Textured Gaussians for Enhanced 3D Scene Appearance Modeling

Brian Chao [1,2] Hung-Yu Tseng [2] Lorenzo Porzi [2] Chen Gao [2] Tuotuo Li [2] Qinbo Li [2]
Ayush Saraf [2] Jia-Bin Huang [2,3] Johannes Kopf [2] Gordon Wetzstein [1] Changil Kim [2]

[1] Stanford University [2] Meta [3] University of Maryland

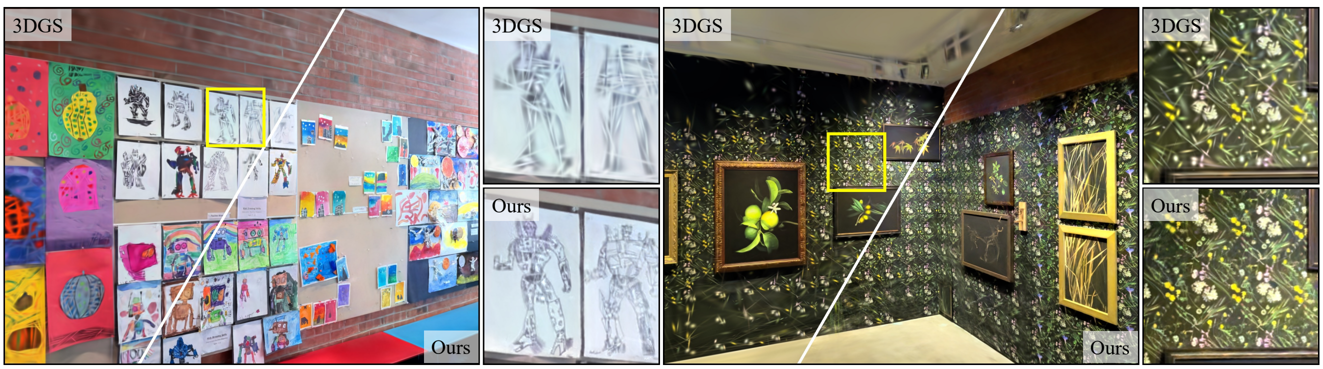

Figure 1. **Textured Gaussians compared to 3D Gaussian Splatting (3DGS).** Our RGBA Textured Gaussians enhance 3D scene appearance modeling, leading to improved rendering quality while using the same number of Gaussians compared to 3DGS [25]. Above, we show that Textured Gaussians faithfully reconstruct the fine details of scenes.

Abstract

3D Gaussian Splatting (3DGS) has recently emerged as a state-of-the-art 3D reconstruction and rendering technique due to its high-quality results and fast training and rendering time. However, pixels covered by the same Gaussian are always shaded in the same color up to a Gaussian falloff scaling factor. Furthermore, the finest geometric detail any individual Gaussian can represent is a simple ellipsoid. These properties of 3DGS greatly limit the expressivity of individual Gaussian primitives. To address these issues, we draw inspiration from texture and alpha mapping in traditional graphics and integrate it with 3DGS. Specifically, we propose a new generalized Gaussian appearance representation that augments each Gaussian with alpha (A), RGB, or RGBA texture maps to model spatially varying color and opacity across the extent of each Gaussian. As such, each Gaussian can represent a richer set of texture patterns and geometric structures, instead of just a single color and ellipsoid as in naïve Gaussian Splatting. Surprisingly, we found that the expressivity of Gaussians can be greatly improved by using alpha-only texture maps, and further augmenting Gaussians with RGB texture maps achieves the highest expressivity. We validate our method on a wide variety of standard benchmark datasets and our own custom captures at both the object and scene levels. We demonstrate image quality improvements over existing methods while using a similar or lower number of Gaussians.

1. Introduction

Neural rendering [52] achieves unprecedented novel-view synthesis and 3D reconstruction quality, offering solutions to a myriad of applications ranging from surface reconstruction [30, 53], SLAM [42, 63], material estimation and relighting [49, 62], to virtual teleportation and human avatar animation [50, 54]. Recently, 3D Gaussian Splatting (3DGS) [25] emerged as a state-of-the-art novel-view synthesis technique, with desirable properties such as high image quality results, fast training and rendering time, and an explicit primitives-based representation. Subsequent works in Gaussian Splatting have focused on topics such as improving surface reconstruction quality [14, 21], scene editing [8, 15], dynamic scene reconstruction [46, 57], and 3D generation [40, 51]. Despite the success, 3DGS sometimes fails to model detailed appearances, as demonstrated in Figure 1.

In 3DGS, each Gaussian can only represent a single color and the shape of an ellipsoid given a camera viewpoint. This dramatically limits the set of appearances and shapes a single Gaussian can represent. To address the issue, Huang and Gong [22] modified the viewing direction calculation in 3DGS such that pixels covered by the same Gaussian exhibit a smooth color and opacity variation. However, this color variation across a single Gaussian is minuscule due to the spherical harmonics representation of colors, preventing them from representing complex textures. Xu et al. [56] proposed only using Gaussians to represent scene geometry and leveraged a learned UV mapping module and a global 2D RGB texture map to project textures onto Gaussian surfaces. However, the unit-sphere representation of textures significantly constrains the capacity of their model, and thus, their method fails to reconstruct objects with complex geometry or large-scale scenes.

Our method builds upon 3DGS and draws inspiration from mesh-based 3D representations that model appearance using texture mapping. As shown in Figure 2, we augment each Gaussian with its alpha, RGB, or RGBA texture map so that each Gaussian is capable of representing a rich set of textures and shapes. We call this representation *Textured Gaussians*. With RGB textures, individual Gaussians can represent higher-frequency color variations. The alpha maps allow Gaussians to represent a wider variety of shapes, instead of just ellipsoids. To achieve this, we build custom CUDA kernels that perform ray-Gaussian intersection and texture mapping and integrate them with 3DGS. Consequently, our method not only inherits all desirable properties of 3DGS, such as fast training and rendering time and explicit 3D representation, but also substantially improves rendered image quality, especially with a small number of Gaussians.

In summary, our contributions include:

- Introduction of a generalized appearance model for 3D Gaussians that handles spatially varying color and opacity by augmenting 3D Gaussians with alpha, RGB, or RGBA texture maps.
- Validation that Textured Gaussians improves 3DGS on a wide variety of *both* object-level and scene-level datasets.
- Demonstration that our method greatly outperforms 3DGS when the number of Gaussians is small and achieves better performance than 3DGS with alpha-only textures given the same model size.

