Texture-GS: Disentangling the Geometry and Texture for 3D Gaussian Splatting Editing

Tian-Xing Xu[1], Wenbo Hu[2†], Yu-Kun Lai[3], Ying Shan[2], and Song-Hai Zhang[1†]

[1] Tsinghua University, China
xutx21@mails.tsinghua.edu.cn,shz@tsinghua.edu.cn
[2] Tencent AI Lab, China
wbhu@tencent.com,yingsshan@tencent.com
[3] Cardiff University, United Kingdom
LaiY4@cardiff.ac.uk

Abstract. 3D Gaussian splatting, emerging as a groundbreaking approach, has drawn increasing attention for its capabilities of high-fidelity reconstruction and real-time rendering. However, it couples the appearance and geometry of the scene within the Gaussian attributes, which hinders the flexibility of editing operations, such as texture swapping. To address this issue, we propose a novel approach, namely Texture-GS, to disentangle the appearance from the geometry by representing it as a 2D texture mapped onto the 3D surface, thereby facilitating appearance editing. Technically, the disentanglement is achieved by our proposed texture mapping module, which consists of a UV mapping MLP to learn the UV coordinates for the 3D Gaussian centers, a local Taylor expansion of the MLP to efficiently approximate the UV coordinates for the ray-Gaussian intersections, and a learnable texture to capture the fine-grained appearance. Extensive experiments on the DTU dataset demonstrate that our method not only facilitates high-fidelity appearance editing but also achieves real-time rendering on consumer-level devices, *e.g.* a single RTX 2080 Ti GPU.

Keywords: Neural rendering · Scene editing · Novel view synthesis · Gaussian splatting · Texture mapping · Disentanglement

1 Introduction

Reconstruction, editing, and real-time rendering of photo-realistic scenes are fundamental problems in computer vision and graphics, with diverse applications such as film production, computer games, and virtual/augmented reality. Polygonal meshes have served as the standard 3D representation within traditional rendering pipelines, owing to their rendering speed and editing flexibility (with texture mapping).

Due to the laborious process of manual mesh-based scene modeling, 3D Gaussian Splatting [14] (3D-GS) has gained considerable attention for its capability

† Corresponding authors.

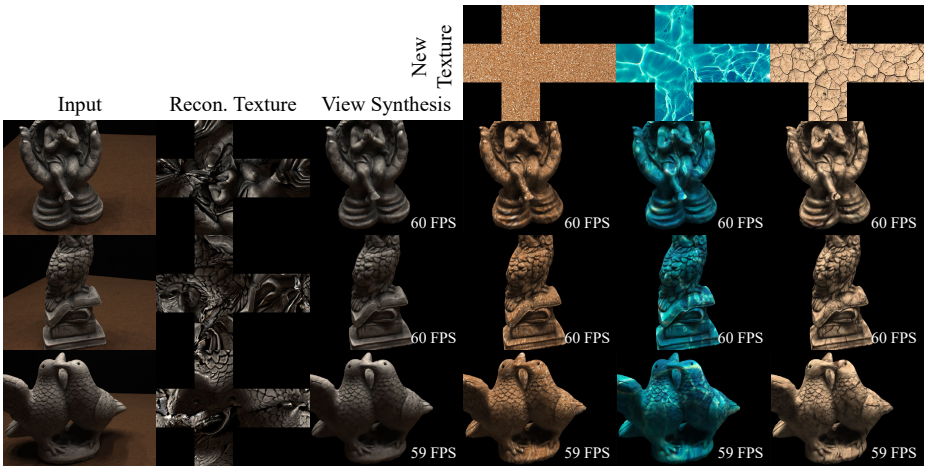


Fig. 1: Texture swapping with our method. We propose to disentangle the appearance from the geometry for 3D-GS, thereby facilitating real-time appearance editing such as texture swapping. The rendering speed is shown in each result.

of faithfully reconstructing complex scenes from multi-view images and real-time rendering. 3D-GS represents the scene as a set of 3D anisotropic Gaussians equipped with per-Gaussian color attributes and supports real-time rendering by splatting these Gaussians onto the image plane. However, this representation couples the appearance and geometry of the scene within the unordered and irregular 3D Gaussians, which hinders the flexibility of appearance editing for 3D-GS compared to meshes, where appearance can be easily parameterized into texture maps. Although considerable efforts have been made to edit 3D Gaussian-based scenes [2, 8, 15, 24, 28], the manipulation of appearance remains inconvenient since these works still follow the entangled representation of 3D-GS.

In this paper, we propose a novel method, named Texture-GS, which aims to explicitly disentangle the geometry and texture for 3D-GS, thereby significantly improving the flexibility of appearance editing for 3D scenes. Texture-GS follows 3D-GS in modeling the geometry as a set of anisotropic 3D Gaussians, but crucially, it represents the view-independent appearance as a 2D texture map. Leveraging this disentangled representation, Texture-GS retains the powerful capabilities of 3D-GS for faithful reconstruction and real-time rendering, while also facilitating various appearance editing applications, such as texture swapping shown in Fig. 1.

The key challenge in implementing our Texture-GS lies in establishing a connection between the geometry (3D Gaussians) and appearance (2D texture map). NeuTex [23] has proposed a texture mapping MLP (Multi-Layer Perceptron) that regresses 2D UV coordinates for every 3D point to represent the radiance of NeRF (Neural Radiance Field) [17] in a 2D texture space. However, evaluating an MLP for each ray-Gaussian intersection is unsuitable for our Texture-GS, as

it would be prohibitively expensive for real-time rendering, which is a key advantage of 3D-GS over NeRF-based methods. To maintain the ability of real-time rendering, one straightforward solution is to employ a texture mapping MLP to pre-compute UV coordinates for each Gaussian based on its center location and then query the *per-Gaussian* color attributes from the texture map before rendering. Given the fact that each 3D Gaussian often covers more than one pixel in practice, this straightforward solution would result in all pixels covered by a single Gaussian being mapped to the same UV location, leading to discontinuities in the texture space. To address this issue, we propose a novel *texture mapping module*. It incorporates an MLP for mapping Gaussian centers into the texture UV space before rendering, along with a Taylor expansion at the Gaussian centers, which serves as an approximation of the MLP and enables efficient mapping of the ray-Gaussian intersections to UV coordinates during rendering. Our texture mapping module not only promotes a smooth texture map, as the Taylor expansion guarantees the local continuity for the UV coordinates of pixels within a projected Gaussian, but also preserves rendering efficiency, as the inference of UV coordinates merely involves a small matrix product.

