Text-Driven Controllable 3D Editing with GStex

Qingyang Bao, Victor Rong, David Lindell

Abstract—Scene representations model both geometry and appearance, and having intuitive ways of editing these attributes is useful for artists and modelers. Differentiable rendering can be applied to massage a representation to correspond to a given text prompt. Gaussian splatting is a new scene representation which has already been applied in a wide variety of settings, including generative AI. However, it tends to struggle with appearance editing due to its coupling of geometry and appearance. One recent work, GStex, applies textures to each Gaussian. This allows a clearer separation between the scene's geometry and appearance, and it has been applied to appearance editing, but not in the generative setting with text-based prompts. In this project, we propose using GStex for text-based appearance editing. We follow the pipeline proposed by GaussCtrl and compare the utility of GStex for this application compared to the original Gaussian splatting representation.

Index Terms—3D Editing, Diffusion Models, Gaussian Splatting, Neural Radiance Fields

1 Introduction

Traditional processes for editing 3D models requires specialized tools and hours of effort to sculpt and retexture original 3D objects. The emergence of Neural Radiance Fields (NeRFs) [1] has spurred significant interest in the creation of photorealistic 3D content, with learned 3D representations demonstrating strong capabilities in producing high-quality renders from novel views. Through neural rendering of scene representations, text-based editing using stable diffusion [2] can be applied to modify a scene according to a given text prompt and a set of reference images.

Following the success of NeRFs, 3D Gaussian splatting (3DGS) [3] has been proposed as a new 3D representation with comparable rendering quality and extremely fast rendering speeds. However, 3DGS tend to struggle with appearance editing when used as the scene representation backbone. For example, InstructGS2GS [4] is a method which takes InstructNeRF2NeRF [5], the first pipeline proposed for editing a NeRF scene using text-based instructions, and instead uses a 3DGS scene. It leverages an image-conditioned diffusion model to edit the rendered image while updating the underlying representation. The results using a 3DGS model are noticeably worse than the results using a NeRF model. GaussCtrl [6], a follow-up work for editing 3DGS scenes, utilized a depth-based ControlNet [7] to improve the multi-view consistency of the edited images. Again, the appearance editing quality is limited.

The constrained quality of 3DGS's edited appearance is due to the coupling of geometry and appearance inherent to the design of Gaussian splatting. 3DGS represents the scene as a set of geometric primitives for efficient rendering. However, each primitive has just a single colour across its extent. As a result, it may not be possible to achieve a high-fidelity edit by manipulating each Gaussian's colours. One recent work, GStex [8], applies a texture to each Gaussian. This allows a clearer separation between the scene's geometry and appearance and it has been applied to appearance editing, though not in the generative setting with text-based prompts.

For this project, we propose using GStex as the scene representation backbone for text-based appearance editing. We follow the pipeline of GaussCtrl, which uses ControlNet with depth-conditioning. Our appearance editing results are best when using GStex as the scene representation and a depth-conditioned ControlNet as our edit method. Following GStex, we first initialize the model using 2D Gaussian splatting (2DGS) [9] and assign textures to each 2D Gaussian to reconstruct the original 3D scene. As depth maps are naturally view-consistent, we condition the editing on the disparities using ControlNet. We then convert the images and their disparities into latent space and then use the denoising process with the given text prompts. Following GaussCtrl's design choices, we also select several reference views during the editing process and align all the edits to these reference views. This greatly improves multi-view consistency across the entire edited dataset.

We evaluate our method on several scenes with different prompts. The experimental results demonstrate that our method has better appearance editing while maintaining consistency with the original scene's geometry thanks to the decoupled appearance and geometry of GStex.