2. Related Work

Novel-View Synthesis (NVS) and Neural Rendering. NVS tackles the problem of generating accurate renderings of 3D scenes from unseen viewpoints given a set of training images and camera poses. NVS traces back to the Structure from Motion (SfM) works [4, 17, 48] and Multiview Stereo (MVS) [13, 44], which served as the basis of several groundbreaking NVS methods [6, 11, 18, 27]. However, these approaches generally require storing hundreds and thousands of input images for reprojection and blending, which leads to massive memory requirements and sometimes fails to reconstruct undersampled regions.

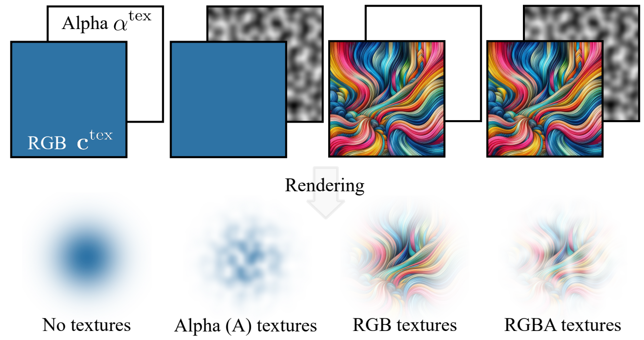

Figure 2. **Variants of our Textured Gaussians model.** Textured Gaussians encapsulate four kinds of color and opacity spatial variations. The top row of the figure shows the texture map associated with each Gaussian, and the bottom row shows the rendered Textured Gaussians. The constant-color and constant-alpha model (no textures) corresponds to the origianl 3DGS formation, which can only represent a single color up to a Gaussian falloff factor within the Gaussian extent. Textured Gaussians can already model spatially varying colors using only alpha textures since each pixel can be alpha-composited differently. The model achieves maximum expressivity when leveraging the full RGBA texture map, where each Gaussian is capable of representing complex shapes and high frequency textures.

Recent advances in neural rendering [52] have made great strides in improving the quality of 3D reconstruction and novel-view rendering. Neural rendering algorithms can be roughly classified by the underlying 3D representation, ranging from point clouds [1, 27, 55], voxels [19, 43, 60], meshes [10, 16, 20, 24, 31], to implicit representations using MLPs [33, 35, 47]. Recently, 3D Gaussian Splatting [25] (3DGS) has emerged as the state-of-the-art NVS technique. 3DGS achieves high image quality for NVS, is fast to optimize and render, and leverages an explicit Gaussian primitive representation which makes the method suitable for a wide variety of tasks such as surface extraction [14, 21], human avatar manipulation [39, 45], and object and scene editing [15, 58].

Our method augments 3DGS with alpha and texture mapping commonly used in mesh-based 3D representations. During optimization and rendering, we intersect outgoing rays from pixels with Gaussians in the scene. Intersection points are then used to query per-Gaussian texture map values via UV mapping and compute RGB color and alpha blending values. This enables each Gaussian to represent complex textures and shapes, significantly improving NVS quality.

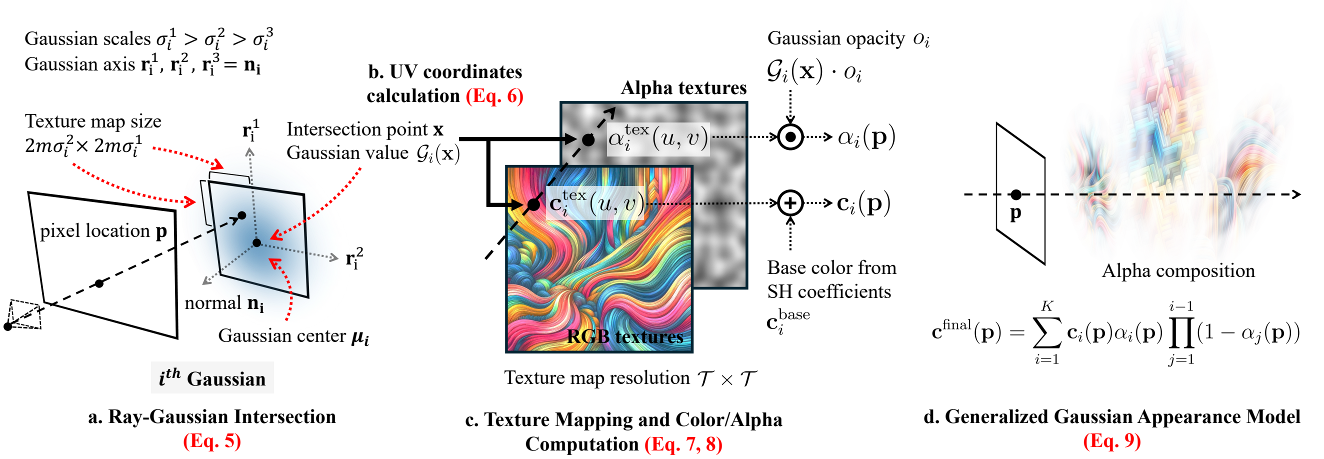

Figure 3. **Textured Gaussians model pipeline.** Our method consists of three major components: ray-Gaussian intersection, RGBA texture mapping, and a generalized Gaussian appearance model. To render the color of a pixel $\mathbf{p}$, we first trace a ray from the camera center $\mathbf{o}$ to the pixel to intersect with 3D Gaussians in the scene. Then, we query texture and alpha values, $\mathbf{c}_{\text{tex}}$ and α_{tex}, from the per-Gaussian RGBA texture maps using the ray-Gaussian intersection point $\mathbf{x}$. Finally, given the retrieved spatially varying color and alpha values, we alpha-composite the color and alpha values of Gaussians that are hit by the pixel ray using the generalized Gaussian appearance model.

Appearance Modeling in 3D Gaussian Splatting. In 3DGS, scene geometry is represented by the position, rotation, and scale of 3D Gaussians. By adjusting these properties, 3DGS can represent intricate geometric structures. Appearance, on the other hand, is modeled using per-Gaussian opacity values and spherical harmonics coefficients. However, pixels within a projected Gaussian are always shaded with the same color up to a Gaussian falloff factor, greatly limiting the expressivity of individual Gaussians. Therefore, recent works have explored ways that allow Gaussians to represent spatially varying features. Huang and Gong [22] defined per-Gaussian SH coefficients for color and opacity and used ray-Gaussian intersections to determine the viewing directions within the Gaussian extent. This enables smooth variation of colors and opacities across pixels covered by a single Gaussian but prevents the reconstruction of higher-frequency details. Xu et al. [56] disentangled appearance and geometry in 3DGS for texture editing, but their method is limited to object-centric scenes with simple geometry due to the unit-sphere parameterization of textures.

Instead of using a global texture map, we enhance each 3D Gaussian with a local texture map on top of the SH coefficients. This allows for a more flexible texture optimization since individual Gaussians are not tied to a shared global texture map. Our method can, therefore, reconstruct objects with complex structures and real-world scenes. Furthermore, each Gaussian can also represent various shapes with the alpha channel in the texture map.

