Pixel-GS: Density Control with Pixel-aware Gradient for 3D Gaussian Splatting

Zheng Zhang[1] Wenbo Hu[2†] Yixing Lao[1]
Tong He[3] Hengshuang Zhao[1†]

[1]The University of Hong Kong [2]Tencent AI Lab [3]Shanghai AI Lab

Abstract. 3D Gaussian Splatting (3DGS) has demonstrated impressive novel view synthesis results and advancing real-time rendering performance. However, the effectiveness of 3DGS heavily relies on the quality of the initial point cloud, as poor initialization can result in blurring and needle-like artifacts. This issue is mainly due to the point cloud growth condition, which only considers the average gradient magnitude of points from observable views, thereby failing to grow for large Gaussians that are observable from many viewpoints while many of them are only covered in the boundaries. To address this, we introduce Pixel-GS to take the area covered by the Gaussian in each view into account during the computation of the growth condition. The covered area is employed to adaptively weigh the gradients from different views, thereby facilitating the growth of large Gaussians. Consequently, Gaussians within the regions with insufficient initializing points can grow more effectively, leading to a more accurate and detailed reconstruction. Besides, we propose a simple yet effective strategy to suppress floaters near the camera by scaling the gradient field according to the distance to the camera. Extensive qualitative and quantitative experiments validate that our method achieves state-of-the-art rendering quality while maintaining real-time rendering, on challenging datasets such as Mip-NeRF 360 and Tanks & Temples. Code and demo are available at: `https://pixelgs.github.io`.

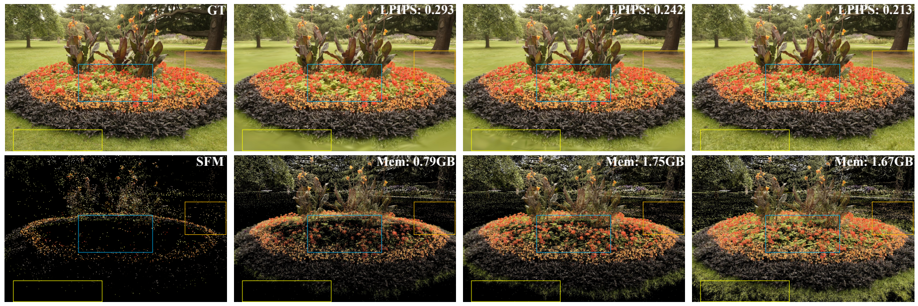
Keywords: View Synthesis · Real-time Rendering · 3D Gaussian Splatting · Adaptive Density Control

1 Introduction

Novel View Synthesis (NVS) is a fundamental problem in computer vision and computer graphics. Recently, 3D Gaussian Splatting (3DGS) [22] has drawn increasing attention for its explicit point-based representation of 3D scenes and real-time rendering performance.

3DGS represents the scene as a set of points associated with geometry (Gaussian scales) and appearance (opacities and colors) attributes. These attributes can be effectively learned by differentiable rendering, while the optimization of the point cloud's density is challenging. 3DGS carefully initializes the point cloud

†Corresponding author.



(a) Ground Truth (b) 3DGS* (original threshold) (c) 3DGS* (lower threshold) (d) Pixel-GS (Ours)

To go from (b) to (d), adjust densification from $\frac{\sum \|\mathbf{g}\|}{\sum 1} > \tau_{\text{pos}}$ to $\frac{\sum \text{pixel} \cdot \|\mathbf{g}\|}{\sum \text{pixel}} > \tau_{\text{pos}}$.

Fig. 1: Our Pixel-GS effectively grows points in areas with insufficient initializing points (a), leading to a more accurate and detailed reconstruction (d). In contrast, 3D Gaussian Splatting (3DGS) suffers from blurring and needle-like artifacts in these areas, even with a lower threshold of splitting and cloning to encourage more grown points (c). The rendering quality (in LPIPS ↓) and memory consumption are shown in the results. 3DGS* is our retrained 3DGS model with better performance.

using the sparse points produced by the Structure from Motion (SfM) process and uses an adaptive density control mechanism to split or clone the points during the optimization process. However, this mechanism relies heavily on the initial point cloud's quality and cannot effectively grow points in areas where the initial point cloud is sparse, resulting in blurry or needle-like artifacts in the synthesized images. In practice, the initial SfM point cloud could suffers from insufficient points in areas with repetitive textures and limited observations. As shown in the first and second columns of Figure 1, the blurry regions in the rendered images are aligned with the areas where only a limited number of points are initialized, as 3DGS fails to generate enough points in these areas.

In essence, this issue of insufficient point growth in certain areas is mainly attributed to the condition determining when to split or clone a point. 3DGS determines this splitting or cloning by checking whether the average gradient magnitude of the points in the Normalized Device Coordinates (NDC) exceeds a threshold. The magnitude of the gradient is equally averaged across different viewpoints, and the threshold is fixed. Large Gaussians are usually visible from many viewpoints, and the size of their projection area varies greatly across views. This variation leads to a significant difference in the number of pixels involved in the gradient calculation. In the mathematical model of the Gaussian distribution, pixels near the center of a projected Gaussian significantly influence the gradient more than those farther from the center. Larger Gaussians often have many viewpoints where the area near the projected center point is not within the screen space, thereby lowering the average gradient, making them difficult to split or clone. Merely lowering the threshold does not resolve this issue, as it tends to encourage unnecessary point growth in areas already well-populated with points, as illustrated in the third column of Figure 1, while still leaving blurry artifacts in the areas with insufficient points.

