf{d}

- 1: Gaussian attributes $c, o, \theta, s \leftarrow \mathcal{D}(\hat{\mathbf{f}}_a, \mathbf{d})$
 - 2: Depth Sort $t \leftarrow \text{Sort}(x)$
 - 3: $\text{cov}_{\text{camera}} \leftarrow F(\theta, s)$ ▷ Construct camera-space covariance matrix
 - 4: $\text{cov}_{\text{screen}} \leftarrow \frac{f^2}{t^2} \cdot \text{cov}_{\text{camera}}$ ▷ Screen-space covariance matrix based on focal and depth
 - 5: Output: Rendered pixel color $\leftarrow R(\text{cov}_{\text{screen}}, c, o, t)$
-

We briefly present the pseudocode of our pipeline. For a given viewing direction and an anchor feature, we use the decoder to decode the Gaussian attributes. Then, we sort the anchors by their camera-space depth. We construct the covariance matrix directly in camera space using the rotation and scale parameters instead of world space. Since our primitives are planar, we obtain the screen-space covariance matrix by directly scaling according to the ratio of depth and focal length. Finally, alpha-blending rendering is performed.

B.5 Proof of Unbiased Projection

As shown in Fig. 2, 3DGS [8] introduces errors due to the use of an approximate affine projection, whereas our method can effectively avoid this issue.

In our projection framework, the function values of the screen-space kernel and the view-space kernel are equivalent. If we ignore the z -axis depth, assuming the view-space mean point is (μ_x^v, μ_y^v) and the screen-space mean point is (μ_x^s, μ_y^s) . For a given depth t and horizontal and vertical focal lengths $f_x = \frac{\text{width}}{2 \cdot \tan \text{Fov}_x}$, $f_y = \frac{\text{height}}{2 \cdot \tan \text{Fov}_y}$, the transformation matrix from space to screen, based on our geometric scaling projection, can be expressed as:

$$P = \begin{bmatrix} \frac{f_x}{t} & 0 \\ 0 & \frac{f_y}{t} \end{bmatrix}. \quad (1)$$

The relationship between the covariance matrices Σ^v and Σ^s can be expressed as:

$$\Sigma^s = P \Sigma^v P^T. \quad (2)$$

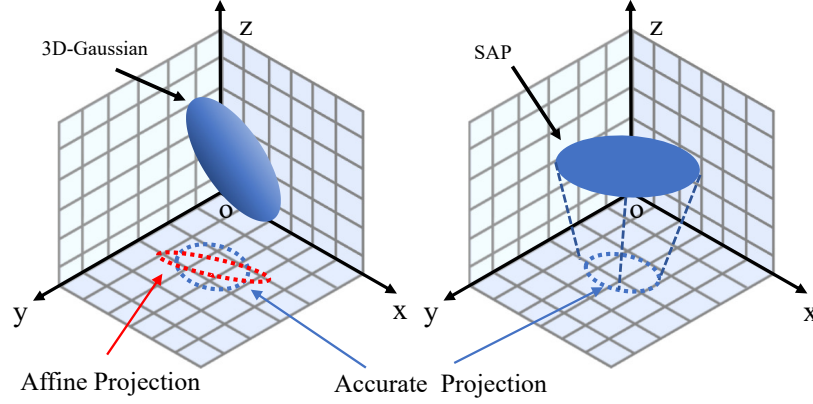


Figure 2: **A simplified diagram of different projection methods.** On the left is the projection process of the 3D Gaussian sphere: the blue represents the standard perspective projection, while the red represents the affine projection approximated by the Jacobian matrix. On the right is our 2D slice transformation, where our projection is an affine transformation that achieves the same effect as the exact projection.