Memory Efficient Gaussian Splatting. In order to achieve even faster rendering time and compact storage requirements for potential applications on edge devices, there has been a plethora of recent work focusing on optimizing memory-efficient Gaussian Splatting models [2]. These algorithms either prune Gaussians and perform quantization of Gaussian attributes [12, 29, 34, 36–38] or exploit the structural relationships between Gaussians [9, 32].

Our work is orthogonal and complementary to these works since we focus on improving the appearance modeling of Gaussians by redistributing the model size budget to the per-Gaussian texture maps given any optimized 3DGS model. These model compression algorithms can be easily integrated with the 3DGS pretraining stage in our optimization pipeline to achieve further gains in compactness.

Concurrent Work. Rong et al. [41] concurrently developed GStex, where a set of fixed-size texels is distributed to the surface of the 2D Gaussian disks based on the scale of the Gaussians. However, their method does not allow each Gaussian to represent different shapes due to the absence of the alpha channel in their texture map.

3. Method

3.1. 3D Gaussian Splatting Model

In 3DGS [25], 3D scenes are represented with 3D Gaussians, and images are rendered using differentiable volume splatting. Specifically, 3DGS explicitly defines 3D Gaussians by their 3D covariance matrix $\boldsymbol{\Sigma}_i \in \mathbb{R}^{3\times 3}$ and center $\boldsymbol{\mu}_i \in \mathbb{R}^3$ (the index i indicating the i^{th} Gaussian), where the

3D Gaussian function value at point $\mathbf{x} \in \mathbb{R}^3$ is defined by:

$$\mathcal{G}_i(\mathbf{x}) = \exp(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)\boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)) \tag{1}$$

where the covariance matrix $\boldsymbol{\Sigma} = \mathbf{RSS}^\top\mathbf{R}^\top$ is factorized into the rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ and the scale matrix $\mathbf{S} \in \mathbb{R}^{3\times3}$. To render a 2D image from a 3D Gaussians representation, 3D Gaussians are transformed from world coordinates to camera coordinates via a world-to-camera transformation matrix $\mathbf{W} \in \mathbb{R}^{3\times3}$ and projected to the 2D image plane via a local affine transformation $\mathbf{J} \in \mathbb{R}^{3\times3}$. The transformed 3D covariance $\boldsymbol{\Sigma}'$ can be calculated as:

$$\boldsymbol{\Sigma}' = \mathbf{JW\Sigma W}^\top\mathbf{J}^\top \tag{2}$$

The covariance Σ_{2D} of the 2D Gaussian $\mathcal{G}^{\text{2D}}$ splatted on the image plane can be approximated as extracting the first two rows and columns of the transformed 3D covariance, $\Sigma_{\text{2D}} = [\Sigma']_{\{1,2\},\{1,2\}} \in \mathbb{R}^{2\times2}$ using the matrix minor notation.

To render the color of a pixel $\mathbf{p} \in \mathbb{R}^3$, the colors associated with each Gaussian are alpha-composited from front to back following the conventional volume rendering equation:

$$\mathbf{c}(\mathbf{p}) = \sum_{i=1}^{K} \mathbf{c}_i\alpha_i \prod_{j=1}^{i-1}(1 - \alpha_j) \tag{3}$$

where i is the index of the Gaussians, $\mathbf{c}_i$ is the color of each Gaussian computed from the per-Gaussian spherical harmonic coefficients and viewing direction, and α_i is the alpha value computed from the opacity o_i associated with each Gaussian and the 2D Gaussian value evaluated at pixel location $\mathbf{p}$:

$$\alpha_i = \mathcal{G}_i^{\text{2D}}(\mathbf{p}) \cdot o_i \tag{4}$$

The attributes (center, rotation, scale, opacity, and spherical harmonic coefficients) of the Gaussians are optimized with gradient descent using photometric losses on the rendered 2D images.

3.2. Textured Gaussians

From the 3DGS appearance model defined in Eq. 3, we observe two properties of Gaussian Splatting:

1. Pixels covered by the same Gaussian are shaded with the same color up to a Gaussian falloff scaling factor.
2. The per-Gaussian opacity only allows Gaussians to represent ellipsoidal shapes.

These two properties greatly restrict the expressivity of individual 3D Gaussian primitives. To allow each Gaussian primitive to represent complex appearances and shapes, we assign a fixed-resolution 2D texture map of size $\mathcal{T} \times \mathcal{T} \times \mathcal{K}, \mathcal{T} \in \mathbb{N}, \mathcal{K} \in \{1, 3, 4\}$ to each Gaussian and transform each Gaussian into a Textured Gaussian. As shown in Figure 2, $\mathcal{K} = 1, 3, 4$ correspond to alpha, RGB, and RGBA texture maps, respectively. The shape of a Textured Gaussian is thus determined by the spatially varying opacity defined by the product of the per-Gaussian opacity and the alpha channel of the texture map. The appearance of a Textured Gaussian is represented by 1) the RGB channels of the *spatially varying* textured map and 2) a set of *spatially constant* spherical harmonic coefficients (SH). Intuitively, the spatially constant SH coefficients represent the low-frequency textures plus view-dependent colors like specular effects, while the spatially varying texture map represents the higher-frequency spatial variations of the texture. The texture is then mapped to the plane $\mathcal{P}$ defined by the two major axes of the i^{th} 3D Gaussian that is centered at $\boldsymbol{\mu}_i \in \mathbb{R}^3$, as shown in Figure 3. The normal $\mathbf{n}_i \in \mathbb{R}^3$ of this plane thus corresponds to the axis with the smallest scale.

For each pixel $\mathbf{p} \in \mathbb{R}^3$ that is currently being rendered, we cast a ray from the camera origin $\mathbf{o} \in \mathbb{R}^3$ to the center of the pixel $\mathbf{p}$ and intersect it with the plane $\mathcal{P}$. The intersection point $\mathbf{x} \in \mathbb{R}^3$ can be calculated as:

$$\mathbf{x} = \mathbf{o} + \frac{(\boldsymbol{\mu}_i - \mathbf{o}) \cdot \mathbf{n}_i}{(\mathbf{p} - \mathbf{o}) \cdot \mathbf{n}_i} \cdot \frac{\mathbf{p} - \mathbf{o}}{\|\mathbf{p} - \mathbf{o}\|} \tag{5}$$

