PATexGS: Perceptual-Adaptive Texture Scheduling for Visual Coherence in Textured Gaussian Splatting

Yuesong Wang[1], **Dounian Ma**[1], **Xiaoyu Chen**[2], **Tao Guan**[1]*

[1]School of Computer Science and Technology, Huazhong University of Science and Technology, China
[2]Department of Computer Science and Engineering, University of California, USA
yuesongwang@hust.edu.cn, dounianma@gmail.com, xic039@ucsd.edu, qd_gt@hust.edu.cn

Abstract

3D Gaussian Splatting (3DGS) has emerged as a mainstream solution for real-time and high-fidelity novel view synthesis. Building on this foundation, methods based on Textured Gaussians further improve the expression ability by incorporating explicit texture mapping into Gaussians. However, their reliance on fixed texture resolution often results in noticeable visual incoherence, triggering artifacts such as aliasing or inconsistent sharpness under different viewpoints. To address these issues, we propose **PATexGS**, a perceptual-adaptive texture scheduling framework designed to improve visual coherence for Textured Gaussians. Specifically, we introduce an entropy-guided texture allocation strategy that dynamically adjusts texture resolution based on each Gaussian's spatial gradient and rendering contribution, constantly preserving details while being memory efficient. Furthermore, we incorporate a mipmap-inspired hierarchical scheduling mechanism that adaptively schedules texture levels according to view-dependent projection scale, effectively suppressing aliasing and further enhancing perceptual consistency. Extensive experiments demonstrate that PATexGS significantly improves visual coherence while maintaining high rendering quality, outperforming existing TexturedGS variants in both perceptual fidelity and storage efficiency.

Introduction

Neural rendering methods have revolutionized 3D reconstruction and novel view synthesis, enabling high-fidelity real-time rendering. Among them, 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) has become a leading paradigm for its efficient training, explicit scene representation, and real-time performance. It has been widely applied in avatars (Kocabas et al. 2024; Shao et al. 2024), city modeling (Lin et al. 2024; Khan et al. 2024), scene editing (Chen et al. 2024b; Cen et al. 2025), and virtual reality (Jiang et al. 2024). However, 3DGS often produces inaccurate geometry and floaters in novel views. To improve surface alignment, 2D Gaussian Splatting (2DGS) (Huang et al. 2024a) aligns oriented 2D Gaussians with underlying surfaces, achieving better geometry but lower rendering quality. Yet, both 3DGS and 2DGS remain limited in their ability to represent high-frequency visual details.

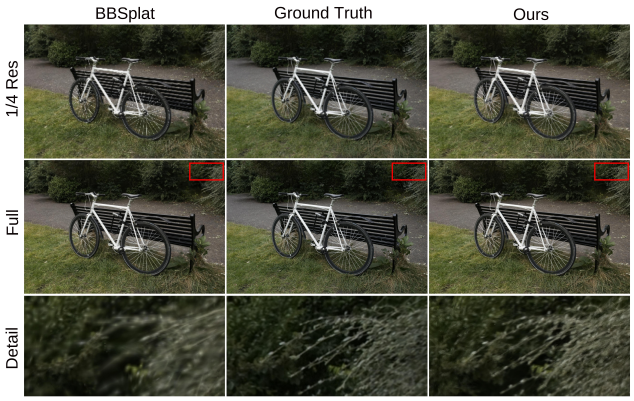

Figure 1: Teaser. Compared to the other textured Gaussian-based method (Svitov et al. 2024), our PATexGS achieves superior visual coherence and faithfully preserves fine details under both zoom-in and zoom-out views.

To enhance the expression ability, many works have explored adjusting the Gaussian kernel or using other types of primitives (Zhu et al. 2025; Hamdi et al. 2024; Wang et al. 2024). Recently, Textured Gaussians (Chao et al. 2025) and related works (Xu et al. 2024; Rong et al. 2025; Huang and Gong 2024; Svitov et al. 2024) draw inspiration from traditional rasterization by assigning a texture map to each Gaussian. With the RGB map for color and the alpha map for shape, it allows Gaussians to represent more complex structures and patterns using fewer primitives. Nonetheless, these approaches typically need to set a fixed texture resolution, which introduces two major problems(illustrated in Fig. 1): **(1)** it fails to adapt to varying scene entropy, causing detail loss in complex regions or memory waste in smooth areas; **(2)** it suffers from aliasing and inconsistent sharpness under view changes, similar to classic rendering artifacts like Moiré patterns or jagged edges.

To address these limitations, we propose **PATexGS**, a Perceptual-Adaptive Texture Scheduling framework for textured Gaussian splatting. Our method introduces two key contributions: First, we design an entropy-guided adaptive texture allocation strategy that adjusts the resolution of each Gaussian's texture map according to its gradient and ren-

*Corresponding author.

dering contribution, progressively increasing detail expression where needed while reducing memory overhead. Second, inspired by mipmapping techniques in graphics, we implement a view-aware hierarchical texture scheduling mechanism that selects appropriate mip levels during rendering based on projected scale, significantly reducing aliasing and improving perceptual consistency. Extensive experiments across diverse datasets (Mip-NeRF 360, Tanks and Temples, and Deep Blending) demonstrate that PATexGS achieves superior visual coherence and rendering quality compared to existing textured Gaussian methods, especially in low-resolution or zoom-out view settings. Our approach also remains efficient and robust under reduced Gaussian budgets, highlighting its potential for scalable high-fidelity rendering in real-world applications.