To evaluate the effectiveness of our Texture-GS, we conduct extensive quantitative and qualitative experiments on the DTU dataset [1]. The results demonstrate that Texture-GS recovers a smooth high-quality 2D texture map from multi-view images, thereby facilitating various editing applications such as global texture swapping and fine-grained texture editing. Besides, our method achieves an average rendering speed of 58 FPS on a single RTX 2080 Ti GPU, demonstrating its real-time rendering capabilities.

Our contributions are summarized below.

- To the best of our knowledge, we are the first to disentangle the geometry and texture for 3D-GS, thereby enabling various editing applications.
- We propose a novel texture mapping module to map ray-Gaussian intersections into a continuous 2D texture space while maintaining efficiency.
- Experiments validate the effectiveness of our method for novel view synthesis, global texture swapping, and local appearance editing, with real-time rendering speed on consumer-level devices.

2 Related Work

Neural UV Mapping. UV mapping plays an essential role in the traditional rendering pipeline, aiming at computing a bijective mapping between the 3D surface and a suitable parametric domain. UV mapping is usually accompanied by 3D shapes and jointly modeled by artists, necessitating considerable labor costs. In recent years, NeRF [17] has gained increasing attention for its superior view synthesis quality, inspiring many follow-up works [5,23,30] to reconstruct 3D geometry with a volumetric density field while concurrently learning UV mapping based on neural networks. NeuTex [23] is the first work to recover a meaningful surface-aware UV mapping function from multi-view images. Provided with any 3D shading point during the NeRF's ray marching process, NeuTex obtains its

radiance by sampling the reconstructed texture value at its UV location output by a texture mapping MLP. To ensure the bijective property of UV mapping, NeuTex adopts a cycle consistency loss to regularize the network. The follow-up Neural Gauge Field [30] draws inspiration from the principle of information conservation during gauge transformation [18] and proposes an information regularization term to maximize the mutual information. Nuvo [21] extends NeuTex with multiple charts and a chart assignment network for general scenes. Apart from the above methods, some approaches focus on specific object categories such as human faces [4, 16, 25], documents [5] and human bodies [3]. However, NeRF-based methods adopt ray marching for rendering, which evaluates a large texture MLP with an additional UV mapping network at hundreds of sample shading points along the ray for each pixel to compute the final color. This process is excessively slow for interactive visualization and real-time applications.

3D Gaussian Editing. 3D Gaussian Splatting [14] (3D-GS) has emerged as an alternative 3D representation to NeRF [17], achieving real-time rendering via splatting 3D Gaussians instead of ray marching. It has received increasing attention for its explicit representation and promising reconstruction quality, which is more suitable for scene editing. Leveraging its explicit, point cloud-like formulation, Point'n Move [11], Gaussian Grouping [28], SA-GS [10] and Feature 3DGS [31] combine semantic segmentation methods such as SAM [15] with 3D GS representation to obtain the mask of the target and explicitly manipulate the selected object in the scene in real-time. SC-GS [12] and Cogs [29] introduce novel frameworks for manipulating and editing dynamic content in 4D space. With the advancement of text-to-image models [19], some works [2, 8] achieve swift and controllable 3D scene editing under text instructions, incorporating a 2D diffusion model to fine-tune 3D-GS representations. PhysGaussian [24] explores the physical properties of 3D Gaussians, employing a custom Material Point Method for physical simulation. Despite yielding promising results, these methods capture the appearance in per-Gaussian color attributes and regard 3D Gaussians as isolated shading elements, neglecting the global appearance. The entanglement of color and density attributes hinders editing flexibility and precludes editing applications such as texture swapping.

3 Preliminaries

3D-GS [14] represents the scene as a set of 3D Gaussians $\mathcal{G} = \{G_i(x)\}_{i=1}^{N}$, where N denotes the number of Gaussians. Each Gaussian is defined by its center position $\mu_i \in \mathbb{R}^3$ and covariance matrix $\Sigma_i \in \mathbb{R}^{3\times3}$, expressed as:

$$G_i(x) = \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma_i^{-1}(x-\mu)\right). \tag{1}$$

The appearance of the scene is represented in the per-Gaussian attributes, *i.e.* an opacity value $o_i \in \mathbb{R}$ to adjust the influence weight and an RGB color $c_i \in \mathbb{R}^3$ given by spherical harmonic (SH) coefficients. To render the scene, 3D-GS splats

3D Gaussians onto the image plane via EWA splatting [32] to form 2D Gaussians G'_i, whose covariance matrix $\Sigma'_i \in \mathbb{R}^{2\times2}$ is defined as $\Sigma'_i = JW\Sigma_i W^T J^T$. Here $W \in \mathbb{R}^{3\times3}$ denotes the viewing transformation matrix and $J \in \mathbb{R}^{2\times3}$ is the Jacobian matrix of the affine approximation of the perspective projection. The final color C_p at pixel p is given by a set of ordered Gaussians with cumulative volumetric rendering:

$$C_p = \sum_{j\in\mathcal{N}_p} c_j \alpha_j \prod_{k=1}^{j-1}(1-\alpha_k), \tag{2}$$

Here $\mathcal{N}_p$ is the number of projected Gaussians within a tile, and $\alpha_j = o_j G'_j(p)$ is the transparency value. Notably, the geometry (the positions and scales of 3D Gaussians) and appearance (per-Gaussian colors) are entangled in the 3D-GS representation, complicating downstream appearance editing applications such as texture swapping.