2 Related Work

2.1 Learned 3D Scene Representations

Differential rendering methods learn implicit 3D scene representations from multi-view images. NeRF [1] proposed modelling an implicit scene representation using a neural network which takes coordinate positions as input, a representation now referred to as a neural field. Though they were able to show state-of-the-art quality at the time, their models took several hours to train due to the computational cost of neural network queries, as well as the volumetric rendering scheme used for implicit representations. A number of works extended NeRF by modifying the neural field architecture and, with great effort, were able to reduce the training and rendering times. However, the largest improvement came when Gaussian splats replaced the neural field altogether. 3DGS [3] instead represents the scene using a set of volumetric point-based primitives. This provides two immediate benefits: 1) The splats can be initialized from an

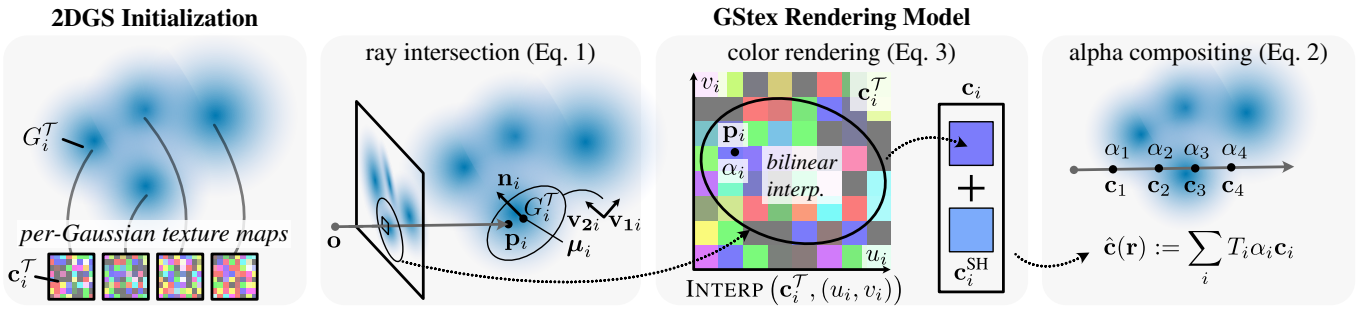

Fig. 1: **Overview of GStex [8].** The model is initialized using 2DGS [9]. An RGB texture map $\mathbf{c}_i^T$ is assigned to each 2D Gaussian primitive G_i^T. A pixel is rendered by casting a ray into the scene; the ray passes through a textured 2D Gaussian that is oriented along $\mathbf{v}_{1i}$ and $\mathbf{v}_{2i}$ with position $\boldsymbol{\mu}_i$ and normal $\mathbf{n}_i$. The color $\mathbf{c}_i$ is computed by mapping the intersection point to the UV coordinates of the Gaussian's texture map $\mathbf{c}_i^T$, bilinearly interpolating the resulting texture value, and adding it to the Gaussian's view-dependent color component $\mathbf{c}_i^{\text{SH}}$ (parameterized by spherical harmonics coefficients). The alpha value α_i is obtained by evaluating the Gaussian function at the intersection point. Finally, the rendered colors of each intersected Gaussian are alpha-composited, resulting in the final rendered pixel color $\hat{\mathbf{c}}$.

accurate point-cloud outputted by COLMAP; 2) The representation can be efficiently rendered using a rasterization scheme.

The success of 3DGS has inspired a surge of new work. Most relevant to this project are 2DGS [9] and GStex [8]. 2D Gaussian splatting represents a 3D scene with 2D Gaussian primitives, each defining an oriented elliptical disk. This gives them significant improvements over 3DGS for modelling geometry. 2D Gaussian splatting has better performance for surface extraction and is able to reconstruct meshes with state-of-the-art accuracy. GStex builds upon 2DGS uses an initial geometric reconstruction of a scene from these 2D Gaussian primitives. Then, they associate a texture map with each 2D Gaussian. By utilizing textured 2D Gaussian primitives, we can decouple the appearance and geometry.

2.2 Diffusion-Based Image Generation

Diffusion-based generative models [10] have become a prevailing framework for image generation, offering both high-quality synthesis and flexible conditioning mechanisms. Among the most influential models of these is Stable Diffusion [2], which significantly improves the efficiency and scalability of diffusion models by operating in a latent space rather than pixel space. This approach enables large-scale training and inference on consumer GPUs.