Our main contributions can be summarized as follows:

- **Per-Gaussian adaptive texture allocation.** We introduce a progressive texture growth mechanism that dynamically adjusts each Gaussian's texture resolution based on its rendering contribution and spatial gradient, enabling efficient detail preservation while avoiding redundant memory usage.
- **Mipmap-inspired hierarchical scheduling.** We propose a view-aware texture level selection strategy by constructing multi-scale mipmaps for each Gaussian, which significantly reduces aliasing and improves perceptual consistency under varying viewing conditions.
- **A robust textured Gaussian rendering framework.** We develop PATexGS as a novel textured Gaussian pipeline that supports high-quality, visually coherent rendering across real-world scenes.

Related Work

Novel View Synthesis

Novel view synthesis (NVS) aims to generate photorealistic novel views from a set of posed images. With the development of Structure-from-Motion (SfM) (Schonberger and Frahm 2016) and Multi-View Stereo (MVS) (Schönberger et al. 2016), it laid the foundation for many breakthrough methods (Zhou et al. 2018; Mildenhall et al. 2019). However, they often require storing numerous input images for reprojection and blending, causing high memory usage and failures in under-sampled regions. Recent advances in neural rendering have significantly improved the fidelity and efficiency of NVS (Mildenhall et al. 2021; Barron et al. 2023). Among them, 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) has emerged as a leading technique by combining explicit Gaussian primitives with real-time rendering capabilities. However, 3DGS often overfits to training views, resulting in geometry-inconsistent floaters in novel views. To eliminate such geometrically inaccurate floaters, many works (Guédon and Lepetit 2024; Huang et al. 2024a; Yu, Sattler, and Geiger 2024; Jiang et al. 2025; Lyu et al. 2024) explicitly impose geometric constraints within the 3DGS framework. 2DGS (Huang et al. 2024a) introduces surface-aligned elliptical Gaussians, enhancing mesh reconstruction accuracy through a ray-ellipse intersection formulation. GOF (Yu, Sattler, and Geiger 2024) proposes a volumetric representation that models surface opacity using compact Gaussian fields, achieving adaptive surface reconstruction even in unbounded or incomplete environments. PGSR (Chen et al. 2024a) incorporates a multi-view geometric consistency loss to ensure that Gaussians maintain geometric accuracy across views. Yet, while these methods improve geometry, they often tend to underperform 3DGS in visual quality. Furthermore, all Gaussian-based methods need numerous Gaussians to express fine visual details.

Textured Gaussians

Recent works (Svitov et al. 2024; Rong et al. 2025; Song et al. 2024; Xu et al. 2024; Chao et al. 2025; Huang and Gong 2024) propose to decouple geometry and appearance by enriching Gaussians with texture information. For instance, Texture-GS (Xu et al. 2024) leverages an MLP to map each Gaussian's local coordinates to a UV texture space, enabling texture-level editing in synthetic settings. Other approaches, such as GStex (Rong et al. 2025), assign each Gaussian a texture map to enhance its ability to capture appearance details. BBSplat (Svitov et al. 2024) and Textured Gaussians (Chao et al. 2025) further introduce RGBA texture maps into Gaussians, using RGB channels for color details and alpha channels for geometry details. These texture-based methods can increase the expression ability of Gaussians and avoid overfitting training views with wrong geometry. While significantly boosting rendering fidelity, they typically need to assign a fixed resolution texture to every Gaussian in advance, regardless of the local scene complexity corresponding to each Gaussian. This uniform allocation not only introduces redundancy in low-entropy regions but also limits the detail capacity in high-entropy areas. Moreover, although textured Gaussians naturally alleviate needle-like artifacts under zoomed-in views due to bilinear interpolation of texture, they still suffer from aliasing and Moiré patterns when zoomed out, similar to traditional rasterization pipelines. As a result, these textured Gaussian-based methods often suffer from visual incoherence.

Anti-Aliasing

Texture aliasing has long been a critical issue in classical rendering pipelines. To mitigate this, mipmap-based strategies were introduced, most notably by Williams (Williams 1983). Mipmapping precomputes a pyramid of progressively downsampled textures, allowing the renderer to select an appropriate resolution level based on screen-space derivatives of texture coordinates, thereby suppressing aliasing and Moiré effects. In neural rendering, similar ideas have been explored (Barron et al. 2021, 2022; Hu et al. 2023). For 3DGS, a fixed, view-independent margin is artificially added around each Gaussian to prevent the loss of gradient when its projected size on the image falls below one pixel. However, this hard margin exacerbates aliasing and leads to inferior anti-aliasing compared to NeRF-based methods. Several works have been proposed to deal with the above issue. Mip-Splatting (Yu et al. 2024) introduces a 3D Gaussian size constraint based on maximal view sampling frequency and replaces traditional screen-space dilation with a