For the camera model established in 3DGS, the perspective projection matrix $M_{\text{perspective}}$ can be expressed as:

$$\begin{bmatrix} \frac{2*z_{near}}{right-left} & 0 & \frac{right+left}{right-left} & 0 \\ 0 & \frac{2*z_{near}}{top-bottom} & \frac{top+bottom}{top-bottom} & 0 \\ 0 & 0 & \frac{z_{far}}{z_{far}-z_{near}} & \frac{-z_{far}*z_{near}}{z_{far}-z_{near}} \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (3)$$

With the additional conditions: bottom = -top; left = -right; $z_{near} = 0.1$; $z_{far} = 100$; top = $\tan\text{Fovy} \cdot z_{near}$; right = $\tan\text{Fovy} \cdot z_{near}$. We can simplify it to:

$$\begin{bmatrix} \frac{1}{\tan\text{Fovy}} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan\text{Fovy}} & 0 & 0 \\ 0 & 0 & 1 & -0.01 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (4)$$

Therefore, for any view-space coordinates $(x^v, y^v, t, 1)$, the transformation to screen space can be written as:

$$\begin{bmatrix} x^n \\ y^n \\ z^n \\ w^n \end{bmatrix} = M_{\text{perspective}} \begin{bmatrix} x^v \\ y^v \\ t \\ 1 \end{bmatrix} \quad (5)$$

where $w^n = t$ and (x^n, y^n, z^n, w^n) are the NDC coordinates.. Therefore, the normalized coordinates x^n and y^n in screen space are given by $\frac{x^n}{t}$ and $\frac{y^n}{t}$, respectively. That is,

$$\begin{bmatrix} x^n \\ y^n \end{bmatrix} = \begin{bmatrix} \frac{x^v}{\tan\text{Fovy}*t} \\ \frac{y^v}{\tan\text{Fovy}*t} \end{bmatrix} \quad (6)$$

According to the defined NDC transformation:

$$\begin{bmatrix} x^s \\ y^s \end{bmatrix} = \begin{bmatrix} \frac{(x^n+1)*\text{width}-1}{2} \\ \frac{(y^n+1)*\text{height}-1}{2} \end{bmatrix} \quad (7)$$

We can obtain:

$$\begin{bmatrix} x^s \\ y^s \end{bmatrix} = \begin{bmatrix} \frac{f_x}{t} * x^v + \frac{\text{width}}{2} - 0.5 \\ \frac{f_y}{t} * y^v + \frac{\text{height}}{2} - 0.5 \end{bmatrix} \quad (8)$$

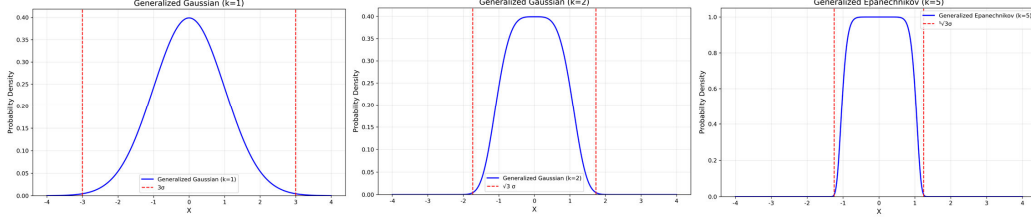


Figure 3: Generalized kernel Function.

Table 1: **Timings.** We compare the training and inference time of our method with that of Scaffold-GS [6], evaluating both with and without our proposed densification strategy, for the "Train" scene in Tanks&Temples dataset [9].

Train Metrics Method	Ours	Ours	Scaffold-GS [6]	Scaffold-GS
	wo ∇_{\max}	w ∇_{\max}	wo ∇_{\max}	w ∇_{\max}
PSNR \uparrow	22.96	23.06	22.45	22.57
#Anchor(k) \downarrow	393.8	584.4	353.3	520.0
Training Time \downarrow	26 minutes	32 minutes	24 minutes	29 minutes
Rendering FPS \uparrow	80	75	88	79

Therefore, for any given (x^s, y^s) , and (μ_x^s, μ_y^s) ,

$$\begin{bmatrix} x^s - \mu_x^s \\ y^s - \mu_y^s \end{bmatrix} = \begin{bmatrix} \frac{f_x}{t} * (x^v - \mu_x^v) \\ \frac{f_y}{t} * (y^v - \mu_y^v) \end{bmatrix} \quad (9)$$

The relationship between them is:

$$\begin{bmatrix} x^s - \mu_x^s \\ y^s - \mu_y^s \end{bmatrix} = P \begin{bmatrix} x^v - \mu_x^v \\ y^v - \mu_y^v \end{bmatrix} \quad (10)$$