InstructPix2Pix (IP2P) [11] was a significant advancement in instruction-guided image editing by enabling fine-grained and semantically rich image manipulation using natural language. IP2P leverages the power of large language models and diffusion-based architectures to achieve greater flexibility and generalization. ControlNet [7] addresses a key limitation in text-to-image generation models such as Stable Diffusion [2], by introducing a mechanism for spatially conditioning the generation process with external inputs. While diffusion models such as Imagen [12] demonstrate remarkable generative capabilities from text prompts alone, they often lack precise control over image layout, structure, or composition. ControlNet bridges this gap by augmenting pretrained diffusion models with additional neural network branches which guide the generation using various forms of structural conditioning, such as edge maps, depth maps and segmentation masks.

2.3 3D Editing with Diffusion Models

InstructNeRF2NeRF [5] was the first text-driven 3D editing work using neural radiance field representations. This method transforms the 3D text-driven editing task into a 2D image editing task. By rendering images from the NeRF model and editing them using IP2P, the 3D representation and its subsequent renders are iteratively finetuned to fit the text prompt. InstructGS2GS [4] replaces the NeRF model in InstructNeRF2NeRF with a 3DGS model, but with limited success. GaussCtrl [6] proposes an improved 3DGS appearance editing pipeline built instead on ControlNet. They first perform a scene reconstruction using 3DGS, and render both RGB and depth images for the entire dataset. After that, they apply depth-conditioned editing for all the rendered RGB images using ControlNet. The original 3DGS is ultimately optimized using the edited images.

3 Method

Our method follows the GaussCtrl [6] pipeline to edit a GStex model with text prompts. Given a collection of images, first we will perform a scene reconstruction using 2DGS and extract the point cloud. Starting from the point cloud, we set the initial points from 2DGS then add texture maps to the Gaussians. Then we can optimize the GStex model with better appearance. Then we employ ControlNet to conduct depth-conditioned editing for all images. Finally, we optimize the GStex model using the edited images to obtain the final edited 3D model. We present our method in more detail in the following subsections.

3.1 Background

3.1.1 GStex

Since GStex [8] is based on 2D Gaussian Splatting, we first briefly overview 2DGS. Each Gaussian $G = \{\mathbf{c}, \alpha, \mathbf{R}, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$ is defined by its radiance $\mathbf{c} \in \mathbb{R}^{3S}$ (with S denoting the number of spherical harmonics coefficients used for