In this paper, we propose to consider the calculation of the mean gradient magnitude of points from the perspective of pixels. During the computation of the average gradient magnitude for a Gaussian, we take into account the number of pixels covered by the Gaussian in each view by replacing the averaging across views with the weighted average across views by the number of covered pixels. The motivation behind this is to amplify the gradient contribution of large Gaussians while leaving the conditions for splitting or cloning small Gaussians unchanged, such that we can effectively grow points in the areas with large Gaussians. Meanwhile, for small Gaussians, the weighted average has only a minimal impact the final gradient, as the variation of covered pixel numbers across different viewpoints is minimal. Therefore, the final number of points in areas with sufficient initial points remains largely unchanged to prevent unnecessary memory consumption and processing time. However, points in areas with insufficient initial points can be effectively grown to reconstruct fine-grained details. As shown in the last column of Figure 1, our method effectively grows points in areas with insufficient initial points and renders high-fidelity images, while directly lowering the threshold in 3DGS to maintain a similar number of final points fails to render blur-free results. Additionally, we observe that floaters tend to appear near the camera, which are points that are not well aligned with the scene geometry and are not contributing to the final rendering. To this end, we propose to scale the gradient field in NDC space according to the depth value of the points, thereby suppressing the growth of floaters near the camera.

To evaluate our method, we conduct extensive experiments on the challenging Mip-NeRF 360 [3] and Tanks & Temples [23] datasets. Experimental results validate that our method consistently outperforms the original 3DGS, both quantitatively (17.8% improvement in terms of LPIPS) and qualitatively. We also show that our method is more robust to the sparsity of the initial point cloud by manually discarding a certain proportion (up to 99%) of the initial SfM point clouds. In summary, we make the following contributions:

- We analyze the reason for the blurry artifacts in 3DGS and propose to optimize the number of points from the perspective of pixels, thereby enabling effective growth of points in areas with insufficient initial points.
- We present a simple yet effective gradient scaling strategy to suppress the floater artifacts near the camera.
- Our method achieves state-of-the-art performance on the challenging Mip-NeRF 360 and Tanks & Temples datasets and is more robust to the variation in the quality of initial points.

2 Related Work

Novel view synthesis. The task of novel view synthesis refers to the process of generating images from viewpoints different from the original input viewpoints. Recently, NeRF [36] has achieved impressive results in novel view synthesis by using neural networks to approximate the radiance field and employing volumetric rendering [10, 28, 33, 34] techniques for rendering. These approaches use implicit functions (such as MLPs [2, 3, 36], feature grid-based representations [6, 13, 30, 38, 48], or feature point-based representations [22, 53]) to fit the scene's radiance field and render the scene along each camera ray accumulatively. Due to the requirement to sample multiple points and determine the density and color of each point along a ray during volumetric rendering, this process is slow in rendering speeds. Subsequent methods [16,43,44,59,61] have refined a pre-trained NeRF into a sparse representation, thus achieving real-time rendering of NeRF. Although some improved scene representations [2–4, 6, 7, 13, 17, 26, 30, 38, 48] have been proposed to enhance one or more aspects of NeRF, such as training cost, rendering results, and rendering speed, 3D Gaussian Splatting (3DGS) [22] still draws increasing attention due to its explicit representation, high-fidelity results, and real-time rendering speed. Subsequent works on 3DGS have further improved it from perspectives such as anti-aliasing [54, 62], reducing memory usage [12, 27, 31, 37, 39, 40], replacing spherical harmonics functions to enhance the modeling capability of high-frequency signals based on reflective surfaces [57], and modeling dynamic scenes [11, 18, 21, 25, 32, 52, 56, 58].

Point-based radiance field. Point-based representations commonly represent scenes using fixed-size, unstructured points, and are rendered by rasterization using GPUs [5, 45, 47]. Although this is a simple and convenient representation to address topological changes, it often results in holes or outliers, leading to artifacts during rendering. To mitigate this issue of discontinuity, researchers have proposed differentiable rendering methods based on points, utilizing points to model local domains [1, 15, 19, 22, 24, 29, 51, 53, 60]. Among these approaches, Aliev et al. [1] and Kopanas et al. [24] employ neural networks to represent point features and utilize 2D CNNs for rendering [1, 24]. Point-NeRF [53] models 3D scenes using neural 3D points and presents strategies for pruning and growing points to repair common holes and outliers in point-based radiance fields. 3DGS [22] renders with rasterization, which is significantly faster than volumetric approaches. It starts with a sparse point cloud initialization from SfM, using three-dimensional Gaussian distributions to fit each point's influence area and spherical harmonics functions to determine their color features. To enhance the representational capability of this point-based spatial function, 3DGS introduces a density control mechanism based on the gradient of each point's NDC (Normalized Device Coordinates) coordinates and opacity, managing the growth and elimination of the point cloud. Recent work [8] on 3DGS has improved the point cloud growth process by incorporating depths and normals to enhance the fitting ability in low-texture areas. In contrast, our Pixel-GS does not require any additional priors or information resources, *e.g.* depths and normals, and can directly grow points in areas with insufficient initializing points, reducing blurring and needle-like artifacts.

Floater artifacts. Most radiance field scene representations encounter floater artifacts [14, 50]. Some works [9, 46] address floaters by introducing depth priors. NeRFshop [20] proposes an editing method to remove floaters. Mip-NeRF 360 [3] introduces a distortion loss to encourage unimodal density distribution along each ray, reducing floaters near the camera. NeRF in the Dark [35] uses a variance loss of weights to decrease floaters. FreeNeRF [55] introduces a penalty term for the density of points close to the camera as a loss to reduce floaters near the camera. "Floaters No More" [42] removes floaters by scaling the gradient field of the spatial domain. Inspired by this approach, our method scales the Gaussian gradient near the camera to remove floaters.

3 Method

We first review the point cloud growth condition of adaptive density control in 3DGS. Then, we propose a method for calculating the average gradient magnitude in the point cloud growth condition from a pixel perspective, significantly enhancing the reconstruction capability in areas with insufficient initial points. Finally, we show that by scaling the spatial gradient field that controls point growth, floaters near the input cameras can be effectively suppressed.

