Repurposing 2D Diffusion Models with Gaussian Atlas for 3D Generation

Supplementary Material

A. Supplementary for GaussianVerse

A.1. Fitting Details

As described in Section 3, we adopted Scaffold-GS [24] as the base model for per-object 3D Gaussian fitting. We observed duplicated Gaussians when the number of offsets was large, even at the default value. Therefore, we reduced the number to 4 to allow more anchors to be initialized with random values. The 3D Gaussians were then optimized according to the objective in Equation 2, with λ'_{rgb} set to 0.8, λ'_{ssim} set to 0.2, λ'_{lpips} set to 0.02, and λ'_{reg} set to 0.01.

A.2. 3DGS Quality of GaussianVerse

We compare our proposed 3DGS fitting method in Section 3 with the densification-constrained fitting method introduced by GaussianCube [57] and the upper-bound method, Scaffold-GS [24], in Figure 11. Compared to GaussianCube, our method achieves better rendering quality. Compared to the upper bound, our results show no significant visual degradation while using significantly fewer 3D Gaussians. As shown in Figure 11, the rendering quality of fitted 3DGS does not improve with an increased number of Gaussians. This observation confirms that many Gaussians contribute minimally to the final rendering quality and can be merged or pruned during fitting.

A.3. Comparison with Other 3D Gaussian Datasets

GaussianVerse is a large-scale dataset consisting of high-quality 3D Gaussian fittings for a wide range of objects. We note that there are a few other studies that also fit per-object 3D Gaussians to assist in the training of diffusion models. A direct comparison is provided in Table 3.

B. Supplementary for Gaussian Atlas Formulation

B.1. Formulation Details

As discussed in Section 4, the adaptive nature of 3DGS fittings in GaussianVerse enables faster transformation of 3DGS to 2D Gaussian Atlas compared to the similar process in [57]. Notably, the computation time is significantly reduced during the non-square Optimal Transport step for *sphere offsetting*, since the number of 3D positions $\{\mathbf{x}\}$ is typically much smaller than N, the number of surface points $\{s\}$ on the standard sphere S. After sphere offsetting, for 3DGS fittings that have a smaller size than N, we pad additional Gaussians by duplicating the ones with the smallest scales and set their opacity to 0. For 3DGS fittings that have more Gaussians than N, we prune the Gaussians with the smallest scales before sphere offsetting. In practice, constructing one Gaussian atlas takes an average of approximately one minute.

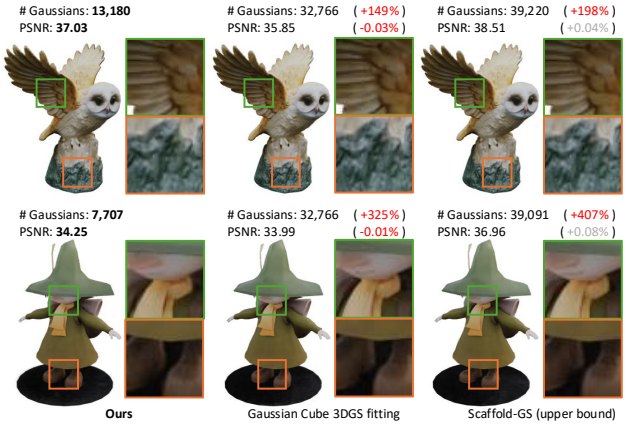

Figure 11. **Comparisons of 3DGS fitting methods.** Our method achieves high fitting quality comparable to the upper bound with significantly fewer valid 3D Gaussians with positive opacity.