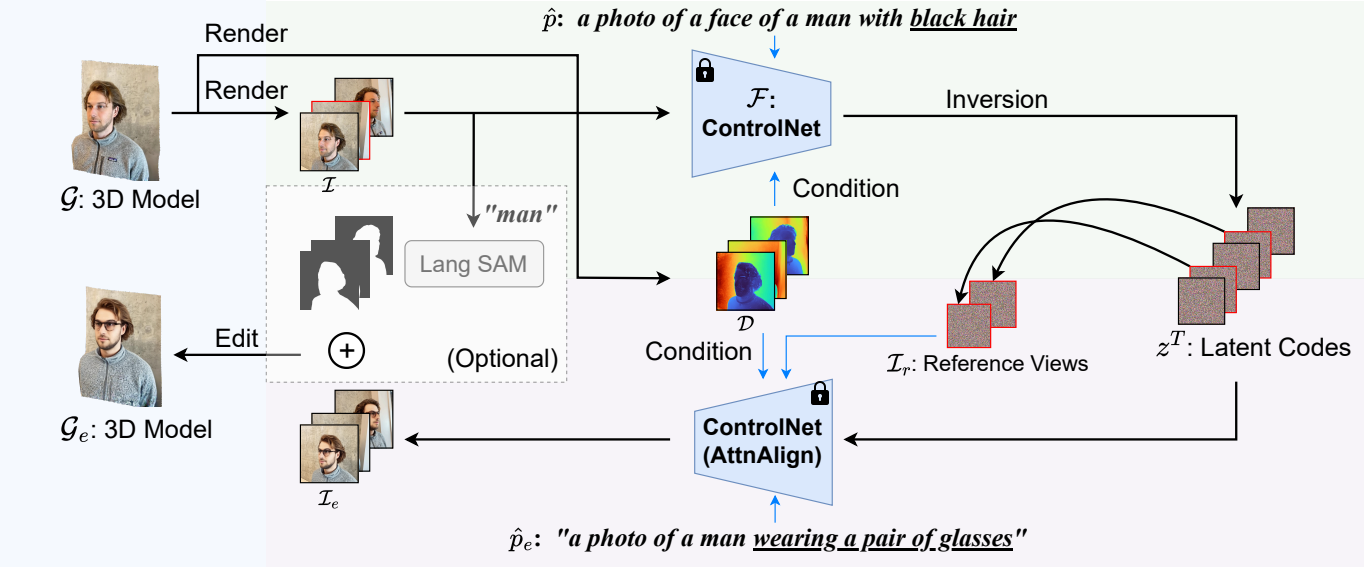

Fig. 2: **Overview of GaussCtrl [6].** Given a 3D Model scene and text instructions, GaussCtrl renders RGB and depth images from a 3DGS model, and uses ControlNet to modify the rendered RGB images. Our method replaces the original 3DGS model with a GStex model for improved appearance fidelity.

view-dependent effects), opacity $\alpha \in \mathbb{R}$, rotation matrix $\mathbf{R} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3] \in \mathrm{SO}_3$, mean position $\boldsymbol{\mu} \in \mathbb{R}^3$, and scale $\boldsymbol{\sigma} \in \mathbb{R}^3$. The surface normal of the 2D Gaussian is defined as $\mathbf{n} := \mathbf{v}_3$, and we set the corresponding scale component to zero.

Different from 3DGS, 2DGS determines visibility and contribution by directly computing ray-Gaussian intersections. Specifically, for a ray $\mathbf{r} := \mathbf{o} + t\mathbf{d}$ and the ith Gaussian G_i, the point of intersection $\mathbf{p}_i$ is computed as:

$$\mathbf{p}_i(\mathbf{r}, G_i) := \mathbf{o} + t\mathbf{d}, \quad t = \frac{\mathbf{n}_i \cdot (\boldsymbol{\mu}_i - \mathbf{o})}{\mathbf{n}_i \cdot \mathbf{d}}. \tag{1}$$

The final color $\hat{\mathbf{c}}(\mathbf{r})$ along a ray is calculated by alpha-compositing the radiance contributions of all intersected Gaussians using a standard volume rendering formulation:

$$\hat{\mathbf{c}}(\mathbf{r}) := \sum_i T_i \alpha_i(\mathbf{p}_i)\mathbf{c}_i(\mathbf{p}_i, \mathbf{d}), \quad T_i := \prod_{j=1}^{i-1}(1 - \alpha_j(\mathbf{p}_j)). \tag{2}$$

GStex modifies the radiance parameters c of 2D Gaussian splatting while keeping the other Gaussian parameters the same as described previously. Specifically, we break down the radiance $\mathbf{c}$ of a Gaussian into two parts. The first part is a spatially varying diffuse texture $\mathbf{c}^{\mathcal{T}}$, which contains RGB values defined across the Gaussian surface. The second part is a residual term $\mathbf{c}^{\mathrm{SH}}$, which captures view-dependent effects through spherical harmonics coefficients. Importantly, the constant coefficient is excluded from $\mathbf{c}^{\mathrm{SH}}$.