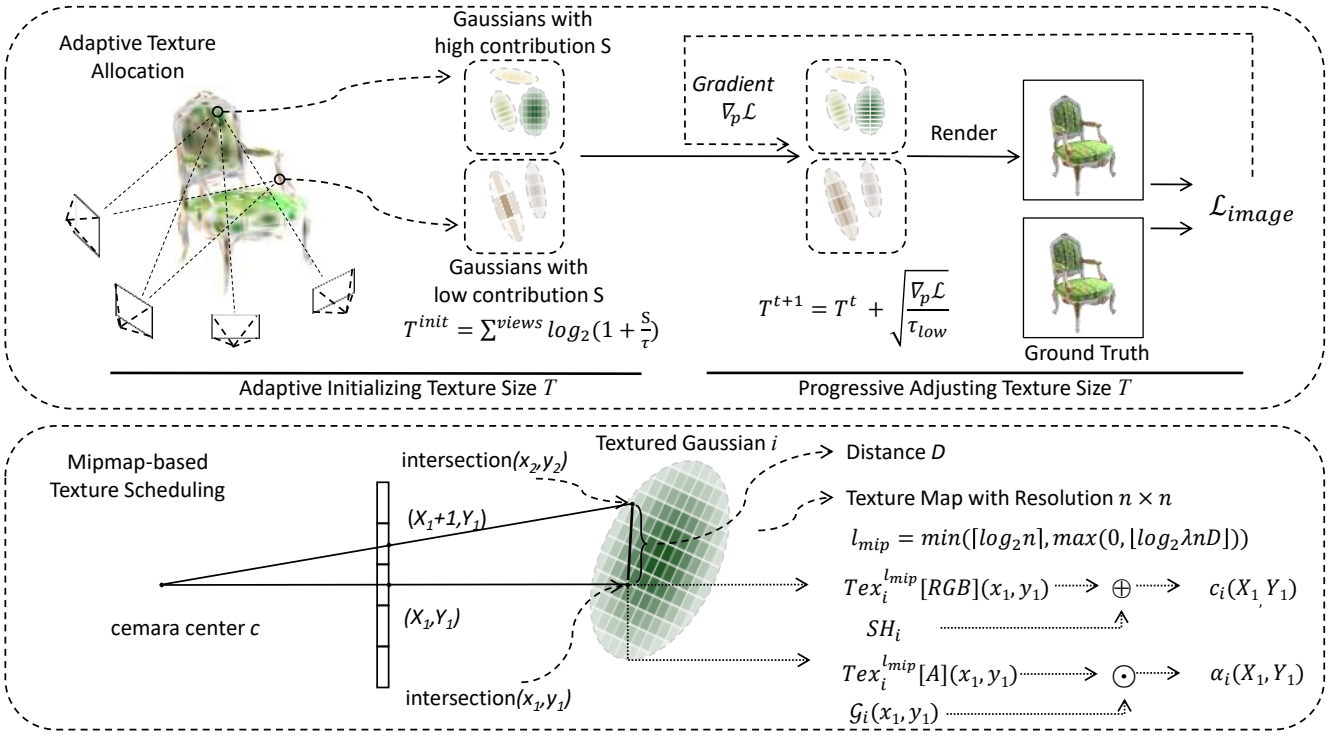

Figure 2: Pipeline. We first initialize texture resolution based on each Gaussian's scene contribution, then progressively adjust it using gradient-based allocation. During rendering, mipmaps are built per Gaussian, and the appropriate level is selected based on the projected pixel footprint. Final colors are obtained via alpha blending.

2D mip filter. MS-GS (Yan et al. 2024) maintains a hierarchy of Gaussians at multiple scales, enabling selective rendering: small Gaussians for high-resolution views and larger ones for zoomed-out views. Analytic-Splatting (Liang et al. 2024) refines pixel-level intensity computation by analytically integrating the Gaussian window over each pixel footprint rather than sampling at the center. Spectral-GS (Huang et al. 2024b) leverages spectral analysis of Gaussian covariance to introduce shape-aware splitting and view-consistent filtering. However, these techniques are for vanilla Gaussian primitives and do not apply to textured Gaussians.

Method

In this section, we first provide a brief overview of the textured Gaussian, which serves as the foundation of our method. We then introduce the adaptive texture allocation strategy and the mipmap-based texture scheduling, forming our PATexGS, as illustrated in Fig. 2.

Preliminaries: Textured Gaussian Splatting

Textured Gaussian Splatting extends 2D Gaussian Splatting by assigning each Gaussian primitive a learnable texture map in addition to geometric parameters. Each 2D Gaussian i is defined by a set of attributes: position μ_i, scale s_i, rotation quaternion r_i, opacity o_i, and a spherical harmonics(SH) function SH_i for view-dependent color. In practice, a local tangent plane $P_i(x, y)$ is defined for Gaussian i using a transformation matrix H_i as follows:

$$P_i(x,y) = H_i \cdot (x,y,1,1)^\top = \begin{bmatrix} R_i S_i & \mu_i \\ 0 & 1 \end{bmatrix} \cdot (x,y,1,1)^\top, \tag{1}$$

where R_i is the rotation matrix derived from a quaternion r_i, S_i is the scaling matrix defined by the scale parameter s_i, and (x, y) is the point coordinate on the P_i. The Gaussian distribution on this plane is then computed using:

$$G(x,y) = \exp\left(-\frac{x^2+y^2}{2}\right). \tag{2}$$

To project the Gaussian onto the screen, 2DGS calculates the intersection point between the camera ray from the pixel $\mathbf{p}$ and the plane P_i, from which the corresponding (x, y) coordinates are obtained. These (x, y) values are used as inputs for texture mapping. Given an RGBA texture map Tex_i for i, the color contribution of i to pixel $\mathbf{p}$ is:

$$c_i(\mathbf{p}) = \text{Tex}_i[RGB](x,y) + SH_i(\vec{d}). \tag{3}$$

And the alpha contribution is:

$$\alpha_i(\mathbf{p}) = \text{Tex}_i[A](x,y) \cdot G_i(x,y). \tag{4}$$

Then the final color of $\mathbf{p}$ is:

$$c(\mathbf{p}) = \sum_{i=1}^{K} c_i(\mathbf{p})\alpha_i(\mathbf{p}) \prod_{j=1}^{i-1}(1-\alpha_j(\mathbf{p})). \tag{5}$$

For more details about textured Gaussians, please refer to (Chao et al. 2025; Svitov et al. 2024).