C. Supplementary for Finetuning LD with Gaussian Atlas

C.1. Training Diffusion Model Without VAEs

The typical fine-tuning approach involves VAE encoding and decoding [14]. However, we argue that such a VAE auto-encoding is inappropriate for Gaussian atlases due to three reasons: *(i)* VAE encoding is a lossy compression of the atlases, whose accuracy is crucial for Gaussian rendering. We demonstrate the impact of using VAE for auto-encoding Gaussian atlases in Figure 12; *(ii)* Even at the maximum bound $\tau = 36,864 = 192 \times 192$, the number of Gaussians is significantly smaller than the number of pixels required for VAE input (e.g., 768×768). Naive up-sampling of the Gaussian Atlas $\mathbf{X}$ would significantly increase computational costs; *(iii)* Even with scaling and shifting, $\mathbf{X}$ does not visually resemble the natural RGB images originally used for VAE training, but rather analogizes latent features. Considering these, we remove the original VAE and fine-tune the LD UNet directly on Gaussian atlases. We make appropriate modifications to the input and output layers of the UNet to accommodate all Gaussian attributes with the correct number of channels [14].

C.2. More Details

Diffusion model training. We set λ_{diff} to 1.0, λ_{rgb} to 10.0, λ_{mask} to 1.0, and λ_{lpips} to 1.0 to finetune the LD

Methods	Venue	# of 3DGS fittings	# of Gaussians per fitting
GSD [30]	ECCV 2024	6,000	1,024
GVGen [10]	ECCV 2024	∼46,000	32,768
GaussianCube [57]	NeurIPS 2024	125,653	32,768
ShapeSplat [27]	3DV 2025	∼65,000	>20,000
GaussianVerse (ours)	-	205,737	10,435

Table 3. GaussianVerse contains large-scale 3DGS fittings with adaptive number of Gaussians, which allows various applications.

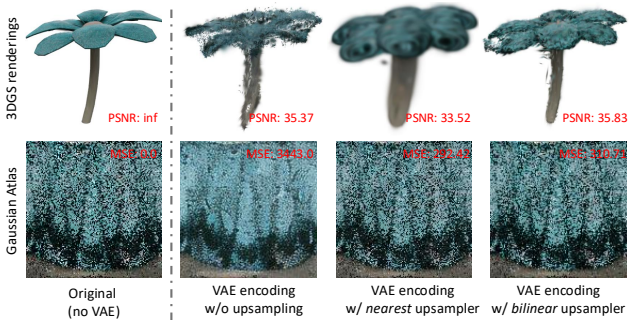

Figure 12. **Latent diffusion VAE degrades 3D Gaussian quality.** In the bottom row, we present a transformed Gaussian atlas and its reconstruction with and without VAE auto-encoding. When the atlas is up-sampled to a higher resolution, no obvious differences are observed in the atlas. However, after rendering the decoded Gaussians to images, we can observe much higher disparities, resulting in significantly reduced PSNR and visual quality. This suggests that incorporating the VAE for 3DGS diffusion is impractical.

Figure 13. **Comparison with a 2D generation approach [45].** Note that we provide MVDream [41] generated 2D images (shown as the main images) to initialize 3D generations of [45] (shown as the smaller images).

UNet. At each training step, we randomly sample one view of 2D renderings to compute the photometric losses. We enable gradient checkpoint to save GPU memory, which allows a local batch size of 8 for each GPU, with a total batch size of 64 across 8 GPUs.

D. More Results

D.1. More Comparisons

Since Omages [55] does not support text-to-3D generation and the implementations of PI3D [21], HexaGen3D [28], and GIMDiffusion [5] are not publicly available, we provide additional qualitative comparisons against Splatter Image [45], a method that also generates 2D representations of 3D objects. As shown in Figure 13, while Splatter Image reconstructs the complete geometry of 3D objects with few artifacts, it struggles to generate coherent appearances with meaningful details. In contrast, our method leverages the prior knowledge embedded in a pre-trained 2D diffusion model, thereby yielding significantly improved generation quality.

D.2. More Generations

More results generated from a diverse list of prompts are shown in Figure 16 and Figure 17. We demonstrate impressive generation quality on text prompts involving descriptions for color, shape, style, abstract semantics, and quantities for objects. Additionally, we include videos of 360-degree renderings of the generated objects along with the supplementary material.