4 Method

Given the multi-view images of a real-world scene, the objective of our Texture-GS is to reconstruct the geometry using a set of 3D Gaussians and the appearance through our proposed *texture mapping module*, in a disentangled manner, enabling flexible texture editing applications. Our texture mapping module consists of a UV mapping MLP that projects 3D points into 2D UV space, and a learnable 2D texture representing the appearance of the 3D scene. It is highly under-constrained to jointly learn the UV mapping MLP and the texture values. Therefore, we first learn the UV mapping MLP using a cycle consistent constraint on the bijective mapping between the 3D space and the UV texture domain (in Sec. 4.1) and then learn the 2D texture values using a photometric loss and Taylor-expansion-based approximation of the MLP for efficiency (in Sec. 4.2).

4.1 Learning UV Mapping MLP

The UV mapping MLP, denoted as $\phi : \mathbb{R}^3 \to \mathbb{R}^2$, aims to regress 2D UV coordinates $u \in \mathbb{R}^2$ from the 3D point $x \in \mathbb{R}^3$ that lies either on or close to the underlying surface of the scene. Following NeuTex [23], we define the texture space as a unit sphere and use cubemaps to visualize them. It is non-trivial to learn the highly under-constrained mapping ϕ without paired training data. Intuitively, we observe that if its inverse 2D-to-3D function, $\phi^{-1} : \mathbb{R}^2 \to \mathbb{R}^3$, can warp the UV space to uniformly cover the underlying surface, then the forward mapping ϕ would serve as a reasonable UV mapping.

Based on this observation, we should first estimate the underlying surface of the object from multi-view images, before the UV mapping learning. To this

end, we train a vanilla 3D-GS and leverage the alpha-blending aggregation in Eq. (2) to obtain the depth value D_p for pixel p:

$$D_p = \sum_{j \in \mathcal{N}_p} d_j \alpha_j \prod_{k=1}^{j-1} (1 - \alpha_k), \tag{3}$$

where $d_j = (R\mu_j + t)_z$ denotes the depth value of the center μ_j in the camera coordinate system. Given the assumption that the editing target has an opaque and smooth surface, the back-projected 3D points $\{x_i\}_{i=1}^{N_d}$ from these depth maps would lie either on or close to the underlying surface.

With these 3D points approximating the underlying surface, we can establish the following constraints to learn the UV mapping ϕ. (1) We introduce a cycle consistency loss $\mathcal{L}_{\text{cycle}}^{\text{3d}}$ in the 3D space to ensure the mutually inverse property of ϕ and ϕ^{-1}:

$$\mathcal{L}_{\text{cycle}}^{\text{3d}} = \frac{1}{N_d} \sum_{i=1}^{N_d} ||x_i - \phi^{-1} \circ \phi(x_i)||. \tag{4}$$

(2) To encourage ϕ^{-1} to wrap the underlying surface with a uniform 2D UV plane, we randomly sample a set of evenly distributed UV coordinates $\mathcal{U} = \{u_i\}_{i=1}^{N_u}$, and then minimize the symmetric Chamfer distance between the 3D points mapped with ϕ^{-1} from $\mathcal{U}$ and the pseudo ground truth point cloud $\mathcal{P} = \{p_i\}_{i=1}^{N_p}$ extracted from the trained 3D-GS via the farthest point sampling on the centers of the 3D Gaussians:

$$\mathcal{L}_{\text{CD}} = \frac{1}{N_u} \sum_{i=1}^{N_u} \min_{p_j \in \mathcal{P}} ||\phi^{-1}(u_i) - p_j|| + \frac{1}{N_p} \sum_{j=1}^{N_p} \min_{u_i \in \mathcal{U}} ||\phi^{-1}(u_i) - p_j||. \tag{5}$$

(3) We also leverage the sampled UVs $\mathcal{U}$ to regularize the bijectivity via the cycle consistency loss $\mathcal{L}_{\text{cycle}}^{\text{2d}}$ in the 2D space:

$$\mathcal{L}_{\text{cycle}}^{\text{2d}} = \frac{1}{N_u} \sum_{i=1}^{N_u} ||u_i - \phi \circ \phi^{-1}(u_i)||. \tag{6}$$

These three constraints are combined together to learn the UV mapping MLP:

$$\mathcal{L}_{\text{UV}} = \mathcal{L}_{\text{cycle}}^{\text{3d}} + \mathcal{L}_{\text{CD}} + \mathcal{L}_{\text{cycle}}^{\text{2d}}. \tag{7}$$

Although NeuTex [23] also utilizes the cycle consistency constraint for learning the UV mapping, it computes the loss term with each sampled 3D shading point on the rays. In contrast, our method only applies constraints to 3D points near the underlying surface, which results in higher efficiency for training.

4.2 Learning Texture Value

Having the UV mapping MLP ϕ, our goal is to reconstruct the high-quality texture from multi-view images through differentiable rendering, enabling flexible

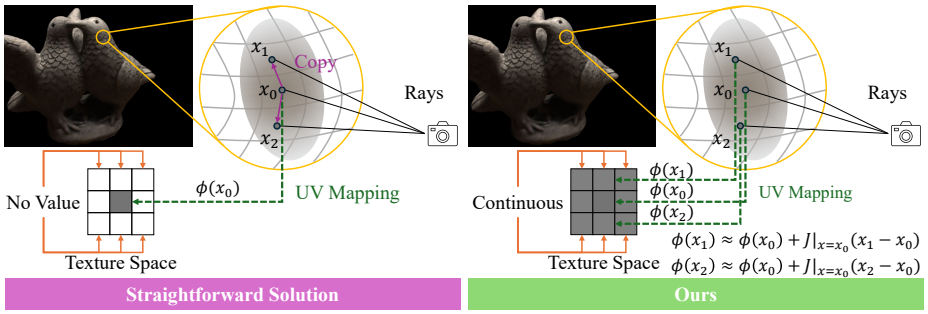


Fig. 2: Comparison with the straightforward solution. The straightforward solution fails to generate a reasonable and continuous texture, while our method can reconstruct a high-quality texture by considering the intersection of 3D Gaussians and each ray.