Figure 3: Quality comparison. Our method avoids the over-smoothing commonly observed in high-entropy regions with other approaches, ensuring detail fidelity across the entire scene and coherent visual appearance.

Adaptive Texture Allocation

While textured Gaussians improve the expression ability, existing methods often assign a fixed-resolution texture for every Gaussian, regardless of whether it contains local details. This leads to memory inefficiency in smooth regions and insufficient resolution in complex areas. Moreover, the training progress could be hard to find the global optimal solution when giving Gaussians high-resolution textures, resulting in an even worse rendering quality. To solve the above issues, we propose an adaptive texture allocation strategy. The core idea is to assign higher-resolution textures to Gaussians that contribute more to the rendered image or lie in regions of high geometric complexity. To avoid an overly large solution space, similar to (Chao et al. 2025), we first train the scene using vanilla 2DGS without texture. Then, we initialize each Gaussian with a 5×5 alpha texture and a 1×1 RGB texture. After a 1000-step warm-up to bridge the energy gap between 2DGS and textured rendering, we compute each Gaussian's contribution score S_i on image j (Fang and Wang 2024) as:

$$S_{ij} = \sum_{p \in \mathcal{I}_j} \delta \left(c_i(p) = c_{\max}(p) \right), \tag{6}$$

where $c_i(p)$ represents the contribution of Gaussian i at pixel p, $c_{\max}(x)$ denotes the highest contribution of a single Gaussian at pixel p, and δ is an indicator function. As S_{ij} counts the number of pixels in image j for which Gaussian i contributes the most, Gaussians with high S_{ij} are responsible for rendering the primary structure and texture of the scene and therefore require higher texture capacity to accurately model appearance details.

On the other hand, if a Gaussian is frequently observed and strongly contributes across multiple views, it is likely located in the core reconstruction region of the scene, where the need for fine details is more critical. Thus, we accumulate the contributions across all n training images, and allocate the Gaussian i a texture map with $T_i^{\text{init}} \times T_i^{\text{init}}$ resolution, where the size T_i^{init} is computed as:

$$T_i^{\text{init}} = \sum_{j=1}^{n} \log_2 \left(1 + \frac{S_{ij}}{\sigma} \right), \tag{7}$$

where σ is a hyperparameter to control the absolute resolution scale of the reconstructed scene.

As the shapes of Gaussians are mainly inherited from 2DGS and may not be well-suited for the textured Gaussians, in addition to contribution-based texture initialization, we further introduce a texture refinement that progressively increases the texture resolution for such Gaussians. Specifically, we adopt the position gradient $\nabla_p \mathcal{L}$, introduced in 2DGS, as an indicator for texture refinement. A larger gradient implies that the Gaussian needs to encode higher information entropy, thereby necessitating a higher-resolution texture to accurately capture the details. However, during the early stage of optimization, aggressively assigning large texture maps to Gaussians can significantly expand the solution space and destabilize training. To mitigate this, we use a progressive training strategy that prioritizes texture growth for low-entropy Gaussians in the early phase and gradually shifts focus to high-entropy ones in later stages. At each iteration, we only grow the texture size of Gaussians whose positional gradients fall within a dynamic range:

$$\tau_{\text{low}} < \nabla_p \mathcal{L} < \tau_{\text{high}}, \tag{8}$$

with

$$\begin{aligned} \tau_{\text{high}} &= \tau_{\text{preset}} + \left(\frac{t - t_0}{t_{\text{final}} - t_0} \right)^3 \cdot (\tau_{\max} - \tau_{\text{preset}}), \\ \tau_{\text{low}} &= 0.9 \cdot \tau_{\text{high}}, \end{aligned} \tag{9}$$

Dataset	Mip-NeRF360			Tanks&Temples			Deep Blending		
Method | Metric	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3DGS (Kerbl et al. 2023)	27.21	0.815	<u>0.214</u>	23.60	0.847	0.181	29.40	<u>0.903</u>	<u>0.243</u>
2DGS* (Huang et al. 2024a)	27.16	0.802	0.237	23.42	0.837	0.197	29.60	0.902	0.251
Mip Splatting (Yu et al. 2024)	**27.79**	**0.827**	0.203	23.75	**0.857**	0.157	29.46	<u>0.903</u>	<u>0.243</u>
BBSplat* (Svitov et al. 2024)	26.92	0.786	0.228	23.69	0.852	**0.150**	29.40	0.902	0.248
BBSplat + our_adaptive	27.11	0.785	0.234	<u>24.10</u>	<u>0.853</u>	0.155	<u>29.70</u>	0.902	0.244
Tex. Gau.* (Chao et al. 2025)	27.26	0.791	0.215	23.72	0.836	0.170	29.22	0.893	0.248
Ours	<u>27.71</u>	<u>0.816</u>	**0.197**	**24.15**	<u>0.853</u>	<u>0.151</u>	**29.80**	0.902	**0.221**
2DGS(10%)*	25.62	0.722	0.358	22.56	0.781	0.298	29.26	0.892	0.284
Ours(10%)	26.54	0.771	0.269	23.43	0.822	0.216	29.69	**0.905**	0.244
2DGS(1%)*	22.64	0.581	0.504	19.68	0.640	0.450	26.80	0.850	0.369
Ours(1%)	23.84	0.649	0.417	20.87	0.712	0.360	28.05	0.879	0.307

Table 1: Quantity comparison on rendering quality. * denotes re-implementation using the corresponding open-source codes. Our method performs consistently well in different scenes. Even when using fewer Gaussians (10% of the default optimized number of Gaussians), our method can still achieve competitive results.

where τ_{preset} is the initial threshold, t is the current training iteration, t_0 is the starting iteration, t_{final} is the ending iteration, and τ_{max} is the maximum positional gradient among all Gaussians at the current iteration.