Given the intersection point $\mathbf{x}$ with $\mathcal{P}$, we perform UV mapping and bilinear interpolation on the texture map associated with the i^{th} Gaussian to query the texture color and alpha value at the UV coordinates $u, v \in \mathbb{R}$, denoted by $\mathbf{c}_i^{\text{tex}}(u, v) \in \mathbb{R}^3$ and $\alpha_i^{\text{tex}}(u, v) \in \mathbb{R}$. Specifically, the UV coordinates (u, v) of the intersection point on the texture map can be calculated as:

$$\begin{aligned} u &= \frac{m \cdot \sigma_i^1 + (\mathbf{x} - \boldsymbol{\mu}_i) \cdot \mathbf{r}_i^1}{2 \cdot m \cdot \sigma_i^1} \cdot (\mathcal{T} - 1) \\ v &= \frac{m \cdot \sigma_i^2 + (\mathbf{x} - \boldsymbol{\mu}_i) \cdot \mathbf{r}_i^2}{2 \cdot m \cdot \sigma_i^2} \cdot (\mathcal{T} - 1) \end{aligned} \tag{6}$$

where $\sigma_i^1, \sigma_i^2 \in \mathbb{R}$ are the scales of the two major axis of the i^{th} Gaussian, $\mathbf{r}_i^1, \mathbf{r}_i^2 \in \mathbb{R}^3$ are the normalized directions of the two major axes of the i^{th} Gaussian, and $m \in \mathbb{R}$ is a scalar multiplier that determines the extent of the texture map with respect to each Gaussian.

Combined with the color computed from the SH coefficients, which we call $\mathbf{c}_i^{\text{base}}$ (the same as the color component in the original 3DGS appearance model in Eq. 3), the final color contribution of the i-th Gaussian to pixel $\mathbf{p}$ is defined by:

$$\mathbf{c}_i(\mathbf{p}) = \mathbf{c}_i^{\text{base}} + \mathbf{c}_i^{\text{tex}}(u, v) \tag{7}$$

and the alpha value of the i-th Gaussian at pixel $\mathbf{p}$ is defined by:

$$\alpha_i(\mathbf{p}) = \alpha_i^{\text{tex}}(u, v) \cdot \mathcal{G}_i(\mathbf{x}) \cdot o_i \tag{8}$$

Finally, to render the color of a pixel $\mathbf{p}$, we modify the 3DGS appearance model in Eq. 3 to incorporate the spa-

Method	Blender [33]	Mip-NeRF 360 [33]	DTU [23]	Tanks and Temples [26]	Deep Blending [18]
Mip-NeRF 360 [3]	30.34 / 0.9510 / 0.0600	**27.69** / 0.7920 / 0.2370	—— / —— / ——	22.22 / 0.7590 / 0.2570	**29.40 / 0.9010 / 0.2450**
Instant-NGP [35]	32.20 / 0.9590 / 0.0550	25.30 / 0.6710 / 0.3710	—— / —— / ——	21.72 / 0.7230 / 0.3300	23.62 / 0.7970 / 0.4230
Xu et al. [56]	—— / —— / ——	—— / —— / ——	30.03 / —— / 0.1440	—— / —— / ——	—— / —— / ——
3DGS*	33.08 / 0.9671 / 0.0440	27.26 / **0.8318** / 0.1871	33.54 / 0.9697 / **0.0551**	24.18 / 0.8541 / 0.1754	28.04 / 0.8940 / 0.2707
Ours	**33.24 / 0.9674 / 0.0428**	27.35 / 0.8274 / **0.1858**	**33.61 / 0.9699** / 0.0556	**24.26 / 0.8542 / 0.1684**	28.33 / 0.8908 / 0.2699
3DGS* (10%)	31.47 / 0.9590 / 0.0594	25.77 / 0.7796 / 0.2860	32.71 / 0.9627 / 0.0811	22.82 / 0.8020 / 0.2728	27.64 / 0.8853 / 0.3101
Ours (10%)	**32.14 / 0.9629 / 0.0489**	**26.32 / 0.7976 / 0.2323**	**32.74 / 0.9661 / 0.0582**	**23.41 / 0.8259 / 0.2122**	**27.98 / 0.8898 / 0.2804**
3DGS* (1%)	26.89 / 0.9160 / 0.1165	22.37 / 0.6293 / 0.4774	30.88 / 0.9320 / 0.1581	19.90 / 0.6736 / 0.4406	23.97 / 0.8167 / 0.4337
Ours (1%)	**28.11 / 0.9343 / 0.0849**	**23.73 / 0.7064 / 0.3365**	**32.43 / 0.9627 / 0.0694**	**21.10 / 0.7399 / 0.3104**	**24.83 / 0.8454 / 0.3552**

Table 1. **Quantitative comparisons of different novel view synthesis methods.** We boldface the best-performing model and underline the second best in terms of different metrics (PSNR ↑ / SSIM ↑ / LPIPS ↓). Our Textured Gaussians model achieves better performance than 3DGS* across most metrics. With fewer Gaussians (1% and 10% of the default optimized number of Gaussians), our method significantly outperforms 3DGS*, achieving strictly better performance across all metrics as indicated by the bold text.

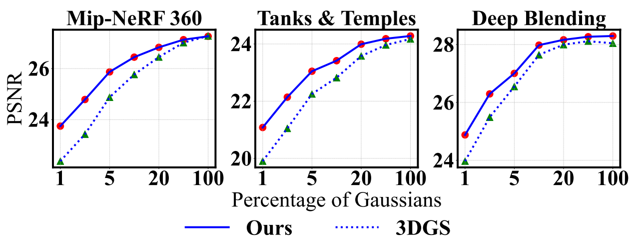

Figure 4. **NVS performance using varying numbers of Gaussians.** When using the same number of Gaussians as 3DGS*, our Textured Gaussians achieve better novel view synthesis results on all five benchmark datasets in PSNR. Here, we show the quantitative performance of 3DGS* and Textured Gaussians models optimized with varying number of Gaussians on the three scene-level datasets.

tially varying texture and opacity:

$$\mathbf{c}^{\text{final}}(\mathbf{p}) = \sum_{i=1}^{K} \mathbf{c}_i(\mathbf{p})\alpha_i(\mathbf{p}) \prod_{j=1}^{i-1}(1-\alpha_j(\mathbf{p})) \tag{9}$$

Eq. 9 is a generalized formulation of 3D Gaussian appearance and encapsulates different variants of Textured Gaussians. For example, $\mathbf{c}_i^{\text{tex}} = 0$ and $\alpha_i^{\text{tex}} = 1$ correspond to the original 3DGS model.

3.3. Optimization of Textured Gaussians

Following 3DGS [25], we optimize our model to minimize the weighted photometric loss:

$$\mathcal{L} = \lambda\mathcal{L}_1 + (1-\lambda)\mathcal{L}_{\text{SSIM}} \tag{10}$$

where $\lambda = 0.8$.