Our textured 2D Gaussian is thus denoted as:

$$G^{\mathcal{T}} = \left\{\mathbf{c}^{\mathcal{T}}, \mathbf{c}^{\mathrm{SH}}, \alpha, \mathbf{R}, \boldsymbol{\mu}, \boldsymbol{\sigma}\right\}.$$

The radiance of the i-th textured 2D Gaussian depends on both the spatial location $\mathbf{p}_i$ and the viewing direction $\mathbf{d}$. The radiance is computed as:

$$\mathbf{c}_i(\mathbf{p}_i, \mathbf{d}) = \mathrm{INTERP}\left(\mathbf{c}_i^{\mathcal{T}}, (u_i(\mathbf{p}), v_i(\mathbf{p}))\right) + \mathrm{SH}\left(\mathbf{c}_i^{\mathrm{SH}}, \mathbf{d}\right), \tag{3}$$

where INTERP denotes the bilinear interpolation function, and $\mathbf{c}_i^{\mathcal{T}}$ represents a $U_i \times V_i$ grid of RGB texels associated with the i-th 2D Gaussian. The ray intersection point $\mathbf{p}_i$ is projected onto the texture coordinates $(u_i(\mathbf{p}), v_i(\mathbf{p})) \in [0, U_i] \times [0, V_i]$, which are used to fetch the corresponding texture value.

For the view-dependent component $\mathbf{c}_i^{\mathrm{SH}} \in \mathbb{R}^{3(S-1)}$, we adopt the same spherical harmonics-based encoding technique as used in 2DGS to model view-dependent appearance. The radiance contribution from $\mathbf{c}_i^{\mathrm{SH}}$ and the viewing direction $\mathbf{d}$ is computed via the function SH, which evaluates RGB spherical harmonics.

This representation generalizes the 2DGS framework, where $\mathbf{c}_i^{\mathcal{T}}$ can be seen as a spatially varying approximation of the zeroth-order spherical harmonic.

3.1.2 ControlNet

ControlNet [7] uses Stable Diffusion to add conditional control to a large pretrained diffusion model, as shown above. Text prompts are encoded using the CLIP [13] text encoder, and diffusion timesteps are encoded with positional encoding. The ControlNet structure is applied to each encoder level of the U-net. In particular, we use ControlNet to create a trainable copy of the 12 encoding blocks and 1 middle block of Stable Diffusion. The outputs are added to the 12 skip-connections and single middle block of the U-net. Since Stable Diffusion follows a typical U-net structure, the ControlNet architecture is likely to be applicable with other models.

3.2 Controllable 3D Editing with GStex

Since our overall pipeline is similar to GaussCtrl [7], we also provide a brief overview of the pipeline in Fig. 2. Since our 3D scene representation has better results on geometry and appearance, we first reconstruct the scene with GStex model. Then we render the RGB and depth images for each camera pose of the original dataset.

Maintaining multi-view consistency between all the images is still a challenging problem. To edit the 3D model and maintain multi-view consistency for edited images, GaussCtrl uses ControlNet for depth-conditioned image editing. The depth maps $\mathcal{D}$ carry information about the 3D model's spatial poses, and so they are helpful to ensure the consistency across multiple views. More precisely, denote the ControlNet model as $\mathcal{F}$. This is comprised of a U-net block $\mathcal{F}_U$ and a ControlNet block $\mathcal{F}_C$. Given a to-be-edited image $\mathcal{I}$ and its corresponding description prompt $\hat{p}$, we start by inverting the original image $\mathcal{I}$ to the Gaussian noise z_ϵ. we first compute its latent code z_0, using the VAE [14] encoder of ControlNet. After that, we iteratively invert it to the noise z^T via DDIM inversion. The whole process can be described as

$$\begin{aligned} \epsilon^t &= \mathcal{F}_U\left(z^t, t, \hat{p}, \mathcal{F}_C(z^t, t, \hat{p}, \mathcal{D})\right), \\ z^{t+1} &= \sqrt{\alpha_{t+1}}\left(\frac{z^t - \sqrt{1-\alpha_t}\cdot\epsilon^t}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t+1}}\epsilon^t, \end{aligned}$$

where t is the time step of the diffusion process and α_t is the scheduling coefficient of the DDIM [15] scheduler. After we obtain z^T from DDIM inversion, we can condition on the edited prompt $\hat{p}_e$. Beginning from z_ϵ, we obtain the edited latent code z_e^0 through the denoising process:

$$\begin{aligned} \epsilon_p^t &= \mathcal{F}_U\left(z_e^t, t, \hat{p}_e, \mathcal{F}_C(z_e^t, t, \hat{p}_e, \mathcal{D})\right), \\ \epsilon_0^t &= \mathcal{F}_U\left(z_e^t, t, p_0, \mathcal{F}_C(z_e^t, t, p_0, \mathcal{D})\right), \\ \epsilon^t &= \epsilon_0^t + \rho\cdot(\epsilon_p^t - \epsilon_0^t), \\ z_e^{t-1} &= \sqrt{\alpha_{t-1}}\left(\frac{z_e^t - \sqrt{1-\alpha_t}\cdot\epsilon^t}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}}\epsilon^t, \end{aligned} \tag{4}$$

where z_e^t denotes the latent code of the edited images ($z_e^T = z_\epsilon$) and Eq. (4) is the classifier-free guidance [16]. We can then get the final edited image $\mathcal{I}_e$ by decoding z_e^0 using the VAE decoder of the ControlNet. After this process, we can now optimize the original GStex model using the edited renders. We refer to this overall method as GStexCtrl.

4 Experiments

We first introduce the experiments setup, including datasets, method baselines, our implementation and evaluation metrics. We then provide both qualitative and quantitative results of our method.

4.1 Dataset

To examine the effectiveness of our method, we evaluate our approach using a diverse set of scenes from multiple benchmark datasets. We collected several scenes from the InstructNeRF2NeRF [5], Mip-NeRF360 [17], and Blended-MVS [18] datasets. We evaluate our method on multiple text-based instructions.

Scene	Prompt	GaussCtrl	GstexCtrl
		CLIP	CLIP
Bear Statue	"A zebra statue"	0.2461	**0.2779**
Dinosaur	"A zebra on the road side"	**0.2802**	0.2795
Garden	"A table with a zebra pattern"	0.2519	**0.3070**
Stone Horse	"a photo of a statue with a zebra pattern in front of a museum"	0.2120	**0.2839**
Rabbit	"a rabbit with a zebra pattern"	0.2727	**0.3307**
	"A white rabbit with black spots"	0.2875	**0.3085**

TABLE 1: **Quantitative metrics.** We show the CLIP scores of our experiments under different prompts and scenes using GaussCtrl and GStexCtrl.

4.2 Baselines

We primarily compare the method with GaussCtrl using 3DGS as the underlying representation [6]. Our method replaces 3DGS with a GStex model. To ensure a fair comparison, we use the same hyperparameters, datasets, and prompts. We preprocess all dataset images using Nerfstudio's preprocessing script [19].

4.3 Evaluation Metrics

Following previous works, we use CLIP Text-Image Directional Similarity to evaluate the alignment of the 3D edit with text instructions [5], [6], [13]. However, as Wu et al. [6] mention, this metric may not always reflect the true visual quality of the edit as it is focuses on global consistency to the prompt while often ignoring local details. Nonetheless, we provide it for reference.

4.4 Implementation

Our method is implemented by extending methods in the Nerfstudio library, a modular PyTorch framework [19]. We use the "splatfacto" model, an improved implementation of 3D Gaussian Splatting provided in Nerfstudio. The implementation of GStex also uses the Nerfstudio framework. We employ Stable Diffusion v1.5 and its corresponding ControlNet for 2D image editing using the Huggingface library. We also apply Language-Based Segment Anything (Lang SAM) [20] when we only need to edit the object without changing the background environment. Our method takes around ten minutes to edit one scene on an NVIDIA RTX 5090 with a GPU memory of 32GB.

4.5 Results

We calculate the CLIP score across text instructions for each scene and summarize the results in Tab. 1. Our method outperforms GaussCtrl in five out of six different prompts. However, the CLIP score may not always reflect the editing

Fig. 3: **Qualitative comparisons.** We show appearance edits using 3DGS and GStex as the underlying representation. For each column, the scene and prompt pairs outlined in Tab. 1 are shown, with 3DGS used in the top row and GStex used in the bottom row.

quality. Therefore, we also include renders of all the experiments that we have done in the paper. Fig. 3 illustrates various editing results of our method. To show the advantage of GStex, we choose several text instructions which produce fine-grained appearance edits of the given objects. In most cases, our method outperforms GaussCtrl.