3.1 Preliminaries

In 3D Gaussian Splatting, Gaussian i under viewpoint k generates a 2D covariance matrix $\Sigma_{\mathrm{2D}}^{i,k} = \begin{pmatrix} a^{i,k} & b^{i,k} \\ b^{i,k} & c^{i,k} \end{pmatrix}$, and the corresponding influence range radius R_k^i can be determined by:

$$R_k^i = 3\left(\frac{a^{i,k}+c^{i,k}}{2} + \sqrt{\left(\frac{a^{i,k}+c^{i,k}}{2}\right)^2 - \left(a^{i,k}c^{i,k} - (b^{i,k})^2\right)}\right), \tag{1}$$

which covers 99% of the probability in the Gaussian distribution. For Gaussian i, under viewpoint k, the coordinates in the camera coordinate system are $\left(\mu_{c,x}^{i,k}, \mu_{c,y}^{i,k}, \mu_{c,z}^{i,k}\right)$, and in the pixel coordinate system, they are $\left(\mu_{p,x}^{i,k}, \mu_{p,y}^{i,k}, \mu_{p,z}^{i,k}\right)$. With the image width being W pixels and the height H pixels, Gaussian i participates in the calculation for viewpoint k when it simultaneously satisfies the following six conditions:

$$\begin{cases} R_k^i > 0, \mu_{c,z}^{i,k} > 0.2, \\ -R_k^i - 0.5 < \mu_{p,x}^{i,k} < R_k^i + W - 0.5, \\ -R_k^i - 0.5 < \mu_{p,y}^{i,k} < R_k^i + H - 0.5. \end{cases} \tag{2}$$

Whether a point is split or cloned is determined by the average magnitude of the gradient of the NDC coordinates for the viewpoints in which the Gaussian participates in the calculation. Specifically, for Gaussian i under viewpoint k, the NDC coordinate is $(\mu_{\mathrm{ndc,x}}^{i,k}, \mu_{\mathrm{ndc,y}}^{i,k}, \mu_{\mathrm{ndc,z}}^{i,k})$, and the loss under viewpoint k is L_k. During adaptive density control per 100 iterations, Gaussian i participates

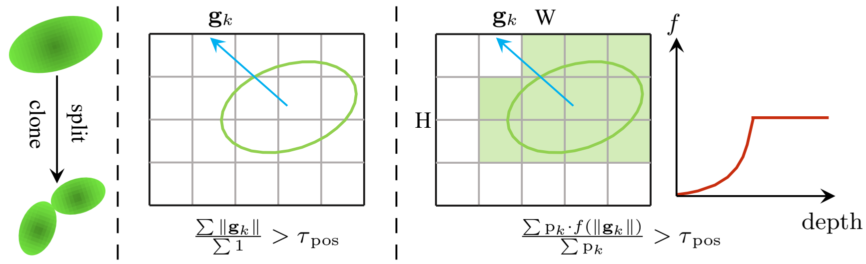


Fig. 2: Pipeline of Pixel-GS. p_k represents the number of pixels participating in the calculation for the Gaussian from this viewpoint, and $\mathbf{g}_k$ represents the gradient of the Gaussian's NDC coordinates. We change the condition for deciding whether a Gaussian should split or clone from the left to the right side.

in the calculation for M^i viewpoints. The threshold τ_{pos} is set to 0.0002 in 3D Gaussian Splatting. When Gaussian satisfies

$$\frac{1}{M^i}\sum_{k=1}^{M^i}\sqrt{\left(\frac{\partial L_k}{\partial \mu_{\text{ndc,x}}^{i,k}}\right)^2+\left(\frac{\partial L_k}{\partial \mu_{\text{ndc,y}}^{i,k}}\right)^2} > \tau_{\text{pos}}, \tag{3}$$

it is transformed into two Gaussians.

3.2 Pixel-aware Gradient

Although the current criteria used to decide whether a point should split or clone are sufficient for appropriately distributing Gaussians in most areas, artifacts tend to occur in regions where initial points are sparse. In 3DGS, the lengths of the three axes of the ellipsoid corresponding to Gaussian i are initialized using the values calculated by:

$$r^i = \sqrt{\frac{\left(d_1^i\right)^2+\left(d_2^i\right)^2+\left(d_3^i\right)^2}{3}}, \tag{4}$$

where d_1^i, d_2^i, and d_3^i are the distances to the three nearest points to Gaussian i, respectively. We observe that areas inadequately modeled often have very sparse initial SfM point clouds. This sparsity leads to the initialization of Gaussians in these areas with ellipsoids characterized by larger axis lengths, resulting in their computation involving too many viewpoints. These Gaussians exhibit larger gradients only in viewpoints where the center point, after projection, is within or near the pixel space. This implies that, from these viewpoints, the large Gaussians cover a larger area in the pixel space after projection. This results in these points having a smaller average gradient size of their NDC coordinates during the adaptive density control process every 100 iterations (Eq. 3), because they participate in the computation from too many viewpoints and only have significant gradient sizes in individual viewpoints. Consequently, it is difficult for these points to split or clone, leading to poor modeling in these areas.