Finally, the distance function value in screen space is:

$$\begin{aligned} \begin{bmatrix} x^s - \mu_x^s \\ y^s - \mu_y^s \end{bmatrix}^T \Sigma^{-s} \begin{bmatrix} x^s - \mu_x^s \\ y^s - \mu_y^s \end{bmatrix} &= \begin{bmatrix} x^v - \mu_x^v \\ y^v - \mu_y^v \end{bmatrix}^T P^T P^{-T} \Sigma^{-v} P^{-1} P \begin{bmatrix} x^v - \mu_x^v \\ y^v - \mu_y^v \end{bmatrix} \\ &= \begin{bmatrix} x^v - \mu_x^v \\ y^v - \mu_y^v \end{bmatrix}^T \Sigma^{-v} \begin{bmatrix} x^v - \mu_x^v \\ y^v - \mu_y^v \end{bmatrix} \end{aligned} \quad (11)$$

In summary, in our framework, we can define any functional kernel using the Mahalanobis distance. At this point, the function values in screen space and view space are equivalent. For 3DGS, an affine projection, approximated by the Jacobian matrix, is used to approximate the covariance. Since the projection from 3D to 2D is non-invertible, this identity does not hold.

B.6 More Details of SAP-Ges

In this paper, we extend the generalized Gaussian kernel:

$$\mathcal{K}^{Ges}(x; \mu, \Sigma) = e^{-\frac{1}{2} M^p(x)} \quad M(x) \in [0, \infty). \quad (12)$$

Its illustration is shown in the first row of Fig. 3. For 3DGS, its pixel radius is 3λ , where λ is the largest eigenvalue of the covariance matrix, which corresponds to the major axis. To ensure that $(3\lambda, 0)^T \Sigma^{-s} (3\lambda, 0) \leq 9$, 3DGS imposes a condition on the covariance matrix Σ^s . This condition guarantees that the squared Mahalanobis distance for the point $(3\lambda, 0)$ is less than or equal to 9, ensuring that the distance function does not exceed a certain threshold. To ensure that the distance value is less than 9, we take the radius of the generalized Gaussian kernel as $3^{1/p}\lambda$. The specific formula is:

$$\left(\begin{pmatrix} 3^{1/p}\lambda, 0 \end{pmatrix}^T \Sigma^{-s} \begin{pmatrix} 3^{1/p}\lambda, 0 \end{pmatrix} \right)^p \leq 9. \quad (13)$$

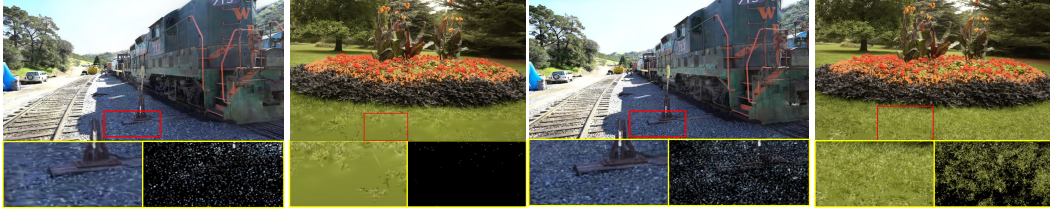


Figure 4: **Comparison of the effects of two splitting methods.** We have magnified the rendered image and the point distribution of the highlighted region. In the left two columns, the original average position gradient is employed, which results in numerous voids leading to blurring; in the right two columns, the combination of average and maximum position gradients effectively fills these voids, yielding improved rendering quality.

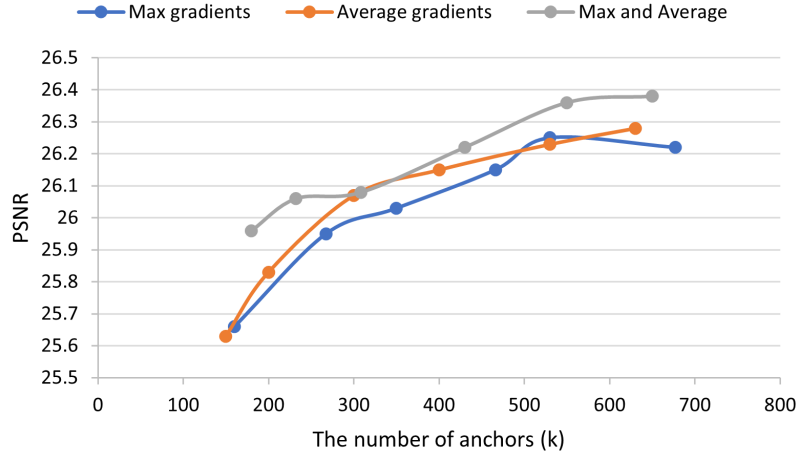


Figure 5: **Relationship between different densification strategies, PSNR, and point count.** Using different densification strategies, setting various thresholds can increase the number of points. However, when the point count reaches a certain threshold, the rendering quality converges.