scene appearance editing, *e.g.* texture swapping, and real-time rendering. It is not practical to predict UV coordinates for each 3D point using ϕ, akin to the NeuTex [23], as evaluating an MLP for each ray-Gaussian intersection consumes too much computational cost for real-time rendering. To this end, a straightforward solution is to apply the UV mapping MLP ϕ to each Gaussian's center to pre-compute UV coordinates, then query the *per-Gaussian* color attributes from the learnable texture before rendering, and finally optimize the texture value through differentiable Gaussian splitting. However, since each 3D Gaussian often covers multiple pixels in practice, this solution would map all pixels covered by a single Gaussian to the same UV location (Fig. 2, left), resulting in degenerated textures. This would significantly hinder appearance editing.

To address this issue, we propose to allow pixels covered by a single Gaussian to have different UV coordinates (Fig. 2, right), which encourages a smooth texture for the convenience of appearance editing. We rewrite the rendering equation in Eq. (2) as:

$$C_p = \sum_{j \in \mathcal{N}_p} \mathcal{C}(G_j, r_p)\alpha_j \prod_{k=1}^{j-1}(1 - \alpha_k). \tag{8}$$

Here, we replace original per-Gaussian color attributes with a color function $\mathcal{C}$, which takes both the 3D Gaussian G_j and the ray $r_p = o + td_p$ along the pixel p as input, where $o \in \mathbb{R}^3$ is the viewpoint location and $d_p \in \mathbb{R}^3$ is a unit view direction vector.

Ray-Gaussian Intersection. To establish the color function $\mathcal{C}$, we initially compute the intersection $I(G_j, r_p)$ between a ray r_p and a 3D Gaussian G_j. However, the vanilla 3D Gaussians denote a volume density function without solid surfaces, hindering the derivation of ray-Gaussian intersection. Fortunately, in most practical cases, the Gaussian should be flat and aligned closely with the opaque surface, as mentioned in previous work [9, 13]. To achieve this, we

introduce a zero-one regularization term to the opacity value o_i

$$\mathcal{L}_{01} = \frac{1}{N}\sum_{i=1}^{N}(\ln(o_i) + \ln(1-o_i)), \tag{9}$$

and prune those Gaussians with an opacity that is less than 0.5. Subsequently, we flatten the 3D Gaussian along its normal vector n_j, which is defined as the eigenvector v_j associated with the smallest eigenvalue of the covariance matrix Σ_j. To ensure the normal n_j points outwards from the surface, we selectively reverse v_j based on the view direction $\mu_j - o$:

$$n_j = \begin{cases} v_j, & \text{if} \quad v_j \cdot (\mu_j - o) < 0, \\ -v_j, & \text{otherwise.} \end{cases} \tag{10}$$

Hence, the intersection between the flattened Gaussian and the ray is given by:

$$I(G_j, r_p) = o + \frac{(\mu_j - o) \cdot n_j}{d_p \cdot n_j} d_p. \tag{11}$$

As this derivation relies heavily on the normal vector n_j of 3D Gaussian, we additionally introduce a normal loss to supervise the rendered normal map $\overline{N}$ with the pseudo ground truth $\overline{N}_{\text{gt}}$ derived from SfM [20] result under the local planarity assumption:

$$\mathcal{L}_{\text{norm}} = \frac{1}{HW}||\overline{N} - \overline{N}_{\text{gt}}||_2^2. \tag{12}$$

To suppress high-frequency noise in $\overline{N}_{\text{gt}}$, we also apply a spatial smoothness regularization term:

$$\mathcal{L}_{\text{sm}} = \frac{1}{HW}\sum_{p}\sum_{q\in\mathcal{N}(p)} \exp(-\gamma||C_{\text{gt}}(p) - C_{\text{gt}}(q)||_1)||\overline{N}(p) - \overline{N}(q)||_1. \tag{13}$$

Besides, to avoid the intersection far away from the center of 3D Gaussian, we clamp the intersection with a radius defined by the eigenvalue of the matrix Σ_i.

Efficient UV Mapping. Due to the high computational cost of evaluating the MLP ϕ for each ray-Gaussian intersection, as shown in the right part of Fig. 2, we propose to approximate the UV coordinates $\phi(I(G_j, r_p))$ utilizing the first two terms of the Taylor expansion (i.e., first-order approximation) of ϕ at each Gaussian center μ_j, written as

$$\tilde{\phi}(I(G_j, r_p)) = \phi(\mu_j) + J|_{x=\mu_j}(I(G_j, r_p) - \mu_j), \tag{14}$$

where $J|_{x=\mu_j}$ denotes the Jacobian matrix of ϕ at the point μ_j. The Jacobian matrix can be efficiently obtained with automatic differentiation and pre-computed for each Gaussian center before rendering. Thus, Eq. (14) solely involves a small matrix product, significantly reducing the computational cost during rendering, and thereby enabling real-time rendering for downstream applications.

Color Function. Leveraging the UV coordinates for each intersection, we can directly query the texture $\mathcal{T}$ to establish the color function $\mathcal{C}$. Notably, view-dependent appearance is common in real-world scenes, typically caused by non-Lambertian materials. 3D-GS [14] addresses this issue by representing the per-Gaussian color as an ordered set of spherical harmonic (SH) coefficients, where we interpret the first SH coefficient as the diffuse term and others as the view-dependent appearance. However, representing the textures with SH coefficients consumes high memory and computational costs, as the resolution of textures is usually much larger than the number of 3D Gaussians. Therefore, we propose to represent the diffuse color within the texture $\mathcal{T}$ and utilize per-Gaussian SH coefficients c_j^{SH} for the residual view-dependent appearance, written as:

$$\mathcal{C}(G_j, r_p) = h(\tilde{\phi}(I(G_j, r_p)), \mathcal{T}) + c_j^{\mathrm{SH}}, \tag{15}$$

where $h(\cdot, \mathcal{T})$ denotes the sampling on the texture $\mathcal{T}$ with UV coordinates using the bilinear interpolation.