Below, we analyze why the Gaussians in the previously mentioned sparser areas can only obtain larger NDC coordinate gradients from viewpoints with sufficient coverage, whereas for viewpoints that only affect the edge areas, the NDC coordinate gradients are smaller. The contribution of a pixel under viewpoint k to the NDC coordinate gradient of Gaussian i can be computed as:

$$\begin{pmatrix} \frac{\partial L_k}{\partial \mu^{i,k}_{\mathrm{ndc,x}}} \\ \frac{\partial L_k}{\partial \mu^{i,k}_{\mathrm{ndc,y}}} \end{pmatrix} = \sum_{pix=1}^{m^i_k} \sum_{j=1}^{3} \left(\frac{\partial L_k}{\partial c^{pix}_j} \frac{\partial c^{pix}_j}{\partial \alpha^i_{k,pix}} \begin{pmatrix} \frac{\partial \alpha^i_{k,pix}}{\partial \mu^{i,k}_{\mathrm{ndc,x}}} \\ \frac{\partial \alpha^i_{k,pix}}{\partial \mu^{i,k}_{\mathrm{ndc,y}}} \end{pmatrix} \right), \tag{5}$$

where both $\frac{\partial \alpha^i_{k,pix}}{\partial \mu^{i,k}_{\mathrm{ndc,x}}}$ and $\frac{\partial \alpha^i_{k,pix}}{\partial \mu^{i,k}_{\mathrm{ndc,y}}}$ contain factor α^i_k, which can be calculated as:

$$\alpha^i_{k,pix} = \sigma^i \exp\left(-\frac{1}{2} \begin{pmatrix} pix_x - \mu^{i,k}_{p,x} \\ pix_y - \mu^{i,k}_{p,y} \end{pmatrix}^T \left(\Sigma^{i,k}_{\mathrm{2D}} \right)^{-1} \begin{pmatrix} pix_x - \mu^{i,k}_{p,x} \\ pix_y - \mu^{i,k}_{p,y} \end{pmatrix} \right), \tag{6}$$

where c^{pix}_j represents the color of the jth channel of the current pixel, and m^i_k represents the number of pixels involved in the calculation for Gaussian i under viewpoint k. As a function of the distance between the center of the projected Gaussian and the pixel center, $\alpha^i_{k,pix}$ exhibits exponential decay as the distance increases. Therefore, pixels located close to the center of the projected Gaussian contribute significantly more to the NDC coordinate gradient of this Gaussian than those positioned further away. In many viewpoints, the edge areas of a large Gaussian participate in the calculations for these viewpoints, which leads to smaller average NDC coordinate gradients for the large Gaussian across different viewpoints. On the other hand, we observe that when a large number of pixels are involved in the calculation for a given viewpoint, the central region of the Gaussian often participates in the calculations for this viewpoint. It is straightforward to see that, when a large number of pixels are involved in the calculation after projection, the projected center point tends to be within the pixel plane, and according to previous calculations, a small number of pixels near the center point can be the main contributors to the gradient of the NDC coordinates.

To solve this problem, we assign a weight to the gradient size of the NDC coordinates for each Gaussian at every viewpoint. This weight is calculated by dividing the number of pixels involved in the computation of the Gaussian at the corresponding viewpoint by the resolution of that viewpoint. The advantage of this computational approach is that, for large Gaussians, the number of pixels involved in the calculations varies significantly across different viewpoints. According to previous derivations, these large Gaussians only receive larger gradients in viewpoints where a higher number of pixels are involved in the calculations. By averaging the magnitude of gradients weighted by the number of participating pixels, we can more effectively promote the splitting or cloning of these Gaussians. Additionally, for smaller Gaussians, since the variation in the number of pixels involved across different viewpoints is minimal, the proposed averaging method results in little change compared to the original conditions and does not lead to excessive additional memory consumption. The new equation to decide whether a Gaussian undergoes split or clone is given by:

$$\frac{\sum_{k=1}^{M^i} m_k^i \sqrt{\left(\frac{\partial L_k}{\partial \mu_{\mathrm{ndc,x}}^{i,k}}\right)^2 + \left(\frac{\partial L_k}{\partial \mu_{\mathrm{ndc,y}}^{i,k}}\right)^2}}{\sum_{k=1}^{M^i} m_k^i} > \tau_{\mathrm{pos}}, \tag{7}$$

where M^i is the number of viewpoints in which Gaussian i participates in the computation during the corresponding 100 iterations of adaptive density control, m_k^i is the number of pixels Gaussian i participates in at viewpoint k divided by the total number of pixels at viewpoint k, and $\frac{\partial L_k}{\partial \mu_{\mathrm{ndc,x}}^{i,k}}$ and $\frac{\partial L_k}{\partial \mu_{\mathrm{ndc,y}}^{i,k}}$ respectively represent the gradients of Gaussian i in the x and y directions of NDC space at viewpoint k. The conditions under which a Gaussian participates in the computation for a pixel is given by:

$$\begin{cases} \sqrt{\left(pix_x - \mu_{p,x}^{i,k}\right)^2 + \left(pix_y - \mu_{p,y}^{i,k}\right)^2} < R_k^i, \\ \prod_{j=1}^{i} \left(1 - \alpha_{k,pix}^{j}\right) \geqslant 10^{-4}, \\ \alpha_{k,pix}^{i} \geqslant \frac{1}{255}, \end{cases} \tag{8}$$

while the conditions under which a Gaussian participates in the computation from a viewpoint is given by Eq. 2.

3.3 Scaled Gradient Field

While using "Pixel-aware Gradient" to decide whether a point should split or clone (Eq. 7) can address artifacts in modeling areas with insufficient viewpoints and repetitive texture, we find that this condition for point cloud growth also exacerbates the presence of floaters near the camera. This is mainly because floaters near the camera occupy a large screen space and have significant gradients in their NDC coordinates, leading to an increasing number of floaters during the point cloud growth process. To address this issue, we scale the gradient field of the NDC coordinates. Specifically, we use the radius to determine the scale of the scene, where the radius is calculated by:

$$\mathrm{radius} = 1.1 \cdot \max_i \left\{ \left\| \mathbf{C}_i - \frac{1}{N} \sum_{j=1}^{N} \mathbf{C}_j \right\|_2 \right\}. \tag{9}$$

In the training set, there are N viewpoints, with $\mathbf{C}_j$ representing the coordinates of the jth viewpoint's camera in the world coordinate system. We scale the gradient of the NDC coordinates for each Gaussian i under the kth viewpoint, with the scaling factor $f(i,k)$ being calculated by:

$$f(i,k) = \mathrm{clip}\left(\left(\frac{\mu_{c,z}^{i,k}}{\gamma_{\mathrm{depth}} \mathrm{radius}}\right)^2, 0, 1\right), \tag{10}$$

where $\mu_{c,z}^{i,k}$ is the z-coordinate of Gaussian i in the camera coordinate system under the kth viewpoint, indicating the depth of this Gaussian from the viewpoint, and γ_{depth} is a hyperparameter set manually.