91 When $p > 1$, the generalized Gaussian kernel is more compact than the Gaussian kernel. When
 92 $p < 1$, the radius of the generalized Gaussian kernel becomes larger.

93 B.7 Experiments on Computational Resources

94 Our method shares a rendering pipeline that is highly similar to 3DGS [8], so the training and
 95 rendering times are approximately equivalent given the same number of primitives. However, because
 96 of potential differences in the gradients produced by our method compared to the original one,
 97 different numbers of points may be generated under the corresponding densification strategies. While
 98 the number of points is approximately proportional to the computation time, our method demonstrates
 99 superior rendering quality under a comparable point count. We report the training and rendering
 100 overhead of our method and the baseline under two different densification strategies, as shown in
 101 Table 1.

102 B.8 Discussion of Densification Strategies

103 When using average position gradient densification, some regions are prone to gradient vanishing,
 104 resulting in holes or blurriness. However, as shown in Fig. 4, by incorporating the maximum position
 105 gradient allows more effective densification in regions with significant viewpoint differences.

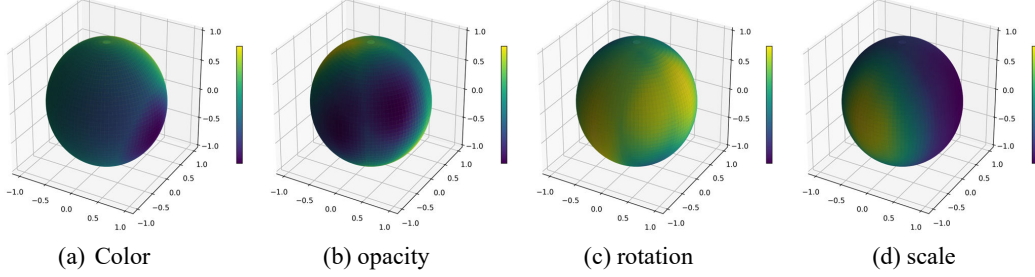


Figure 6: **View-dependent Gaussian attribute decoding.** We uniformly sample directions on the sphere to evaluate the performance of our attribute decoder.

Table 2: **Comparison of different kernels.** We compared different parameter designs of SAP-Ges.

SAP-Ges($p = 2$)	SAP-Ges($p = 0.5$)	SAP-Ges($p_{learnable}$)	SAP-Gaussian($p = 1$)
PSNR↑ / Mem↓ / FPS↑	PSNR↑ / Mem↓ / FPS↑	PSNR↑ / Mem↓ / FPS↑	PSNR↑ / Mem↓ / FPS↑
24.65 / 124.61 / 90	24.43 / 86.83 / 70	24.52 / 99.89 / 76	25.04 / 103.53 / 82

We tested the effects of three different densification methods. As shown in Fig. 5, the x-axis represents the number of points, and the y-axis represents PSNR. When combining the maximum and average position gradients, the model achieves the highest results. From the trend between the number of points and the rendering quality, it can be observed that when the number of points is low, the rendering quality is directly proportional to the number of points. However, once a certain threshold is reached, increasing the number of points further does not improve rendering quality. Instead, it may introduce more floating artifacts, leading to a worse rendering result.

B.9 View-dependent Gaussian Attribute Decoding

We visualized our Gaussian attribute decoder, as shown in Fig. 6. Specifically, we uniformly sampled directions on the sphere and used these as input to the decoder, obtaining the corresponding decoded attributes for each direction. We visualized properties such as color, opacity, rotation, and scale. As illustrated, our decoder maintains a degree of continuity across adjacent viewpoints while exhibiting anisotropic characteristics.