Our optimization procedure consists of two stages. We first optimize a 3DGS model for 30000 iterations. The learning rates and adaptive density control (ADC) parameters are reported in the Supplementary Materials. In the second stage, we initialize all attributes of the Gaussians with the optimized vanilla 3DGS model, and jointly optimize them with the per-Gaussian 2D texture maps for another 30000 iterations. We disable the ADC control in the second stage to control the number of Gaussians for fair baseline comparisons. This two-stage optimization process greatly speeds up convergence and improves image quality, since jointly optimizing all parameters is a highly ill-posed problem.

We implement custom CUDA kernels to perform fast ray-Gaussian intersection, UV mapping, and color composition. All experiments are conducted on clusters of Nvidia H100 GPUs. Please refer to the Supplemental Materials for more details on implementation and optimization.

4. Results and Analysis

We show selected qualitative and quantitative results in this section and refer the readers to the Supplemental Materials for an extensive set of results and video renderings.

Datasets and Evaluation Protocols. We evaluate the novel-view synthesis performance of our method on the 8 synthetic scenes from the Blender dataset [33], all 9 scenes from the Mip-NeRF 360 dataset [3], 5 scenes from the DTU dataset [23], 2 scenes from the Tanks and Temples dataset [26], and 2 scenes from the Deep Blending dataset [18]. Please refer to the Supplemental Materials for a detailed description of the dataset preparation process.

For comparisons with Mip-NeRF 360 [3] and Instant-NGP [35], we refer to Kerbl et al. [25] for results in scene-level datasets [3, 18, 26], and refer to NeRFBaselines [28] for results on the Blender dataset [33]. Results on the DTU dataset was not reported in NeRFBaselines so we omit the corresponding entries in Table 1. Xu et al. [56] pointed out in Section 5.4 of their paper that their algorithm cannot handle scene-level datasets and objects with complex structures, so we only show their reported results on the DTU

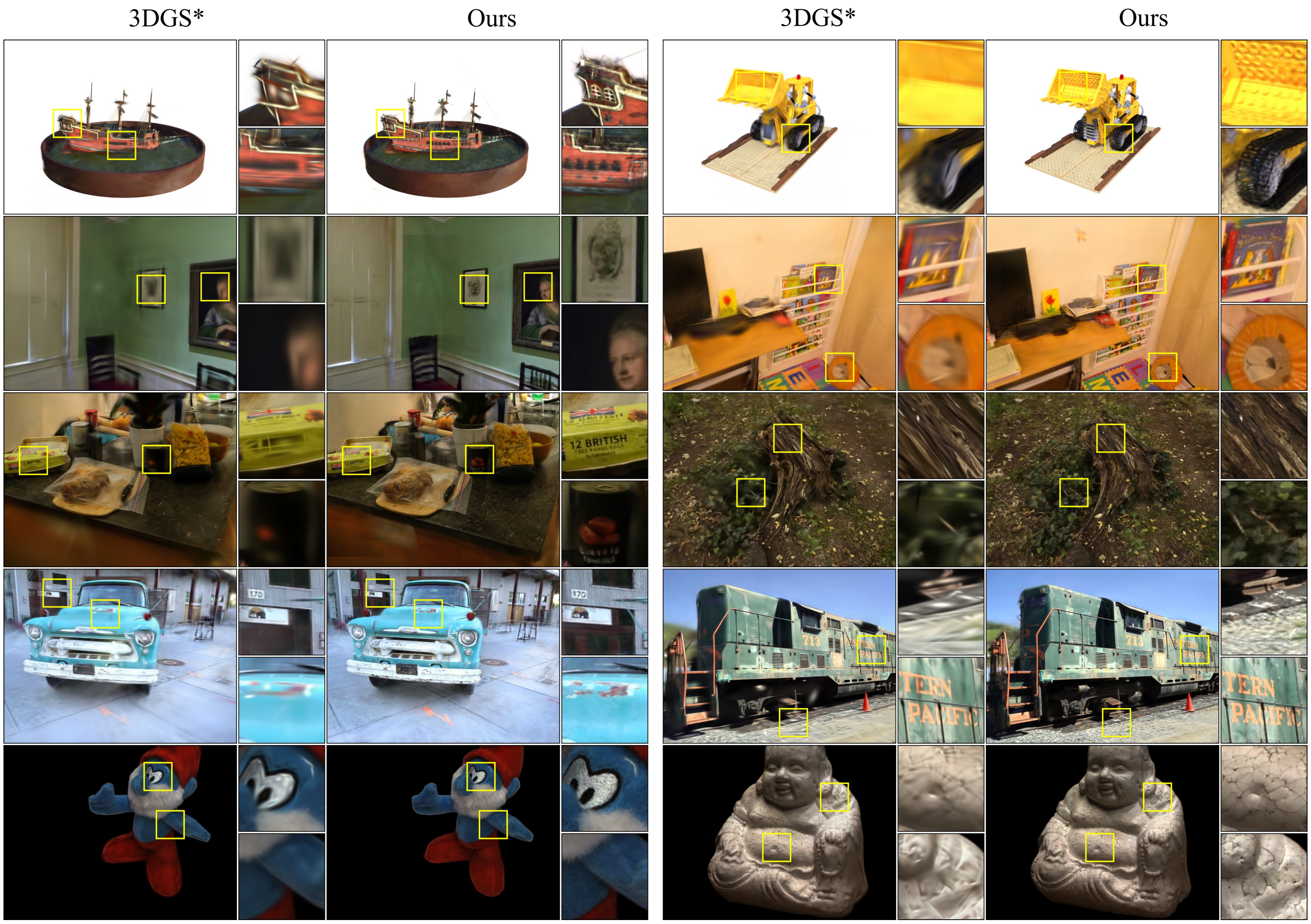

Figure 5. **Qualitative NVS results of benchmark datasets.** Novel-view rendering results for both object-level and scene-level standard benchmark datasets. Given the same small number of Gaussians (on average 2.5k and 39k Gaussians for object-level and scene-level datasets, respectively), 3DGS* fails to reconstruct high-frequency textures and complex shapes while our RGBA Textured Gaussians model succeeds. Please refer to the Supplemental Materials for results of our models optimized using varying number of Gaussians.

dataset.

In addition to standard benchmark test scenes, we also capture our own set of scenes that contain artworks with highly detailed textures to help demonstrate the effectiveness of our method.

3DGS Baseline. We build our method on top of our own implementation of a slightly modified 3DGS that closely follows the algorithm described in Gaussian Opacity Fields [61]. Specifically, we use a ray-based formulation that allows for the exact evaluation of 3D Gaussian values without approximating splatted 3D Gaussians as 2D Gaussians. In addition, we used a revised densification strategy [59, 61]. For fair comparisons with our method, we report the performance of our 3DGS implementation and label it as 3DGS* throughout the results section and refer the readers to the Supplemental Materials for quantitative results reported in the original 3DGS paper [25].