The final loss for supervision is defined as:

$$\mathcal{L} = \mathcal{L}_{\mathrm{GS}} + \lambda(\mathcal{L}_1^{\mathrm{noSH}} + \lambda_{\mathrm{ssim}}\mathcal{L}_{\mathrm{ssim}}^{\mathrm{noSH}}), \tag{16}$$

$$\text{where} \quad \mathcal{L}_{\mathrm{GS}} = \mathcal{L}_1 + \mathcal{L}_{\mathrm{mask}} + \lambda_{\mathrm{ssim}}\mathcal{L}_{\mathrm{ssim}} + \lambda_{01}\mathcal{L}_{01} + \lambda_{\mathrm{n}}(\mathcal{L}_{\mathrm{norm}} + \mathcal{L}_{\mathrm{sm}}). \tag{17}$$

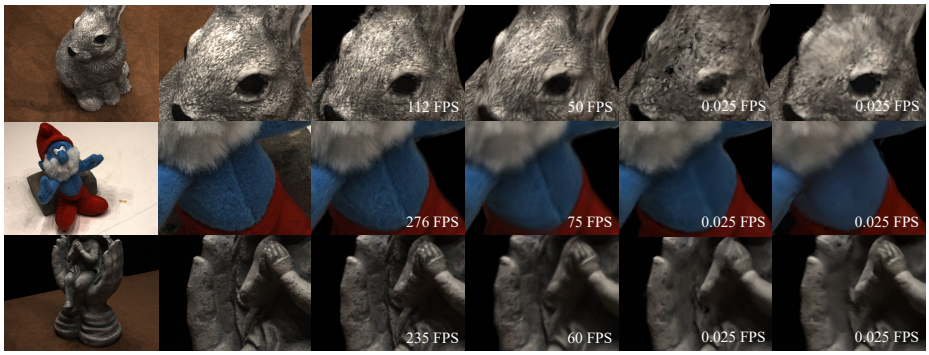
$\mathcal{L}_1$ and $\mathcal{L}_{\mathrm{ssim}}$ are the L1 and D-SSIM loss terms to regularize the visual quality of rendered images, as in 3D-GS [14]. $\mathcal{L}_{\mathrm{mask}}$ is the mask loss when viewpoints do not entirely cover the object, as in NeuTex [23]. $\mathcal{L}_1^{\mathrm{noSH}}$ and $\mathcal{L}_{\mathrm{ssim}}^{\mathrm{noSH}}$ are the L1 and D-SSIM loss terms for rendering without per-Gaussian SH coefficients to encourage most (especially view-independent) appearance information to be stored in the texture. Please refer to the supplementary material for more details.

5 Experiments

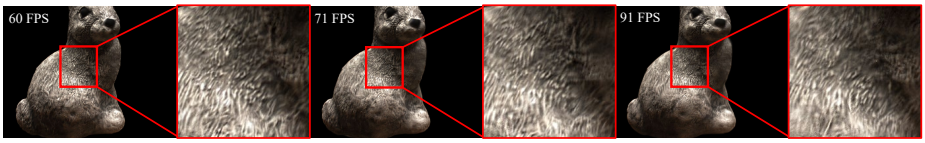
We conduct experiments on synthetic scenes from NeRF Synthetic [17], Google Scanned Objects [7], and Objaverse [6], as well as real scenes from DTU dataset [1] to validate the effectiveness and generalization ability of our model. We utilize the foreground masks provided by IDR [27] to segment the object. Following NeuTex [23], we sample 5 DTU scenes for quantitative experiments, withhold 4 views as test data, and train our method with the remaining images for novel view synthesis. We report three commonly adopted performance metrics, including Peak Signal-to-Noise Ratio (PSNR), L1 metric and Learned Perceptual Image Patch Similarity (LPIPS). All evaluation experiments are conducted on a single RTX 2080 Ti GPU.

5.1 Novel View Synthesis

We present comprehensive comparisons of our method with previous state-of-the-art (SOTA) editing methods that decouple the geometry and appearance,



(a) Comparison with previous state-of-the-art editing methods



(b) Under different numbers of 3D Gaussians

Fig. 3: Visualization of novel view synthesis results of our method

including NeRF-based NeuTex [23] and Neural Gauge Fields [30] (NGF) on novel view synthesis. Although NeuMesh [26] and Seal-3D [22] also support appearance editing applications, they require an extra finetuning stage (ranging from seconds to hours) to inject the edited appearance into the 3D representation, which is inconsistent with real-time editing objectives. Naturally, our method is more constrained than the vanilla 3D-GS [14], hence we also compare our method with 3D-GS to investigate the loss of rendering quality and computational efficiency.

Fig. 3a illustrates the visual comparison on zoom-in crops of test images. Although the rendering quality is slightly inferior to 3D-GS, our method can still synthesize photo-realistic rendered images. Compared with NeRF-based approaches, our method achieves superior performance, especially in regions observable from merely a few views of training data, such as the rabbit's head and the inner side of the palm. Thanks to the explicit geometry of 3D Gaussians, Texture-GS can generate higher-quality view synthesis results, while NeRF requires more views to learn the implicit representation. Most importantly, our method achieves real-time rendering speed, enabling instant preview of the editing results to facilitate interactive editing applications.

We present the quantitative comparison of the average metrics in Tab. 1a. Our method is 2000× faster than NeRF-based methods, maintaining a rendering quality that is on par and sometimes superior on the same hardware. The average training time of NeuTex [23] is 30 hours per scene, while ours amounts to 90 minutes.

Table 1: Comparison of novel view synthesis results on the DTU dataset.