B.10 More Ablation on Kernels

We conducted experiments on SAP-Ges with different parameter settings, on the Tanks&Temples Dataset [9], Table 2:

- $p = 2$: This produces a more compact kernel than the Gaussian, resulting in more points and higher memory usage, but faster rendering.
- $p = 0.5$: This leads to a wider kernel range with fewer points and the lowest memory consumption. However, rendering is the slowest because of large Gaussian extents.
- Learnable p : The results lie between the above two. We attribute this to the challenge of finding an optimal learning rate for p .
- Gaussian kernel ($p = 1$): This setting achieves the best rendering performance with a good trade-off between memory and speed.

B.11 Additional Results

Here we list the error metrics used in our evaluation across all methods and scenes considered, as shown in Tab. 3 4.

Table 3: Quantitative evaluation of rendering efficiency per scene in Mip-NeRF360[10].

	PSNR↑									
	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	Avg.
Mip-NeRF360[10]	24.40	21.64	26.94	26.36	22.81	31.40	29.44	32.02	33.11	27.57
3D-GS [8]	25.18	21.48	27.24	26.62	22.45	31.49	28.98	31.35	32.10	27.43
2D-GS [3]	24.82	20.99	26.91	26.41	22.52	30.86	28.45	30.62	31.64	27.03
StopThePop [11]	25.20	21.50	27.16	26.69	22.43	30.83	28.59	31.13	31.93	27.28
Scaffold-GS [6]	24.81	21.42	27.17	26.27	23.08	31.93	29.34	31.30	32.70	27.55
DisC-GS [12]	-	-	-	-	-	-	-	-	-	28.01
3DGS-MCMC [7]	25.67	22.09	27.65	27.47	23.2	32.32	29.26	31.82	32.54	28.00
SAP(Ours)	25.26	21.48	27.48	26.76	23.02	32.88	29.94	31.90	33.73	28.05

	SSIM↑									
	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	Avg.
Mip-NeRF360	0.693	0.583	0.816	0.746	0.632	0.913	0.895	0.920	0.939	0.793
3D-GS	0.763	0.603	0.862	0.772	0.632	0.917	0.906	0.925	0.939	0.814
2D-GS	0.731	0.573	0.845	0.764	0.630	0.918	0.908	0.927	0.940	0.804
StopThePop	0.767	0.604	0.862	0.775	0.635	0.917	0.903	0.925	0.939	0.814
Scaffold-GS	0.725	0.587	0.842	0.784	0.644	0.925	0.914	0.928	0.946	0.810
DisC-GS	-	-	-	-	-	-	-	-	-	0.833
3DGS-MCMC	0.784	0.609	0.866	0.810	0.665	0.934	0.921	0.937	0.950	0.831
SAP(Ours)	0.758	0.631	0.875	0.780	0.707	0.945	0.915	0.939	0.962	0.835

	LPIPS↓									
	bicycle	flowers	garden	stump	treehill	room	counter	kitchen	bonsai	Avg.
Mip-NeRF360	0.289	0.345	0.164	0.254	0.338	0.211	0.203	0.126	0.177	0.234
3D-GS	0.213	0.338	0.109	0.216	0.327	0.221	0.202	0.127	0.206	0.217
2D-GS	0.271	0.378	0.138	0.263	0.369	0.214	0.197	0.125	0.194	0.239
StopThePop	0.206	0.335	0.107	0.210	0.319	0.216	0.200	0.126	0.202	0.213
Scaffold-GS	0.256	0.359	0.146	0.284	0.338	0.202	0.191	0.126	0.185	0.232
DisC-GS	-	-	-	-	-	-	-	-	-	0.189
3DGS-MCMC	0.202	0.246	0.112	0.194	0.300	0.181	0.174	0.114	0.176	0.188
SAP(Ours)	0.227	0.314	0.123	0.248	0.305	0.173	0.192	0.123	0.169	0.208

Table 4: Quantitative evaluation of rendering efficiency per scene in Tanks & Temples Dataset [9] and Deep Blending [13] .