For the selected Gaussians in the dynamic range, T_i^{t+1} for the next iteration is recomputed as:

$$T_i^{t+1} = T_i^t + \sqrt{\frac{\nabla_p \mathcal{L}}{\tau_{\text{low}}}}. \tag{10}$$

This schedule ensures that in the early phase, texture resources are allocated to relatively low-entropy regions to ensure convergence stability, while in the later stages, Gaussians responsible for fine details receive higher-resolution textures to support accurate fitting.

Mipmap-based Texture Scheduling

With adaptive texture allocation, textured Gaussians can get rid of visual incoherence caused by the different clarity of details. However, they still suffer from aliasing when viewing distant changes. To address this, we integrate a mipmap-style mechanism into the textured Gaussian pipeline. Unlike traditional mipmapping techniques, which typically apply only to RGB textures and require additional supersampling or post-filtering to handle geometric aliasing, our approach can leverage the inherent opacity channel of textured Gaussians to construct RGBA mipmap pyramids. This enables us to suppress aliasing both in appearance and geometry without the need for costly ray-based supersampling.

Specifically, for a Gaussian with texture size $T_i = n$, we construct a mipmap pyramid $\{\text{Tex}_i^l\}_{l=0}^L$ with $L = \lceil \log_2 n \rceil$. Level $l = 0$ is fixed as the original texture with resolution $n \times n$. Each subsequent level $l = k$ has resolution $2^{L-k} \times 2^{L-k}$, and is obtained by bilinear downsampling from the previous level $k - 1$.

At rendering time, the appropriate mip level l_{mip} is selected based on the projected footprint of each Gaussian in screen space. Given a target pixel located at (X_1, Y_1), we first get the intersection point of the camera ray on the corresponding 2D Gaussian plane, denoted as (x_1, y_1). We then consider its two neighboring pixels, $(X_1 + 1, Y_1)$ and $(X_1, Y_1 + 1)$, and compute their respective intersection points with the same Gaussian plane, denoted as (x_2, y_2) and (x_3, y_3). Next, we calculate the Euclidean distances D_{12} and D_{13} between the intersection point (x_1, y_1) and its neighbors (x_2, y_2), (x_3, y_3). The appropriate mipmap level l_{mip} is then selected based on the two distances:

$$l_{\text{mip}} = \min(\lceil \log_2 n \rceil, \max(0, \lfloor \log_2(\lambda_{\text{mip}} \cdot n \cdot \max(D_{12}, D_{13}))\rfloor)), \tag{11}$$

where λ_{mip} is a tunable hyperparameter.

The final rendered RGB color for Gaussian i at X_1, Y_1 is computed using the RGB channels as:

$$c_i(X_1, Y_1) = \text{Tex}_i^{l_{\text{mip}}}[\text{RGB}](x_1, y_1) + SH_i(\vec{d}), \tag{12}$$

where $\vec{d}$ is the view direction toward the Gaussian.

For the alpha channel, we compute the alpha as:

$$\alpha_i[X_1, Y_1] = G_i(x_1, y_1) \cdot \text{Tex}_i^{l_{\text{mip}}}[\text{A}](x_1, y_1), \tag{13}$$

where $G_i(\mathbf{u})$ is the Gaussian weight at coordinate $\mathbf{u}$ computed by Eq. 2. This modification integrates the Gaussian distribution with the alpha mipmap. It allows textured Gaussians to express geometry boundaries well when zoomed out without using supersampling.

Experiments

Implementation

Following Textured Gaussians (Chao et al. 2025), we first train a 2DGS model and use its reconstructed point cloud as the initialization for our method. During training, we set the threshold τ_{preset} to $1e{-}5$, σ to $H \times W/1e{+}4$ and the mipmap scaling factor λ_{mip} to 0.4. We also integrate our method with BBSplatting (Svitov et al. 2024) using the same parameters to demonstrate its generalization ability. All experiments are conducted on a single NVIDIA RTX 4090.