4.1. Quantitative Results

In Table 1, we show quantitative comparisons in terms of PSNR/SSIM/LPIPS between our Textured Gaussians model using full RGBA textures and 3DGS* with the same number of Gaussians, and results of other baseline methods. Our method generally achieves better results than 3DGS* and all other methods, since Texture Gaussians are more expressive than vanilla 3D Gaussians due to the added texture maps.

In the bottom half of Table 1, we also show comparisons between our method and 3DGS* when using a fraction of the default number of Gaussians, denoted by the percentage in parentheses. For our models with different numbers of Gaussians, we distribute a fixed amount of texels to all Gaussians (i.e., the same memory overhead for all Textured Gaussians models compared with 3DGS*). Therefore, the texture map resolution of models with fewer Gaussians will be larger. We see that our method truly outshines 3DGS*

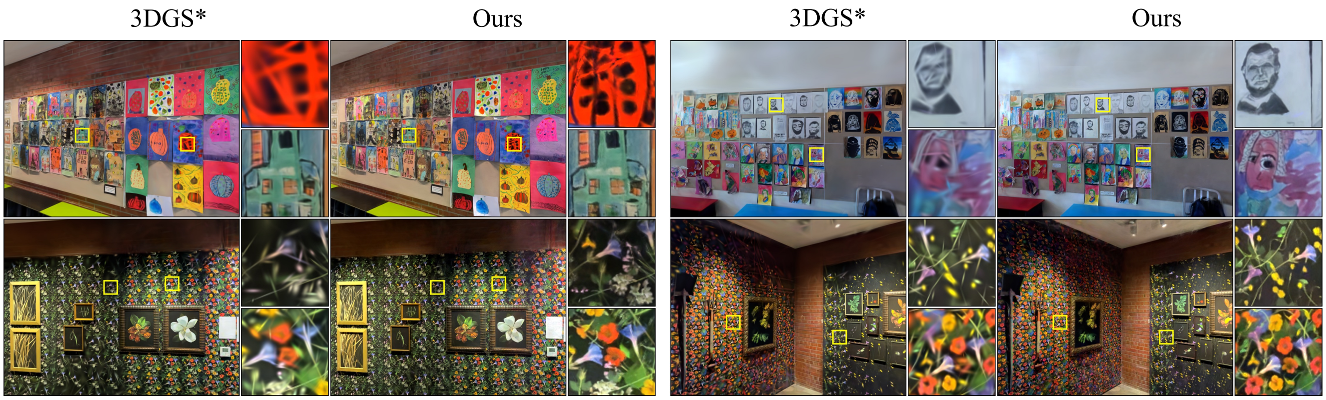

Figure 6. **Qualitative NVS results of custom datasets.** Our RGBA Textured Gaussians model achieves sharper reconstruction compared to 3DGS* when using the same small number of Gaussians (on average around 100k Gaussians).

Method	3DGS*		Alpha-only		RGBA	
	PSNR	#GS (M)	PSNR	#GS (M)	PSNR	#GS (M)
Blender	33.08	0.27	33.12	0.19	33.03	0.21
Mip-NeRF 360	27.26	4.4	27.37	3.1	27.26	3.5
DTU	33.53	0.28	33.45	0.24	33.41	0.22
Tanks and Temples	24.17	2.9	24.38	2.6	24.28	1.3
Deep Blending	28.04	2.8	28.36	1.6	28.52	1.0

Table 2. **Same model size comparisons.** We compare the PSNR performance between 3DGS* and our model using alpha-only and RGBA textures with the same model size. The average number of optimized Gaussians (in millions) from each dataset is shown in the #GS (M) columns. We boldface and underline the best and second-best performing models, respectively. Alpha-only texture models generally achieve better performance than 3DGS* and RGBA texture models with the same model size.

with fewer Gaussians, achieving nearly 2 dB improvements over 3DGS* when using 1% of the default number of Gaussians. Figure 4 shows the trend of the quality of novel view synthesis as we vary the number of Gaussians. Our method consistently outperforms 3DGS* when using different numbers of Gaussians.

Table 2 compares the performance of 3DGS* and our Textured Gaussians model with the same model size using alpha-only textures and RGBA textures. Since our models use per-Gaussian texture maps, they require fewer Gaussians than 3DGS* to achieve the same model size. Our alpha-only textures model generally outperforms the 3DGS* and RGBA textures models, indicating that there is a sweet spot for distributing the model size budget between Gaussian parameters and texture map channels. This is further explored in Section 4.3.

4.2. Qualitative Results

Novel-View Synthesis. We show novel view synthesis results of our model with RGBA textures and 3DGS* on both standard benchmark and custom-captured datasets in Figures 5 and 6. Our method reconstructs much sharper details of the scene compared to 3DGS* when using the same small number of Gaussians. Please refer to the Supplemental Materials for results of our models optimized using varying number of Gaussians.

Color Component Decomposition. We show the *alpha-modulated and composited* results of the two color components, $\mathbf{c}^{\text{tex}}$ and $\mathbf{c}^{\text{base}}$, of the optimized alpha-only, RGB and RGBA textured Gaussians model with 1% of the default number of Gaussians in Figure 7. A 3DGS* model with the same number of Gaussians is shown for comparison. For RGB and RGBA models, $\mathbf{c}^{\text{tex}}$ reconstructs fine-grained textures that 3DGS* cannot. For alpha-only and RGBA models, $\mathbf{c}^{\text{base}}$ also reconstructs high-frequency details due to spatially varying alpha composition enabled by alpha textures.

4.3. Ablation Studies

Texture Map Variants. We ablate variants of our model that use different texture maps, namely alpha-only, RGB, and RGBA texture maps, and the same number of Gaussians in Figure 8. Experiments are conducted on Blender, Tanks and Temples, and Deep Blending datasets with the default optimized number (top row) and 1% of the default optimized number (bottom row) of Gaussians. We see that our model achieves the best performance with RGBA textures. Interestingly, using alpha-only textures already outperforms 3DGS*, and our RGB textures model despite being one-third the size, striking a better balance between performance and model size. This is because alpha-textured Gaussians can represent complex shapes through spatially varying opacity and reconstruct high-frequency textures through spatially varying alpha composition. In contrast, RGB-textured Gaussians are still limited to representing ellipsoids.