(a) Comparison with the SOTAs

Method	DTU			
	PSNR↑	L1↓	LPIPS↓	FPS
NeuTex	30.39	0.0158	0.1613	0.025
NGF	29.44	0.0166	0.1506	0.025
3DGS	30.99	0.0121	0.1079	198
Ours	30.03	0.0135	0.1440	58

(b) Different number of 3D Gaussians

#Gauss	DTU			
	PSNR↑	L1↓	LPIPS↓	FPS
100%	30.03	0.0135	0.1440	58
50%	29.57	0.0142	0.1555	69
20%	28.75	0.0155	0.1705	82
5%	27.86	0.0172	0.1841	104

Texture-GS allows pixels covered by a single Gaussian to possess different colors, thereby enhancing the representational power for each Gaussian. To further accelerate the rendering speed, we simplify the geometry by pruning 50%, 80% and 95% 3D Gaussians according to their opacity values. As illustrated in Tab. 1b, reducing the number of Gaussians inevitably leads to a minor performance decline, while greatly reducing the computational costs. Fig. 3b shows that Texture-GS can synthesize high-quality images for objects with rich texture even with 4.4K 3D Gaussians representing the geometry, which demonstrates the potential extensibility to a wide range of computing platforms.

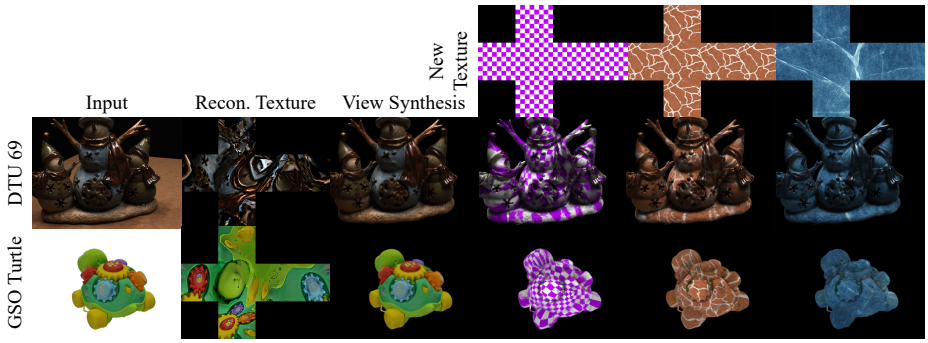
5.2 Texture Editing

Texture Swapping. Leveraging the decoupled representation, we can support real-time appearance editing such as changing the global texture of reconstructed objects. As shown in Fig. 4, our method can render photo-realistic and sharp images under various textures while maintaining the consistency of the geometry and appearance under different views. Even for the porcelain with complex geometry structure, our method can reconstruct a high-quality texture from multi-view images. We also present a visual comparison with NeuTex [23] in Fig. 5. Our method learns a more uniform texture space while achieving real-time rendering speed. More visual results can be seen in supplementary material.

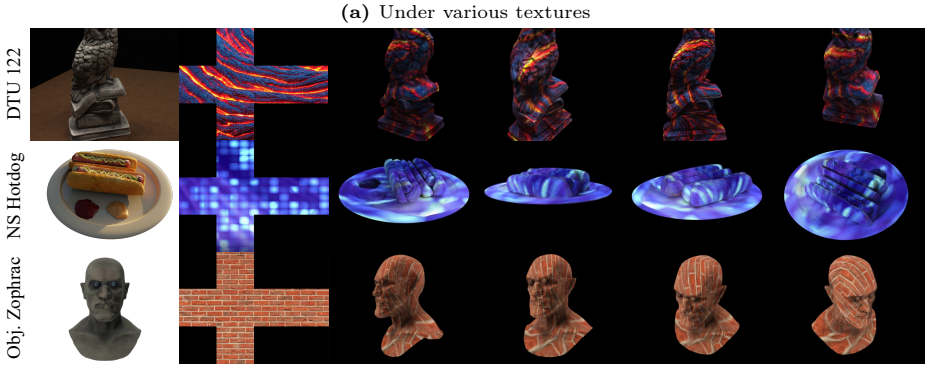
Texture Painting. We also investigate the capability of our method in fine-grained texture modification and show the visual results in Fig. 6. We conduct fine-grained editing on the reconstructed texture by adding the ECCV text on the leg of the stone statue and painting hearts on the wing. Notably, even for small patterns such as hearts, our method can yield consistent and photo-realistic rendering results across multiple views, demonstrating the practical utility of Texture-GS in practical 3D design workflows.

5.3 Ablation Study

We conduct ablation studies on the DTU dataset to evaluate the effectiveness of the different components and reports the quantitative results in Tab. 2. More can be seen in the supplementary material.



(a) Under various textures



(b) Under different views

Fig. 4: Visualization of texture swapping results of our method

Intersection-based UV Mapping. We compare with the straightforward solution (Pre-fetching), which pre-fetches per-Gaussian colors from the textures using the centers before rendering. As shown in Tab. 2 (left), since our method enhances the representation power of each Gaussian by enabling distinct colors for different pixels, our method achieves a consistent improvement of all metrics. We also provide a visual analysis in Fig. 7. Our method can reconstruct continuous and high-quality textures, whereas the texture space of pre-fetching is discontinuous due to the lack of supervision around the mapped centers in the UV space. The view synthesis results after changing the texture with a chessboard also demonstrate the importance of intersection-based UV mapping.

Per-Gaussian SH Coefficients. We also utilize the trained model to render test images without per-Gaussian term c_j^{SH} in Eq. (15). As shown in Tab. 2 (left), although eliminating the SH coefficients (Ours(no SH)) leads to a minor performance decline, our method can still yield photo-realistic view synthesis results. Fig. 7 shows the visual comparison on DTU scene 114, which contains a porcelain material statue. We observe that the per-Gaussian SH coefficients mainly capture the view-dependent appearance, such as darkening the belly region, while the 2D texture represents the detailed view-independent appearance.