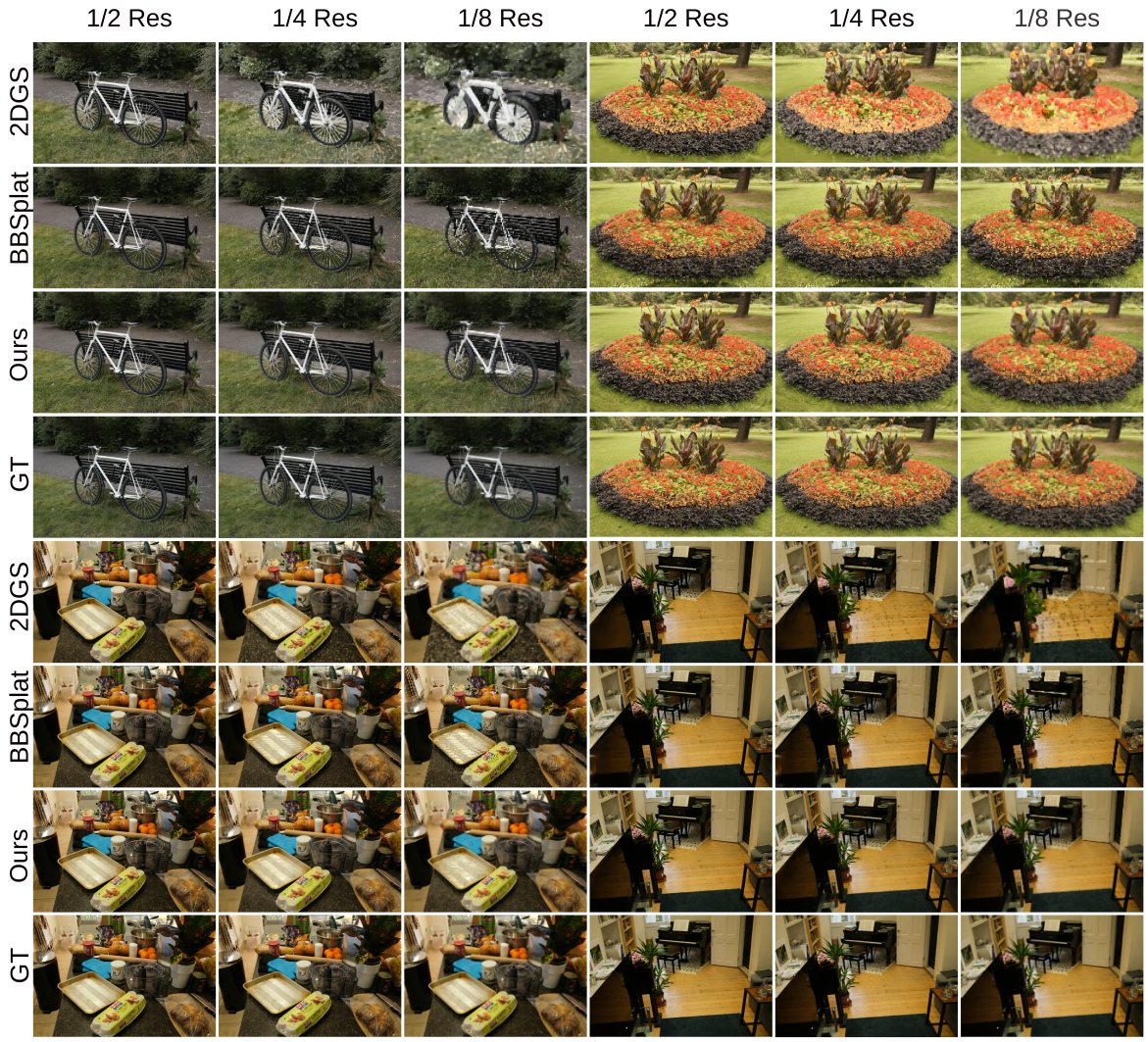

Figure 4: Quality comparison for anti-aliasing. Our method remains highly consistent with the ground truth while zooming out, providing a highly consistent visual experience when the viewing distance changes.

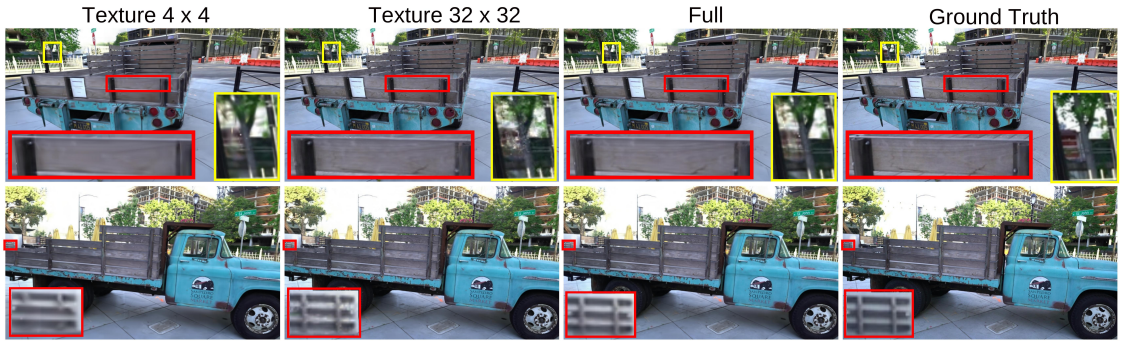

Figure 5: Effects of adaptive texture allocation. Using only low-resolution textures will result in blurring, while rashly increasing the texture resolution will result in noise. Only adaptive texture allocation can ensure high-quality rendering results.

Resolution	Full			1/2			1/4			1/8		
Method | Metric	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
2DGS* (Huang et al. 2024a)	27.16	0.802	0.237	25.82	0.805	0.184	21.31	0.661	0.233	18.20	0.502	0.289
BBSplat* (Svitov et al. 2024)	26.92	0.786	0.228	27.66	0.828	0.154	26.88	0.815	0.146	25.12	0.768	0.182
BBSplat + our_mipmap	26.92	0.786	0.229	27.82	0.833	0.151	27.91	0.851	0.116	**27.20**	0.857	0.111
Tex. Gaus.* (Chao et al. 2025)	27.26	0.791	0.215	26.12	0.793	0.171	21.87	0.662	0.227	19.00	0.529	0.293
Ours	**27.71**	**0.816**	**0.197**	**28.63**	**0.862**	**0.125**	**28.49**	**0.880**	**0.098**	27.12	**0.878**	**0.099**

Table 2: Quantity comparison for anti-aliasing. * denotes re-implementation using the corresponding open-source codes. Our method consistently performs well at various resolutions.