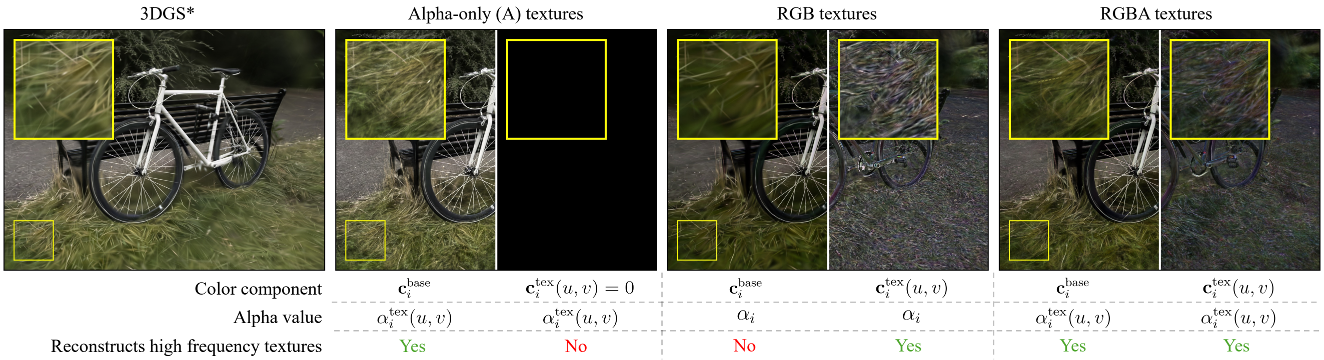

Figure 7. **Color component decomposition visualization.** We show the *alpha-modulated and composited* $\mathbf{c}^{\text{base}}$ and $\mathbf{c}^{\text{tex}}$ color components in the Textured Gaussians appearance model, as described in Eq. 7, 8, and 9. Zoom-in crops of the bottom-left of the figures are shown as insets with yellow borders. The brightness of the texture color component is adjusted for better visualization. While 3DGS* fails to restore fine details with a small number of Gaussians, our Textured Gaussians successfully reconstruct sharp textures. With RGB textures, the base color component reconstructs the low-frequency textures in the scene, while the texture map color component reconstructs the high-frequency textures. With alpha-only and RGBA textures, the base color component can also reconstruct high-frequency textures due to spatially varying alpha compositing.

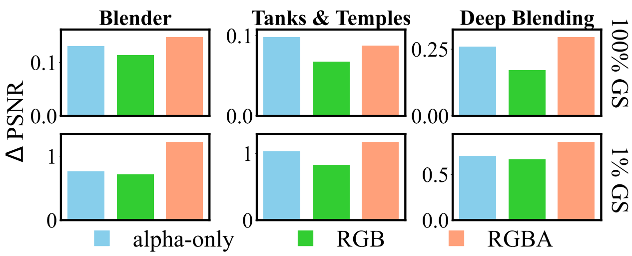

Figure 8. **Ablation study on texture map variants.** We show the PSNR performance improvements (ΔPSNR) over 3DGS* of our Textured Gaussians model when using different texture map variants. We see that in general, models with full RGBA textures achieve the best results. Interestingly, the alpha-only textures model achieves better quality than the RGB textures model and performs comparably to the RGBA textures model while using one-third and one-fourth of the model size, respectively.

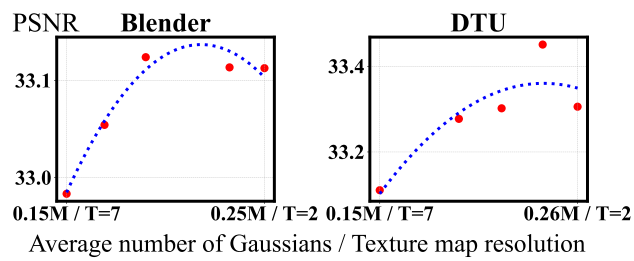

Figure 9. **Ablation study on texture map resolution and the number of Gaussians given the same model size.** We optimized different alpha-only Textured Gaussians models (marked as red dots) of the same model size with different texture map resolutions and number of Gaussians. A quadratic curve (blue dashed line) is fitted to the red dots to illustrate the trend of the performance. The best-performing model variant strikes a balance between the two values to achieve the best novel view synthesis results.

Texture Map Resolution and the Number of Gaussians. In Figure 9, we optimize alpha-only Textured Gaussians models of the same model size with different texture map resolutions and the number of Gaussians, and show the trend of novel view synthesis performance on the Blender and DTU datasets. We observe that there is a sweet spot between texture map resolution and the number of Gaussians that achieves the best image reconstruction quality. Maximizing texture map resolution corresponds to greatly reducing the number of Gaussians, which is harmful for reconstructing detailed geometric structures. Minimizing the resolution of the texture map, on the other hand, simply degenerates into 3DGS*.

5. Discussions

Limitations. The use of 2D diffuse texture maps in our model assumes that all textures lie on a local surface and does not model spatially varying specular color. Thus, extending our texture representation to represent local 3D volume textures or even 5D radiance fields is crucial. Furthermore, using factorized representations such as TensoRF [7] or triplane [5] to represent these high-dimensional textures could also be an interesting research problem.

Conclusions. In this paper, we augment 3DGS with texture maps to allow individual Gaussians to model spatially varying colors and opacity. As such, each Gaussian can represent a much richer set of appearances and shapes. This greatly improves the expressivity of individual Gaussians, leading to a better novel-view synthesis quality. Our method achieves better quality than 3DGS when using the same number of Gaussians, and achieves better or comparable quality when using the same model size.

References

[1] K. Aliev, A. Sevastopolsky, M. Kolos, D. Ulyanov, and V. Lempitsky. Neural Point-Based Graphics, 2019. arXiv preprint. 2

[2] Milena T. Bagdasarian, Paul Knoll, Yi-Hsin Li, Florian Barthel, Anna Hilsmann, Peter Eisert, and Wieland Morgenstern. 3DGS.zip: A survey on 3D Gaussian Splatting Compression Methods, 2024. arXiv preprint. 3

[3] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *CVPR*, 2022. 5

[4] Matthew Brown and David G Lowe. Unsupervised 3D Object Recognition and Reconstruction in Unordered Datasets. In *3DIM*, 2005. 2

[5] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient Geometry-Aware 3D Generative Adversarial Networks. In *CVPR*, 2022. 8

[6] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth Synthesis and Local Warps for Plausible Image-Based Navigation. *ACM TOG*, 32 (3), 2013. 2

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial Radiance Fields. In *ECCV*, 2022. 8

[8] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. GaussianEditor: Swift and Controllable 3D Editing with Gaussian Splatting. In *CVPR*, 2024. 1

[9] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC: Hash-Grid Assisted Context for 3D Gaussian Splatting Compression. In *ECCV*, 2024. 3

[10] Chong Bao and Bangbang Yang, Zeng Junyi, Bao Hujun, Zhang Yinda, Cui Zhaopeng, and Zhang Guofeng. NeuMesh: Learning Disentangled Neural Mesh-Based Implicit Field for Geometry and Texture Editing. In *ECCV*, 2022. 2

[11] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson De Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating Textures. In *CGF*, 2008. 2