The primary inspiration for using squared terms as scaling coefficients in Eq. 10 comes from "Floaters No More" [42]. This paper highlights that floaters in NeRF [36] mainly arise from regions close to the camera occupying more pixels after projection, thereby receiving excessive gradients during optimization. Consequently, these regions close to the camera are optimized first, obscuring the originally correct spatial positions from being optimized. The number of pixels occupied is inversely proportional to the square of the distance to the camera, and hence we scale the gradients by squared distance.

In summary, a major issue with pixel-based optimization is the imbalance in the spatial gradient field, leading to inconsistent optimization speeds across different areas. Adaptive scaling of the gradient field in different spatial regions can effectively address this problem. Therefore, the final calculation equation that determines whether a Gaussian undergoes a "split" or "clone" is given by:

$$\frac{\sum_{k=1}^{M^i} m_k^i f\left(i,k\right) \sqrt{\left(\frac{\partial L_k}{\partial \mu_{\text{ndc,x}}^{i,k}}\right)^2 + \left(\frac{\partial L_k}{\partial \mu_{\text{ndc,y}}^{i,k}}\right)^2}}{\sum_{k=1}^{M^i} m_k^i} > \tau_{\text{pos}}. \tag{11}$$

4 Experiments

4.1 Experimental Setup

Datasets and benchmarks. We evaluate our method across a total of 30 real-world scenes, including all scenes from Mip-NeRF 360 (9 scenes) [3] and Tanks & Temples (21 scenes) [23], which are among the most widely used datasets in the field of 3D reconstruction. They both contain bounded indoor scenes and unbounded outdoor scenes, allowing for a comprehensive evaluation of our method's performance.

Evaluation metrics. We assess the quality of reconstruction through PSNR↑, SSIM↑ [49], and LPIPS↓ [63]. Among them, PSNR reflects pixel-aware errors but does not quite correspond to human visual perception as it treats all errors as noise without distinguishing between structural and non-structural distortions. SSIM accounts for structural transformations in luminance, contrast, and structure, thus more closely mirroring human perception of image quality. LPIPS uses a pre-trained deep neural network to extract features and measures the high-level semantic differences between images, offering a similarity that is closer to human perceptual assessment compared to PSNR and SSIM.

Implementation details. Our method only requires minor modifications to the original code of 3DGS, so it is compatible with almost all subsequent works on 3DGS. We use the default parameters of 3DGS to ensure consistency with the original implementation, including maintaining the same threshold τ_{pos} for

Table 1: Quantitative results on the Mip-NeRF 360 dataset. Cells are highlighted as follows: best , second best , and third best . We also show the results of three challenging scenes. 3DGS* is our retrained 3DGS model with better performance.

Method	Mip-NeRF 360 (all scenes)			Flowers			Bicycle			Stump		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Plenoxels [13]	23.08	0.625	0.463	20.10	0.431	0.521	21.91	0.496	0.506	20.66	0.523	0.503
INGP-Base [38]	25.30	0.671	0.371	20.35	0.450	0.481	22.19	0.491	0.487	23.63	0.574	0.450
INGP-Big [38]	25.59	0.699	0.331	20.65	0.486	0.441	22.17	0.512	0.446	23.47	0.594	0.421
Mip-NeRF 360 [3]	27.69	0.792	0.237	21.73	0.583	0.344	24.37	0.685	0.301	26.40	0.744	0.261
3DGS [22]	27.21	0.815	0.214	21.52	0.605	0.336	25.25	0.771	0.205	26.55	0.775	0.210
3DGS* [22]	27.71	0.826	0.202	21.89	0.622	0.328	25.63	0.778	0.204	26.90	0.785	0.207
Pixel-GS (Ours)	27.88	0.834	0.176	21.94	0.652	0.251	25.74	0.793	0.173	27.11	0.796	0.181

Table 2: Quantitative results on the Tanks & Temples dataset. We also show the results of three challenging scenes. * indicates retraining for better performance.

Method	Tanks & Temples (all scenes)			Train			Barn			Caterpillar		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3DGS* [22]	24.19	0.844	0.194	22.02	0.812	0.209	28.46	0.869	0.182	23.79	0.809	0.211
Pixel-GS (Ours)	24.38	0.850	0.178	22.13	0.823	0.180	29.00	0.888	0.144	24.08	0.832	0.173

splitting and cloning points as in the original 3DGS. For all scenes, we set a constant γ_{depth} value in Eq. 10 as 0.37 which is obtained through experimentations. All experiments are conducted on one RTX 3090 GPU with 24GB memory.

4.2 Main Results

We select several representative methods for comparison, including NeRF-based methods including Plenoxels [13], INGP [38], and Mip-NeRF 360 [3], and the 3DGS [22]. We use the official implementation for all of the compared methods, and the same train-test split as Mip-NeRF 360, selecting one out of every eight photos for testing.

Quantitative results. The quantitative results (PSNR, SSIM, and LPIPS) on the Mip-NeRF 360 and Tanks & Temples datasets are presented in Tables 1 and 2, respectively. We also provide the results of three challenging scenes for each dataset for more detailed information. Here, we retrain the 3DGS (noted as 3DGS*) as doing so yields a better performance than the original 3DGS (noted as 3DGS). We can see that our method consistently outperforms all the other methods, especially in terms of the LPIPS metric, while maintaining real-time rendering speed. Besides, compared to 3DGS, our method shows significant improvements in the three challenging scenes in both datasets and achieves better performance over the entire dataset. It quantitatively validates the effectiveness of our method in improving the quality of reconstruction.