	Truck	Train	Avg.	PSNR \uparrow		
				Dr Johnson	Playroom	Avg.
Mip-NeRF360	24.91	19.52	22.22	29.14	29.66	29.40
3D-GS	25.39	22.04	23.71	29.06	29.86	29.46
2D-GS	-	-	22.96	-	-	29.49
StopThePop	24.93	21.48	23.21	29.42	30.31	29.86
Scaffold-GS	25.89	22.48	24.19	29.73	30.83	30.28
DisC-GS	-	-	24.96	-	-	30.42
3DGS-MCMC	26.11	22.47	24.29	29.00	30.33	29.67
SAP(Ours)	26.31	23.76	25.04	29.62	30.41	30.02

	Truck	Train	Avg.	SSIM \uparrow		
				Dr Johnson	Playroom	Avg.
Mip-NeRF360	0.857	0.660	0.758	0.901	0.900	0.900
3D-GS	0.878	0.813	0.845	0.898	0.901	0.900
2D-GS	-	-	0.825	-	-	0.899
StopThePop	0.878	0.808	0.843	0.903	0.905	0.904
Scaffold-GS	0.885	0.822	0.854	0.910	0.907	0.909
DisC-GS	-	-	0.866	-	-	0.907
3DGS-MCMC	0.890	0.830	0.860	0.890	0.900	0.895
SAP(Ours)	0.905	0.835	0.870	0.908	0.911	0.910

	Truck	Train	Avg.	LPIPS \downarrow		
				Dr Johnson	Playroom	Avg.
Mip-NeRF360	0.159	0.354	0.257	0.237	0.252	0.245
3D-GS	0.148	0.208	0.189	0.247	0.246	0.247
2D-GS	-	-	0.217	-	-	0.259
StopThePop	0.142	0.204	0.173	0.234	0.235	0.234
Scaffold-GS	0.143	0.204	0.174	0.235	0.242	0.239
DisC-GS	-	-	0.120	-	-	0.199
3DGS-MCMC	0.140	0.240	0.190	0.330	0.310	0.320
SAP(Ours)	0.114	0.175	0.145	0.228	0.244	0.236

References

- [1] H. Li, J. Liu, M. Sznajder, and O. Camps, “3d-hgs: 3d half-gaussian splatting,” *arXiv preprint arXiv:2406.02720*, 2024.
- [2] R. Xu, W. Chen, J. Wang, Y. Liu, P. Wang, L. Gao, S. Xin, T. Komura, X. Li, and W. Wang, “Supergaussians: Enhancing gaussian splatting using primitives with spatially varying colors,” *arXiv preprint arXiv:2411.18966*, 2024.
- [3] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, “2d gaussian splatting for geometrically accurate radiance fields,” in *ACM SIGGRAPH 2024 conference papers*, 2024, pp. 1–11.
- [4] Z. Yu, T. Sattler, and A. Geiger, “Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes,” *ACM Transactions on Graphics*, 2024.
- [5] Z. Yang, X. Gao, Y.-T. Sun, Y. Huang, X. Lyu, W. Zhou, S. Jiao, X. Qi, and X. Jin, “Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 61 192–61 216, 2025.
- [6] T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai, “Scaffold-gs: Structured 3d gaussians for view-adaptive rendering,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 654–20 664.
- [7] S. Kheradmand, D. Rebain, G. Sharma, W. Sun, Y.-C. Tseng, H. Isack, A. Kar, A. Tagliasacchi, and K. M. Yi, “3d gaussian splatting as markov chain monte carlo,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 80 965–80 986, 2024.
- [8] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3d gaussian splatting for real-time radiance field rendering,” *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.
- [9] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, “Tanks and temples: Benchmarking large-scale scene reconstruction,” *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–13, 2017.
- [10] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, “Mip-nerf 360: Unbounded anti-aliased neural radiance fields,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5470–5479.
- [11] L. Radl, M. Steiner, M. Parger, A. Weinrauch, B. Kerbl, and M. Steinberger, “Stopthepop: Sorted gaussian splatting for view-consistent real-time rendering,” *ACM Transactions on Graphics (TOG)*, vol. 43, no. 4, pp. 1–17, 2024.
- [12] H. Qu, Z. Li, H. Rahmani, Y. Cai, and J. Liu, “Disc-gs: Discontinuity-aware gaussian splatting,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 112 284–112 309, 2024.
- [13] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, “Deep blending for free-viewpoint image-based rendering,” *ACM Transactions on Graphics (ToG)*, vol. 37, no. 6, pp. 1–15, 2018.