[12] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. In *NeurIPS*, 2024. 3

[13] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. Multi-View Stereo for Community Photo Collections. In *ICCV*, 2007. 2

[14] Antoine Guédon and Vincent Lepetit. SuGaR: Surface-Aligned Gaussian Splatting for Efficient 3D Mesh Reconstruction and High-Quality Mesh Rendering. In *CVPR*, 2024. 1, 2

[15] Antoine Guédon and Vincent Lepetit. Gaussian Frosting: Editable Complex Radiance Fields with Real-Time Rendering. In *ECCV*, 2024. 1, 2

[16] Yuan-Chen Guo, Yan-Pei Cao, Chen Wang, Yu He, Ying Shan, and Song-Hai Zhang. VMesh: Hybrid Volume-Mesh Representation for Efficient View Synthesis. In *SIGGRAPH Asia*, 2023. 2

[17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004. 2

[18] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep Blending for Free-Viewpoint Image-Based Rendering. *ACM TOG*, 37(6), 2018. 2, 5

[19] D. Hu, Z. Zhang, T. Hou, T. Liu, H. Fu, and M. Gong. Multiscale Representation for Real-Time Anti-Aliasing Neural Rendering. In *ICCV*, 2023. 2

[20] Ronghang Hu, Nikhila Ravi, Alexander C. Berg, and Deepak Pathak. Worldsheet: Wrapping the World in a 3D Sheet for View Synthesis from a Single Image. In *ICCV*, 2021. 2

[21] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH*, 2024. 1, 2

[22] Zhentao Huang and Minglun Gong. Textured-GS: Gaussian Splatting with Spatially Defined Color and Opacity, 2024. arXiv preprint. 2, 3

[23] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engil Tola, and Henrik Aanæs. Large Scale Multi-View Stereopsis Evaluation. In *CVPR*, 2014. 5

[24] H. Kato, Y. Ushiku, and T. Harada. Neural 3D Mesh Renderer. In *CVPR*, 2018. 2

[25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM TOG*, 42(4), 2023. 1, 2, 3, 5, 6

[26] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM TOG*, 36(4), 2017. 5

[27] Georgios Kopanas, Julien Philip, Thomas Leimkühler, and George Drettakis. Point-Based Neural Rendering with Per-View Optimization. *CGF*, 40(4), 2021. 2

[28] Jonas Kulhanek and Torsten Sattler. NerfBaselines: Consistent and Reproducible Evaluation of Novel View Synthesis Methods, 2024. arXiv preprint. 5

[29] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian Representation for Radiance Field. In *CVPR*, 2024. 3

[30] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-Fidelity Neural Surface Reconstruction. In *CVPR*, 2023. 1

[31] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. Deep Appearance Models for Face Rendering. *ACM TOG*, 37(4), 2018. 2

[32] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering. In *CVPR*, 2024. 3

[33] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020. 2, 5

[34] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3D Scene Representation via Self-Organizing Gaussian Grids. In *ECCV*, 2024. 3

[35] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM TOG*, 41(4), 2022. 2, 5

[36] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compgs: Smaller and faster gaussian splatting with vector quantization. In *ECCV*, 2024. 3

[37] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis. In *CVPR*, pages 10349–10358, 2024.

[38] Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. Reducing the Memory Footprint of 3D Gaussian Splatting. *ACM CGIT*, 7(1), 2024. 3

[39] Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3DGS-Avatar: Animatable Avatars via Deformable 3D Gaussian Splatting. In *CVPR*, 2024. 2

[40] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. DreamGaussian4D: Generative 4D Gaussian Splatting, 2023. arXiv preprint. 1

[41] Victor Rong, Jingxiang Chen, Sherwin Bahmani, Kiriakos N Kutulakos, and David B Lindell. GStex: Per-Primitive Texturing of 2D Gaussian Splatting for Decoupled Appearance and Geometry Modeling, 2024. 3

[42] Antoni Rosinol, John J Leonard, and Luca Carlone. NeRF-SLAM: Real-time Dense Monocular Slam with Neural Radiance Fields. In *IROS*, 2023. 1

[43] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance Fields without Neural Networks. In *CVPR*, 2022. 2

[44] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *CVPR*, 2006. 2

[45] Zhijing Shao, Zhaolong Wang, Zhuang Li, Duotun Wang, Xiangru Lin, Yu Zhang, Mingming Fan, and Zeyu Wang. SplattingAvatar: Realistic Real-Time Human Avatars with Mesh-Embedded Gaussian Splatting. In *CVPR*, 2024. 2

[46] Meng-Li Shih, Jia-Bin Huang, Changil Kim, Rajvi Shah, Johannes Kopf, and Chen Gao. Modeling Ambient Scene Dynamics for Free-View Synthesis. In *SIGGRAPH*, 2024. 1

[47] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. In *NeurIPS*, 2019. 2

[48] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo Tourism: Exploring Photo Collections in 3D. *ACM TOG*, 25 (3), 2006. 2

[49] Pratul P. Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T. Barron. NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis. In *CVPR*, 2021. 1

[50] Shih-Yang Su, Frank Yu, Michael Zollhöfer, and Helge Rhodin. A-NeRF: Articulated Neural Radiance Fields for Learning Human Shape, Appearance, and Pose. In *NeurIPS*, 2021. 1

[51] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation. In *ICLR*, 2024. 1

[52] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in Neural Rendering. *CGF*, 41(2), 2022. 1, 2

[53] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. In *NeurIPS*, 2021. 1

[54] Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video. In *CVPR*, 2022. 1

[55] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-Nerf: Point-Based Neural Radiance Fields. In *CVPR*, 2022. 2

[56] Tian-Xing Xu, Wenbo Hu, Yu-Kun Lai, Ying Shan, and Song-Hai Zhang. Texture-GS: Disentangling the Geometry and Texture for 3D Gaussian Splatting Editing. In *ECCV*, 2024. 2, 3, 5

[57] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3D Gaussians for High-Fidelity Monocular Dynamic Scene Reconstruction. In *CVPR*, 2024. 1

[58] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian Grouping: Segment and Edit Anything in 3D Scenes. In *ECCV*, 2024. 2

[59] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details in 3d gaussian splatting. In *ACM-MM*, 2024. 6

[60] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for Real-Time Rendering of Neural Radiance Fields. In *ICCV*, 2021. 2

[61] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes. *ACM TOG*, 2024. 6

[62] Xiuming Zhang, Pratul P. Srinivasan, Boyang Deng, Paul Debevec, William T. Freeman, and Jonathan T. Barron. NeRFactor: Neural Factorization of Shape and Reflectance under an Unknown Illumination. *ACM TOG*, 40(6), 2021. 1

[63] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. NICE-SLAM: Neural Implicit Scalable Encoding for SLAM. In *CVPR*, 2022. 1