5 Limitations

Since our method is based on a depth-conditioned ControlNet model, it will fail to modify the original object according to our desired prompt if we adjust the geometry. As the GaussCtrl paper argues, significant changes to the original geometry are not typically required in the literature of 3D appearance editing. Instead, editing tasks usually involve changing object styles, modifying environments, and adding local features.

Another limitation is the texel budget of the GStex model. For large numbers of texels (i.e. 10^8 and above), the edit is slow due to GPU memory constraints. It also takes longer for the pretrained model to optimize and converge. When we use too many parameters for to represent the fine appearance, modifications become slower. However, this problem is simply a matter of resources – with enough iterations, it performs well.

Finally, the text conditioning of the diffusion model is still limited. As a result, the set of prompts we were able to use is constrained, and the specific wording had to be finetuned for each scene. Though this portion of the method is separate from the 3D representation, it can lead to noticeable discrepancies in the edits. For example, the 3DGS edits of the rabbit scene shown in Fig. 3 are significantly different from the GStex edits.

6 Conclusion

In this project, we compared the use of two scene representations in a controllable text-driven 3D editing method. We find that integrating a GStex model rather than a 3DGS model improves the appearance performance, while the multi-view consistency and original geometric structure are still maintained thanks to a depth-conditioned image editing process. We evaluated the performance of our method on various scenes and text prompts and show both quantitatively and qualitative that our method outperforms GaussCtrl for appearance editing.

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 10 684–10 695.

[3] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering." *ACM Trans. Graph.*, vol. 42, no. 4, pp. 139–1, 2023.

[4] C. Vachha and A. Haque, "Instruct-gs2gs: Editing 3d gaussian splats with instructions," https://instruct-gs2gs.github.io/, 2024, online.

[5] A. Haque, M. Tancik, A. A. Efros, A. Holynski, and A. Kanazawa, "Instruct-nerf2nerf: Editing 3d scenes with instructions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 19 740–19 750.

[6] J. Wu, J.-W. Bian, X. Li, G. Wang, I. Reid, P. Torr, and V. A. Prisacariu, "Gaussctrl: Multi-view consistent text-driven 3d gaussian splatting editing," in *European Conference on Computer Vision*. Springer, 2024, pp. 55–71.

[7] L. Zhang, A. Rao, and M. Agrawala, "Adding conditional control to text-to-image diffusion models," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 3836–3847.

[8] V. Rong, J. Chen, S. Bahmani, K. N. Kutulakos, and D. B. Lindell, "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)*. IEEE, 2025, pp. 3508–3518.

[9] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2d gaussian splatting for geometrically accurate radiance fields," in *ACM SIGGRAPH 2024 conference papers*, 2024, pp. 1–11.

[10] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[11] T. Brooks, A. Holynski, and A. A. Efros, "Instructpix2pix: Learning to follow image editing instructions," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 18 392–18 402.

[12] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in neural information processing systems*, vol. 35, pp. 36 479–36 494, 2022.
[13] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PmLR, 2021, pp. 8748–8763.
[14] D. P. Kingma, M. Welling *et al.*, "Auto-encoding variational bayes," 2013.
[15] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.
[16] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.
[17] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5470–5479.
[18] Y. Yao, Z. Luo, S. Li, J. Zhang, Y. Ren, L. Zhou, T. Fang, and L. Quan, "Blendedmvs: A large-scale dataset for generalized multi-view stereo networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1790–1799.
[19] M. Tancik, E. Weber, E. Ng, R. Li, B. Yi, T. Wang, A. Kristoffersen, J. Austin, K. Salahi, A. Ahuja *et al.*, "Nerfstudio: A modular framework for neural radiance field development," in *ACM SIGGRAPH 2023 conference proceedings*, 2023, pp. 1–12.
[20] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4015–4026.