Resolution	Full			1/2			1/4			1/8		
Method | Metric	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Full	23.43	0.822	0.216	24.16	0.869	0.137	24.26	0.889	0.094	23.65	0.897	0.082
w/o mip.	23.41	0.822	0.215	23.87	0.861	0.141	22.91	0.856	0.128	21.15	0.827	0.151
w/o alpha mip.	23.42	0.822	0.216	23.96	0.862	0.140	23.21	0.864	0.120	21.59	0.841	0.137
w/o RGB mip.	23.43	0.822	0.216	24.09	0.865	0.137	23.99	0.880	0.102	23.35	0.882	0.097

Table 3: Ablation study on mipmap-based texture scheduling. Our full method consistently outperforms variants (without mipmaps, RGB mipmaps, or alpha mipmaps), especially under extreme downsampling (1/8).

Method | Metric	PSNR↑	SSIM↑	LPIPS↓	Storage/M↓
Full	23.43	0.822	0.216	45
Similar par.	23.29	0.820	0.219	42.2
4×4	23.10	0.801	0.258	25.1
8×8	23.26	0.817	0.230	36.7
16×16	23.19	0.818	0.210	78.2
32×32	23.02	0.814	0.203	223.0

Table 4: Ablation study on adaptive texture allocation. Similar par. stands for manually adjusting texture resolution to achieve similar parameters as our full method.

Quantitative and Qualitative Comparison

Following 3DGS, we test our method on 9 scenes from the Mip-NeRF 360 dataset (Barron et al. 2022), 2 scenes from the Tanks and Temples dataset (Knapitsch et al. 2017), and 2 scenes from the Deep Blending dataset (Mildenhall et al. 2021). We qualitatively compare our reconstruction results with other methods from two perspectives. First, we evaluate the ability to preserve fine details, as shown in Fig. 3. For a visually coherent reconstruction, all regions must appear consistently sharp, rather than having a mixture of clear and blurry areas. Our method achieves the highest fidelity to details, whereas other methods often show noticeable blurring in certain regions. Meanwhile, our method can also reduce texture storage given the same number of Gaussians. We then examine rendering quality under zoom-out conditions. As illustrated in the same figure, both baseline 2DGS and BBSplat suffer from degradation when zoomed out, such as grid-like artifacts or Moiré patterns. In contrast, our approach best preserves the global appearance and structural integrity of the scene, resulting in significantly improved visual coherence across varying camera distances. Quantitative comparisons in Tab. 1 and Tab. 2 further support these findings as our results consistently perform well.

Ablation Study

In this section, we further conduct several experiments to demonstrate the effectiveness of each module we propose. To more clearly show the effect of each module, we have used only 10% of the default optimized number of Gaussians.s As shown in Fig. 5, adaptive texture allocation significantly improves detail rendering. Fixed-resolution textures either blur high-entropy regions due to insufficient capacity or, when overly large, risk local optima and introduce noise (note that LPIPS is calculated via neural networks and can't distinguish noise very well). In contrast, our adaptive allocation strategy gradually increases texture resolution, avoiding optimization traps while accurately preserving fine details. Quantitative comparisons in Tab. 4 further confirm that even with a similar parameter budget, fixed-resolution baselines still underperform our adaptive approach. For anti-aliasing, without mipmap-based scheduling (whether for the RGB channel or the alpha channel), zoom-out views can suffer from aliasing, while our full method preserves consistent quality, as shown in Tab. 3.

Conclusion

In this paper, we propose **PATexGS**, a perceptual-adaptive texture scheduling framework designed to enhance visual coherence in textured Gaussian splatting. Our method introduces a texture allocation strategy that adaptively changes the texture resolution per-Gaussian based on rendering contribution and spatial gradients, as well as a mipmap texture scheduling mechanism to mitigate aliasing under view changes. Through extensive experiments, PATexGS has been proven consistently to outperform existing textured Gaussian baselines. Our future work may explore the disentanglement of texture and lighting in textured Gaussians, enabling relighting.

Acknowledgments

This work is supported by the Natural Science Foundation of China under Grant 62302174. The computation is completed in the HPC Platform of Huazhong University of Science and Technology. We also thank Farsee2 Technology Ltd for providing devices to support the validation of our method.

References

Barron, J. T.; Mildenhall, B.; Tancik, M.; Hedman, P.; Martin-Brualla, R.; and Srinivasan, P. P. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, 5855–5864.

Barron, J. T.; Mildenhall, B.; Verbin, D.; Srinivasan, P. P.; and Hedman, P. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5470–5479.

Barron, J. T.; Mildenhall, B.; Verbin, D.; Srinivasan, P. P.; and Hedman, P. 2023. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 19697–19705.

Cen, J.; Fang, J.; Yang, C.; Xie, L.; Zhang, X.; Shen, W.; and Tian, Q. 2025. Segment any 3d gaussians. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 1971–1979.

Chao, B.; Tseng, H.-Y.; Porzi, L.; Gao, C.; Li, T.; Li, Q.; Saraf, A.; Huang, J.-B.; Kopf, J.; Wetzstein, G.; et al. 2025. Textured gaussians for enhanced 3d scene appearance modeling. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 8964–8974.

Chen, D.; Li, H.; Ye, W.; Wang, Y.; Xie, W.; Zhai, S.; Wang, N.; Liu, H.; Bao, H.; and Zhang, G. 2024a. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*.

Chen, Y.; Chen, Z.; Zhang, C.; Wang, F.; Yang, X.; Wang, Y.; Cai, Z.; Yang, L.; Liu, H.; and Lin, G. 2024b. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 21476–21485.

Fang, G.; and Wang, B. 2024. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European Conference on Computer Vision*, 165–181. Springer.

Guédon, A.; and Lepetit, V. 2024. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5354–5363.

Hamdi, A.; Melas-Kyriazi, L.; Mai, J.; Qian, G.; Liu, R.; Vondrick, C.; Ghanem, B.; and Vedaldi, A. 2024. Ges: Generalized exponential splatting for efficient radiance field rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19812–19822.