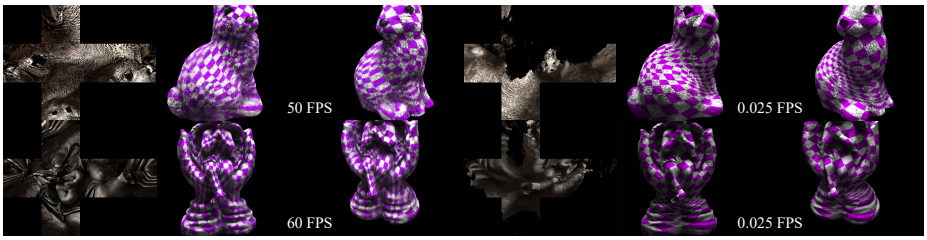


Fig. 5: Visual comparison of our method with NeuTex [23] on texture swapping.

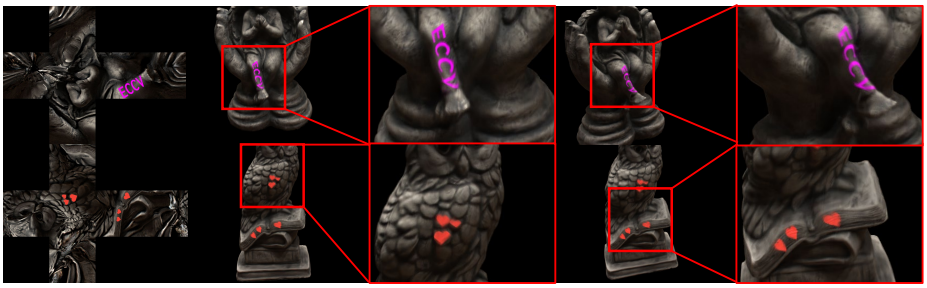


Fig. 6: Visualization of texture painting results of our method

Per-Gaussian SH v.s. SH-based Texture. An alternative approach to capturing view-dependent appearance is to represent the texture image with a grid of SH coefficients. Tab. 2 (right) illustrates the model complexity, rendering speed and visual quality of textures with different SH degrees on DTU scene 114. Textures with SH coefficients enhance the representation power to capture complex appearance variations under different views, albeit at the expense of substantial memory usage and computational cost. Conversely, our method applies SH coefficients with 3 degrees on the Gaussian attributes, thereby achieving high visual quality with negligible computational overhead and additional memory usage.

$\mathcal{L}_1^{\text{noSH}}$ and $\mathcal{L}_{\text{ssim}}^{\text{noSH}}$. To validate the effectiveness of no SH coefficients rendering regularization, we present comparison results in Tab. 2 (left). Although the

Table 2: Ablation studies of our method on the DTU dataset

Method	DTU		
	PSNR↑	L1↓	LPIPS↓
Ours	30.03	0.0135	0.1440
Ours(no SH)	27.63	0.0187	0.1566
w.o. Reg	30.62	0.0126	0.1374
w.o. Reg(no SH)	25.10	0.0264	0.1757
Pre-fetching	29.28	0.0149	0.1557
Taylor 2-order	31.10	0.0115	0.1339

SH Deg	DTU Scene 114			
	#Params	FPS	PSNR↑	L1↓
D=0	18.6 M	55	24.64	0.0305
D=1	72.6 M	31	24.79	0.0302
D=2	162.6 M	14	25.43	0.0280
D=3	288.6 M	7	27.88	0.0188
Ours	20.8 M	47	27.80	0.0189

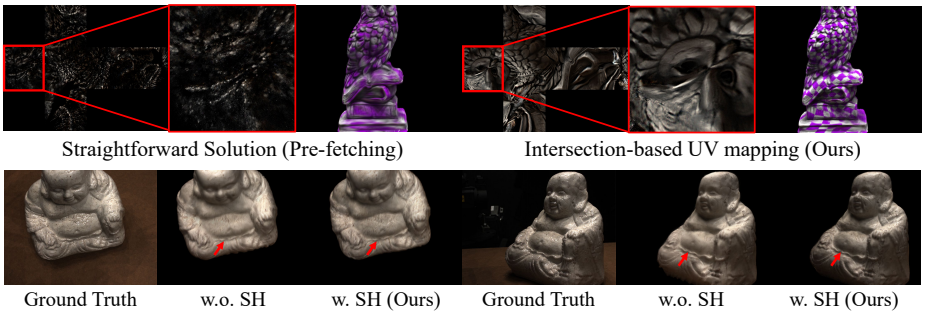


Fig. 7: Ablation study of different components in our method.

ablation of the regularization term (w.o. Reg) yields a minor improvement of average metrics, it also leads to a significant performance decline for view synthesis without per-Gaussian SH coefficients (w.o. Reg (noSH)). Without the regularization, more appearance information of reconstructed objects is captured by per-Gaussian attributes, which may hinder downstream editing applications.
Higher order Taylor expansion. We observe a minor performance gain when using second-order expansion, along with a lower rendering speed (58 FPS $\rightarrow$ 50 FPS). For computational simplicity and efficiency, we opt for the first-order Taylor expansion.

5.4 Limitations

We define the texture space as a unit sphere, thus it is ill-suited to represent multiple objects or outdoor scenes with our method. Representing the UV mapping with multiple charts, such as Nuvo [21], can address this problem, which is not the focus of this paper. In addition, our method may produce improper UV mapping for objects with thin structures or cavities. Taking NeRF Synthetic [17] chair back as an example, our method cannot differentiate the front face from the back face, leading to the wrong texture values and UV mapping.

6 Conclusion

We present a novel method, namely Texture-GS, which disentangles the appearance and geometry for 3D-GS to facilitate various real-time appearance editing operations, such as texture swapping. To achieve this, we incorporate a texture mapping module into 3D-GS, which consists of a UV mapping MLP that projects 3D points into 2D UV space, a local Taylor expansion for efficient UV mapping, and a learnable 2D texture to capture the appearance of 3D scenes. Experiments demonstrate our method can reconstruct high-fidelity textures from multi-view images, and support various real-time texture editing applications. We hope this work will provide a more profound comprehension of the relationship between 3D-GS and meshes, serving as a launchpad for further exploration.