Qualitative results. In Figures 1 and 3, we showcase the comparisons between our method and 3DGS*. We can see our approach significantly reduces the blurring and needle-like artifacts, *e.g.* the region of the flowers in the second row and the blow-up region in the last row, compared against the 3DGS*. These regions are initialized with insufficient points from SfM, and our method effectively

Table 3: Ablation study. The metrics are derived from the average values across all scenes of the Mip-NeRF 360 and Tanks & Temples datasets, respectively.

	Mip-NeRF 360			Tanks & Temples		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3DGS* [22]	27.71	0.826	0.202	24.23	0.844	0.194
Pixel-aware Gradient	27.74	0.833	0.176	21.80	0.791	0.239
Scaled Gradient Field	27.72	0.825	0.202	24.34	0.843	0.198
Complete Model	27.88	0.834	0.176	24.38	0.850	0.178

Table 4: Impact of lowering τ_{pos}. We show the corresponding quality and efficiency metrics when lowering the threshold τ_{pos} of point growth for 3DGS* and our method.

Dataset	Strategy	PSNR↑	SSIM↑	LPIPS↓	Train	FPS	Memory
Mip-NeRF 360	3DGS* ($\tau_{\text{pos}} = 2 \times 10^{-4}$)	27.71	0.826	0.202	25m40s	126	0.72GB
	3DGS* ($\tau_{\text{pos}} = 1.28 \times 10^{-4}$)	27.83	0.833	0.181	43m23s	90	1.4GB
	Ours ($\tau_{\text{pos}} = 2 \times 10^{-4}$)	27.88	0.834	0.176	41m25s	89	1.2GB
Tanks & Temples	3DGS* ($\tau_{\text{pos}} = 2 \times 10^{-4}$)	24.19	0.844	0.194	16m3s	135	0.41GB
	3DGS* ($\tau_{\text{pos}} = 1 \times 10^{-4}$)	23.86	0.842	0.187	27m59s	87	0.94GB
	Ours ($\tau_{\text{pos}} = 2 \times 10^{-4}$)	24.38	0.850	0.178	26m36s	92	0.84GB

grows points in these areas, leading to a more accurate and detailed reconstruction. Please refer to the supplemental materials for the point cloud comparison. These examples clearly validate that our method is more robust to the quality of initialization point clouds and can reconstruct high-fidelity details.

4.3 Ablation Studies

To evaluate the effectiveness of individual components of our method, *i.e.* the pixel-aware gradient and the scaled gradient field, we conduct ablation studies on the Mip-NeRF 360 and Tanks & Temples datasets. The quantitative and qualitative results are presented in Table 3 and Figure 4, respectively. We can see that both the pixel-aware gradient and the scaled gradient field contribute to the improvement of the reconstruction quality in the Mip-NeRF 360 dataset. However, the pixel-aware gradient strategy reduces the reconstruction quality in the Tanks & Temples dataset. This is mainly due to floaters that tend to appear near the camera in some large scenes in Tanks & Temples and the pixel-aware gradient encourages more Gaussians, as shown in column (b) of Figure 4. Notably, this phenomenon also exists for the 3DGS when the threshold τ_{pos} is lowered, which also promotes more Gaussians, as shown in Table 4. But importantly, the combination of both proposed strategies achieves the best performance in the Tanks & Temples dataset, as shown in Table 3, since the scaled gradient field can suppress the growth of floaters near the camera. In summary, the ablation studies demonstrate the effectiveness of our proposed individual components and the necessity of combining them to achieve the best performance.

(a) Ground Truth (b) Pixel-GS (Ours) (c) 3DGS* [22]

Fig. 3: Qualitative comparison between Pixel-GS (Ours) and 3DGS*. The first three scenes are from the Mip-NeRF 360 dataset (*Bicycle, Flowers, and Treehill*), while the last four scenes are from the Tanks & Temples dataset (*Barn, Caterpillar, Playground, and Train*). The blow-up regions or arrows highlight the parts with distinct differences in quality. 3DGS* is our retrained 3DGS model with better performance.

(a) 3DGS* (b) Pixel-aware Gradient (c) Scaled Gradient Field (d) Complete Model

Fig. 4: Qualitative results of the ablation study. The PSNR↑ results are shown on the corresponding images.

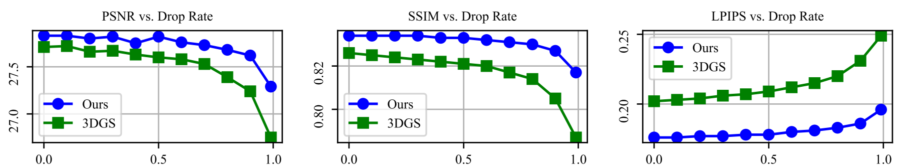


Fig. 5: Reconstruction quality (PSNR↑, SSIM↑, and LPIPS↓) *vs.* Dropping rate of initializing points. Here, the dropping rate refers to the percentage of points dropped from the original SfM point clouds for initializing Gaussians. The results are obtained on the Mip-NeRF 360 dataset.

4.4 Analysis

The impact of lowering the threshold τ_{pos}. As the blurring and needle-like artifacts in 3DGS mainly occur in areas with insufficient initializing points, one straightforward solution would be to lower the threshold τ_{pos} to encourage the growth of more points. To verify this, we experiment on the Mip-NeRF 360 and Tanks & Temples datasets by lowering the threshold τ_{pos} from $2e-4$ to $1.28e-4$ for 3DGS to make the final optimized number of points comparable to ours. From Table 4, we can see that lowering the threshold τ_{pos} for 3DGS significantly increases the memory consumption and decreases the rendering speed, while still falling behind ours in terms of reconstruction quality. As can be seen from the qualitative comparison in Figure 1, this is because the point cloud growth mechanism of 3DGS struggles to generate points in areas with insufficient initializing points and only yields unnecessary points in areas where the initial SfM point cloud is already dense. In contrast, although our method also results in additional memory consumption, our method's point cloud distribution is more uniform, enabling effectively growing points in areas with insufficient initializing points, thereby leading to a more accurate and detailed reconstruction while still maintaining real-time rendering speed.