Hu, W.; Wang, Y.; Ma, L.; Yang, B.; Gao, L.; Liu, X.; and Ma, Y. 2023. Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 19774–19783.

Huang, B.; Yu, Z.; Chen, A.; Geiger, A.; and Gao, S. 2024a. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, 1–11.

Huang, L.; Guo, J.; Dan, J.; Fu, R.; Wang, S.; Li, Y.; and Guo, Y. 2024b. Spectral-GS: Taming 3D Gaussian Splatting with Spectral Entropy. *arXiv preprint arXiv:2409.12771*.

Huang, Z.; and Gong, M. 2024. Textured-GS: Gaussian Splatting with Spatially Defined Color and Opacity. *arXiv preprint arXiv:2407.09733*.

Jiang, K.; Sivaram, V.; Peng, C.; and Ramamoorthi, R. 2025. Geometry Field Splatting with Gaussian Surfels. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 5752–5762.

Jiang, Y.; Shen, Z.; Hong, Y.; Guo, C.; Wu, Y.; Zhang, Y.; Yu, J.; and Xu, L. 2024. Robust dual gaussian splatting for immersive human-centric volumetric videos. *ACM Transactions on Graphics (TOG)*, 43(6): 1–15.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4): 139–1.

Khan, M.; Fazlali, H.; Sharma, D.; Cao, T.; Bai, D.; Ren, Y.; and Liu, B. 2024. AutoSplat: Constrained Gaussian Splatting for Autonomous Driving Scene Reconstruction. arXiv:2407.02598.

Knapitsch, A.; Park, J.; Zhou, Q.-Y.; and Koltun, V. 2017. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4): 1–13.

Kocabas, M.; Chang, J.-H. R.; Gabriel, J.; Tuzel, O.; and Ranjan, A. 2024. Hugs: Human gaussian splats. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 505–515.

Liang, Z.; Zhang, Q.; Hu, W.; Zhu, L.; Feng, Y.; and Jia, K. 2024. Analytic-splatting: Anti-aliased 3d gaussian splatting via analytic integration. In *European conference on computer vision*, 281–297. Springer.

Lin, J.; Li, Z.; Tang, X.; Liu, J.; Liu, S.; Liu, J.; Lu, Y.; Wu, X.; Xu, S.; Yan, Y.; et al. 2024. Vastgaussian: Vast 3d gaussians for large scene reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5166–5175.

Lyu, X.; Sun, Y.-T.; Huang, Y.-H.; Wu, X.; Yang, Z.; Chen, Y.; Pang, J.; and Qi, X. 2024. 3dgsr: Implicit surface reconstruction with 3d gaussian splatting. *ACM Transactions on Graphics (TOG)*, 43(6): 1–12.

Mildenhall, B.; Srinivasan, P. P.; Ortiz-Cayon, R.; Kalantari, N. K.; Ramamoorthi, R.; Ng, R.; and Kar, A. 2019. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (ToG)*, 38(4): 1–14.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1): 99–106.

Rong, V.; Chen, J.; Bahmani, S.; Kutulakos, K. N.; and Lindell, D. B. 2025. Gstex: Per-primitive texturing of 2d gaussian splatting for decoupled appearance and geometry modeling. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3508–3518. IEEE.

Schonberger, J. L.; and Frahm, J.-M. 2016. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4104–4113.

Schönberger, J. L.; Zheng, E.; Frahm, J.-M.; and Pollefeys, M. 2016. Pixelwise view selection for unstructured multi-view stereo. In *European conference on computer vision*, 501–518. Springer.

Shao, Z.; Wang, Z.; Li, Z.; Wang, D.; Lin, X.; Zhang, Y.; Fan, M.; and Wang, Z. 2024. Splattingavatar: Realistic real-time human avatars with mesh-embedded gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1606–1616.

Song, Y.; Lin, H.; Lei, J.; Liu, L.; and Daniilidis, K. 2024. Hdgs: Textured 2d gaussian splatting for enhanced scene rendering. *arXiv preprint arXiv:2412.01823*.

Svitov, D.; Morerio, P.; Agapito, L.; and Del Bue, A. 2024. BillBoard Splatting (BBSplat): Learnable Textured Primitives for Novel View Synthesis. *arXiv preprint arXiv:2411.08508*.

Wang, Y.; Zhong, N.; Chen, M.; Wang, L.; and Guo, Y. 2024. Tangram-splatting: Optimizing 3d gaussian splatting through tangram-inspired shape priors. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 3075–3083.

Williams, L. 1983. Pyramidal parametrics. In *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, 1–11.

Xu, T.-X.; Hu, W.; Lai, Y.-K.; Shan, Y.; and Zhang, S.-H. 2024. Texture-gs: Disentangling the geometry and texture for 3d gaussian splatting editing. In *European Conference on Computer Vision*, 37–53. Springer.

Yan, Z.; Low, W. F.; Chen, Y.; and Lee, G. H. 2024. Multi-scale 3d gaussian splatting for anti-aliased rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20923–20931.

Yu, Z.; Chen, A.; Huang, B.; Sattler, T.; and Geiger, A. 2024. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 19447–19456.

Yu, Z.; Sattler, T.; and Geiger, A. 2024. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (ToG)*, 43(6): 1–13.

Zhou, T.; Tucker, R.; Flynn, J.; Fyffe, G.; and Snavely, N. 2018. Stereo magnification: learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)*, 37(4): 1–12.

Zhu, J.; Yue, J.; He, F.; and Wang, H. 2025. 3D Student Splatting and Scooping. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 21045–21054.