Acknowledgements

Tian-Xing Xu completed this work during his internship at Tencent AI Lab. The project was supported by the Tencent Rhino-Bird Elite Training Program, the National Key Research and Development Program of China (No. 2023YFF0905104), the Natural Science Foundation of China (No. 62132012, 62361146854), Beijing Municipal Science and Technology Project (No. Z221100007722001) and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

References

1. Aanæs, H., Jensen, R.R., Vogiatzis, G., Tola, E., Dahl, A.B.: Large-scale data for multiple-view stereopsis. International Journal of Computer Vision **120**, 153–168 (2016)
2. Chen, Y., Chen, Z., Zhang, C., Wang, F., Yang, X., Wang, Y., Cai, Z., Yang, L., Liu, H., Lin, G.: Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. arXiv preprint arXiv:2311.14521 (2023)
3. Chen, Y., Wang, X., Chen, X., Zhang, Q., Li, X., Guo, Y., Wang, J., Wang, F.: Uv volumes for real-time rendering of editable free-view human performance. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16621–16631 (2023)
4. Chen, Z., Yin, K., Fidler, S.: Auv-net: Learning aligned uv maps for texture transfer and synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1465–1474 (2022)
5. Das, S., Ma, K., Shu, Z., Samaras, D.: Learning an isometric surface parameterization for texture unwrapping. In: European Conference on Computer Vision. pp. 580–597. Springer (2022)
6. Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A universe of annotated 3d objects. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13142–13153 (2023)
7. Downs, L., Francis, A., Koenig, N., Kinman, B., Hickman, R., Reymann, K., McHugh, T.B., Vanhoucke, V.: Google scanned objects: A high-quality dataset of 3d scanned household items. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 2553–2560. IEEE (2022)
8. Fang, J., Wang, J., Zhang, X., Xie, L., Tian, Q.: Gaussianeditor: Editing 3d gaussians delicately with text instructions. arXiv preprint arXiv:2311.16037 (2023)
9. Guédon, A., Lepetit, V.: Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. arXiv preprint arXiv:2311.12775 (2023)
10. Hu, X., Wang, Y., Fan, L., Fan, J., Peng, J., Lei, Z., Li, Q., Zhang, Z.: Semantic anything in 3d gaussians. arXiv preprint arXiv:2401.17857 (2024)
11. Huang, J., Yu, H.: Point'n move: Interactive scene object manipulation on gaussian splatting radiance fields. arXiv preprint arXiv:2311.16737 (2023)
12. 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. arXiv preprint arXiv:2312.14937 (2023)

13. Jiang, Y., Tu, J., Liu, Y., Gao, X., Long, X., Wang, W., Ma, Y.: Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. arXiv preprint arXiv:2311.17977 (2023)
14. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
15. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., et al.: Segment anything. arXiv preprint arXiv:2304.02643 (2023)
16. Ma, L., Li, X., Liao, J., Wang, X., Zhang, Q., Wang, J., Sander, P.V.: Neural parameterization for dynamic human head editing. ACM Transactions on Graphics (TOG) **41**(6), 1–15 (2022)
17. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
18. Moriyasu, K.: An elementary primer for gauge theory. World Scientific (1983)
19. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10684–10695 (2022)
20. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
21. Srinivasan, P.P., Garbin, S.J., Verbin, D., Barron, J.T., Mildenhall, B.: Nuvo: Neural uv mapping for unruly 3d representations. arXiv preprint arXiv:2312.05283 (2023)
22. Wang, X., Zhu, J., Ye, Q., Huo, Y., Ran, Y., Zhong, Z., Chen, J.: Seal-3d: Interactive pixel-level editing for neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17683–17693 (2023)
23. Xiang, F., Xu, Z., Hasan, M., Hold-Geoffroy, Y., Sunkavalli, K., Su, H.: Neutex: Neural texture mapping for volumetric neural rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7119–7128 (2021)
24. Xie, T., Zong, Z., Qiu, Y., Li, X., Feng, Y., Yang, Y., Jiang, C.: Physgaussian: Physics-integrated 3d gaussians for generative dynamics. arXiv preprint arXiv:2311.12198 (2023)
25. Xu, B., Hu, J., Hou, F., Lin, K.Y., Wu, W., Qian, C., He, Y.: Bi-directional deformation for parameterization of neural implicit surfaces. arXiv preprint arXiv:2310.05524 (2023)
26. Yang, B., Bao, C., Zeng, J., Bao, H., Zhang, Y., Cui, Z., Zhang, G.: Neumesh: Learning disentangled neural mesh-based implicit field for geometry and texture editing. In: European Conference on Computer Vision. pp. 597–614. Springer (2022)
27. Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Ronen, B., Lipman, Y.: Multiview neural surface reconstruction by disentangling geometry and appearance. Advances in Neural Information Processing Systems **33** (2020)
28. Ye, M., Danelljan, M., Yu, F., Ke, L.: Gaussian grouping: Segment and edit anything in 3d scenes. arXiv preprint arXiv:2312.00732 (2023)
29. Yu, H., Julin, J., Milacski, Z.Á., Niinuma, K., Jeni, L.A.: Cogs: Controllable gaussian splatting. arXiv preprint arXiv:2312.05664 (2023)
30. Zhan, F., Liu, L., Kortylewski, A., Theobalt, C.: General neural gauge fields. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net (2023), https://openreview.net/pdf?id=XWkWK2UagFR

31. Zhou, S., Chang, H., Jiang, S., Fan, Z., Zhu, Z., Xu, D., Chari, P., You, S., Wang, Z., Kadambi, A.: Feature 3dgs: Supercharging 3d gaussian splatting to enable distilled feature fields. arXiv preprint arXiv:2312.03203 (2023)
32. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Ewa splatting. IEEE Transactions on Visualization and Computer Graphics **8**(3), 223–238 (2002)