Robustness to the quality of initialization point clouds. Finally, SfM algorithms often fail to produce high-quality point clouds in some areas, *e.g.*, too few observations, repetitive textures, or low textures. The point cloud produced by SfM is usually the necessary input for 3DGS and our method. Therefore, we explore the robustness of our method to the quality of initialization point clouds by randomly dropping points from the SfM point clouds used for initialization and compared the results with that of 3DGS. Figure 5 shows how the reconstruction quality varies with the proportion of dropped points. We can see that our method consistently outperforms 3DGS in terms of all the metrics (PSNR, SSIM, and LPIPS). And more importantly, our method is less affected by the dropping rate than 3DGS. Notably, even though the 99% initializing points have been dropped, the reconstruction quality of our method still surpasses that of 3DGS initialized with complete SfM point clouds, in terms of LPIPS. These results demonstrate the robustness of our method to the quality of initialization point clouds, which is crucial for real-world applications.

5 Discussion and Conclusion

Limitation While our method can enhance the quality of scene modeling, it does increase the number of Gaussians required to model a scene, which in turn raises memory consumption. This issue can be addressed by employing point cloud pruning techniques. For example, when our method is combined with the RadSplat's [41] point cloud pruning method, the average memory consumption on the Mip-NeRF 360 dataset can be reduced from 1.2GB to 0.4GB without compromising the quality. Additionally, when the camera's distance from the scene center changes significantly, it may be necessary to adjust the hyperparameter γ_{depth} to better eliminate floaters.

Conclusion The blurring and needle-like artifacts in 3DGS are mainly attributed to its inability to grow points in areas with insufficient initializing points. To address this issue, we propose Pixel-GS, which considers the number of pixels covered by a Gaussian in each view to dynamically weigh the gradient of each view during the computation of the growth condition. This strategy effectively grows Gaussians with large extents, which are more likely to exist in areas with insufficient initializing points, such that our method can adaptively grow points in these areas while avoiding unnecessary growth in areas with enough points. We also introduce a simple yet effective strategy to deal with floaters by scaling the gradient field by the distance to the camera. Extensive experiments demonstrate that our method significantly reduces blurring and needle-like artifacts and effectively suppresses floaters, achieving state-of-the-art performance in terms of rendering quality. Meanwhile, although our method consumes slightly more memory consumption, the increased points are mainly distributed in areas with insufficient initializing points, which are necessary for high-quality reconstruction, and our method still maintains real-time rendering speed. Finally, our method is more robust to the number of initialization points, thanks to our effective pixel-aware gradient and scaled gradient field.

Acknowledgement. This work is supported by the National Natural Science Foundation of China (No. 62201484), HKU Startup Fund, and HKU Seed Fund for Basic Research.

References

1. Aliev, K.A., Sevastopolsky, A., Kolos, M., Ulyanov, D., Lempitsky, V.: Neural point-based graphics. In: ECCV (2020) 4
2. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: ICCV (2021) 4
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: CVPR (2022) 3, 4, 9, 10
4. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. In: ICCV (2023) 4
5. Botsch, M., Hornung, A., Zwicker, M., Kobbelt, L.: High-quality surface splatting on today's gpus. In: EUROGRAPHICS (2005) 4
6. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: ECCV (2022) 4
7. Chen, Z., Li, Z., Song, L., Chen, L., Yu, J., Yuan, J., Xu, Y.: Neurbf: A neural fields representation with adaptive radial basis functions. In: ICCV (2023) 4
8. Cheng, K., Long, X., Yang, K., Yao, Y., Yin, W., Ma, Y., Wang, W., Chen, X.: Gaussianpro: 3d gaussian splatting with progressive propagation. In: ICML (2024) 4
9. Chung, J., Oh, J., Lee, K.M.: Depth-regularized optimization for 3d gaussian splatting in few-shot images. In: CVPR (2024) 4
10. Drebin, R.A., Carpenter, L., Hanrahan, P.: Volume rendering. In: SIGGRAPH (1988) 4
11. Duan, Y., Wei, F., Dai, Q., He, Y., Chen, W., Chen, B.: 4d gaussian splatting: Towards efficient novel view synthesis for dynamic scenes. arXiv:2402.03307 (2024) 4
12. Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D., Wang, Z.: Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. arXiv:2311.17245 (2023) 4
13. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR (2022) 4, 10
14. Goli, L., Reading, C., Sellán, S., Jacobson, A., Tagliasacchi, A.: Bayes' rays: Uncertainty quantification for neural radiance fields. In: CVPR (2024) 4
15. Gross, M., Pfister, H.: Point-based graphics (2011) 4
16. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. in 2021 ieee. In: ICCV (2021) 4
17. Hu, W., Wang, Y., Ma, L., Yang, B., Gao, L., Liu, X., Ma, Y.: Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In: ICCV (2023) 4
18. Huang, Y.H., Sun, Y.T., Yang, Z., Lyu, X., Cao, Y.P., Qi, X.: Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. In: CVPR (2024) 4
19. Insafutdinov, E., Dosovitskiy, A.: Unsupervised learning of shape and pose with differentiable point clouds. In: NeurIPS (2018) 4
20. Jambon, C., Kerbl, B., Kopanas, G., Diolatzis, S., Drettakis, G., Leimkühler, T.: Nerfshop: Interactive editing of neural radiance fields. PACMCGIT (2023) 4
21. Katsumata, K., Vo, D.M., Nakayama, H.: An efficient 3d gaussian representation for monocular/multi-view dynamic scenes. arXiv:2311.12897 (2023) 4
22. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. TOG (2023) 1, 4, 10, 11, 12

23. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. TOG (2017) 3, 9
24. Kopanas, G., Philip, J., Leimkühler, T., Drettakis, G.: Point-based neural rendering with per-view optimization. In: Computer Graphics Forum (2021) 4
25. Kratimenos, A., Lei, J., Daniilidis, K.: Dynmf: Neural motion factorization for real-time dynamic view synthesis with 3d gaussian splatting. arXiv:2312.00112 (2023) 4
26. Kulhanek, J., Sattler, T.: Tetra-nerf: Representing neural radiance fields using tetrahedra. In: ICCV (2023) 4
27. Lee, J.C., Rho, D., Sun, X., Ko, J.H., Park, E.: Compact 3d gaussian representation for radiance field. In: CVPR (2024) 4
28. Levoy, M.: Efficient ray tracing of volume data. TOG (1990) 4
29. Lin, C.H., Kong, C., Lucey, S.: Learning efficient point cloud generation for dense 3d object reconstruction. In: AAAI (2018) 4
30. Liu, L., Gu, J., Zaw Lin, K., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. In: NeurIPS (2020) 4
31. Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., Dai, B.: Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In: CVPR (2024) 4
32. Luiten, J., Kopanas, G., Leibe, B., Ramanan, D.: Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. arXiv:2308.09713 (2023) 4
33. Max, N.: Optical models for direct volume rendering. TVCG (1995) 4
34. Max, N., Chen, M.: Local and global illumination in the volume rendering integral. Tech. rep. (2005) 4
35. Mildenhall, B., Hedman, P., Martin-Brualla, R., Srinivasan, P.P., Barron, J.T.: Nerf in the dark: High dynamic range view synthesis from noisy raw images. In: CVPR (2022) 5
36. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV (2020) 3, 4, 9
37. Morgenstern, W., Barthel, F., Hilsmann, A., Eisert, P.: Compact 3d scene representation via self-organizing gaussian grids. arXiv:2312.13299 (2023) 4
38. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. TOG (2022) 4, 10
39. Navaneet, K., Meibodi, K.P., Koohpayegani, S.A., Pirsiavash, H.: Compact3d: Compressing gaussian splat radiance field models with vector quantization. arXiv:2311.18159 (2023) 4
40. Niedermayr, S., Stumpfegger, J., Westermann, R.: Compressed 3d gaussian splatting for accelerated novel view synthesis. In: CVPR (2024) 4
41. Niemeyer, M., Manhardt, F., Rakotosaona, M.J., Oechsle, M., Duckworth, D., Gosula, R., Tateno, K., Bates, J., Kaeser, D., Tombari, F.: Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. arXiv preprint arXiv:2403.13806 (2024) 14
42. Philip, J., Deschaintre, V.: Floaters no more: Radiance field gradient scaling for improved near-camera training. In: EGSR (2023) 5, 9
43. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In: ICCV (2021) 4
44. Reiser, C., Szeliski, R., Verbin, D., Srinivasan, P., Mildenhall, B., Geiger, A., Barron, J., Hedman, P.: Merf: Memory-efficient radiance fields for real-time view synthesis in unbounded scenes. TOG (2023) 4

45. Ren, L., Pfister, H., Zwicker, M.: Object space ewa surface splatting: A hardware accelerated approach to high quality point rendering. In: Computer Graphics Forum (2002) 4
46. Roessle, B., Barron, J.T., Mildenhall, B., Srinivasan, P.P., Nießner, M.: Dense depth priors for neural radiance fields from sparse input views. In: CVPR (2022) 4
47. Sainz, M., Pajarola, R.: Point-based rendering techniques. Computers & Graphics (2004) 4
48. Sun, C., Sun, M., Chen, H.T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In: CVPR (2022) 4
49. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. TIP (2004) 9
50. Warburg, F., Weber, E., Tancik, M., Holynski, A., Kanazawa, A.: Nerfbusters: Removing ghostly artifacts from casually captured nerfs. In: ICCV (2023) 4
51. Wiles, O., Gkioxari, G., Szeliski, R., Johnson, J.: Synsin: End-to-end view synthesis from a single image. In: CVPR (2020) 4
52. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Wang, X.: 4d gaussian splatting for real-time dynamic scene rendering. In: CVPR (2024) 4
53. Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., Neumann, U.: Point-nerf: Point-based neural radiance fields. In: CVPR (2022) 4
54. Yan, Z., Low, W.F., Chen, Y., Lee, G.H.: Multi-scale 3d gaussian splatting for anti-aliased rendering. In: CVPR (2024) 4
55. Yang, J., Pavone, M., Wang, Y.: Freenerf: Improving few-shot neural rendering with free frequency regularization. In: CVPR (2023) 5
56. Yang, Z., Yang, H., Pan, Z., Zhang, L.: Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In: ICLR (2024) 4
57. Yang, Z., Gao, X., Sun, Y., Huang, Y., Lyu, X., Zhou, W., Jiao, S., Qi, X., Jin, X.: Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting. arXiv:2402.15870 (2024) 4
58. Yang, Z., Gao, X., Zhou, W., Jiao, S., Zhang, Y., Jin, X.: Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In: CVPR (2024) 4
59. Yariv, L., Hedman, P., Reiser, C., Verbin, D., Srinivasan, P.P., Szeliski, R., Barron, J.T., Mildenhall, B.: Bakedsdf: Meshing neural sdfs for real-time view synthesis. In: SIGGRAPH (2023) 4
60. Yifan, W., Serena, F., Wu, S., Öztireli, C., Sorkine-Hornung, O.: Differentiable surface splatting for point-based geometry processing. TOG (2019) 4
61. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. In: ICCV (2021) 4
62. Yu, Z., Chen, A., Huang, B., Sattler, T., Geiger, A.: Mip-splatting: Alias-free 3d gaussian splatting. In: CVPR (2024) 4
63. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: CVPR (2018) 9