2D flattening and surface cutting may introduce unexpected artifacts in the final 3D generations [5]. However, when rendering the generated 2D atlases back as 3D Gaussians, no defects are observed at the 'seams' of disconnected 2D boundaries, and our final 3D generations are coherent and natural, similar to other direct 3D generation approaches [10, 57]. This proves the concept of our approach — 3D generation achieved by a 2D diffusion model.

D.3. Diversity of Generations

With the same text prompt, we initiate the reverse diffusion with different random noise to demonstrate the diversity of generations in Figure 14. We observe a high diversity of geometries and appearances of the generated objects. This further proves that our proposed 2D representations of 3D Gaussians does not negatively alter the underlying semantics of the original 3D properties.

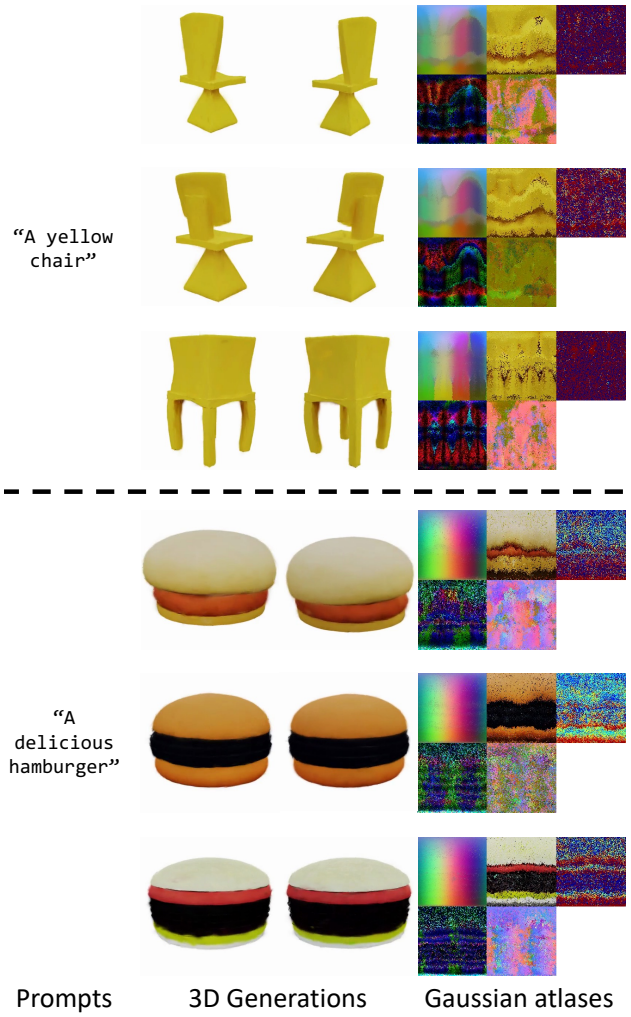

Figure 14. **Diverse text-to-3D generation results from the same prompt.** We present the generated Gaussian atlases in the order from top left to bottom right: 3D location $\mathbf{x}$, albedo $\mathbf{c}$, color-coded opacity $\mathbf{o}$, normalized scale $\mathbf{s}$, and the last three channels of normalized quaternion $\mathbf{r}$.

E. Discussions

Limitations. Our experiments revealed a trade-off between generation quality and the parameter N, which denotes the number of Gaussians per Gaussian atlas. A larger N allows for finer-grained details in the generated outputs, but it also increases both training and inference costs. In this paper, we set $N = 128 \times 128 = 16,384$ Gaussians as a good compromise between computational efficiency and quality. However, in our current setup — using approximately three times fewer Gaussians than in [57] — our 3D generations do not exhibit significantly finer details compared to state-of-the-art methods that employ substantially more Gaussians.

Moreover, since this work repurposes a specific checkpoint of the LD model, no major updates have been made to the model architecture. Consequently, our model inherits the limitations of the original LD model for 3D generation, including reduced quality for long text prompts (see Figure 15).

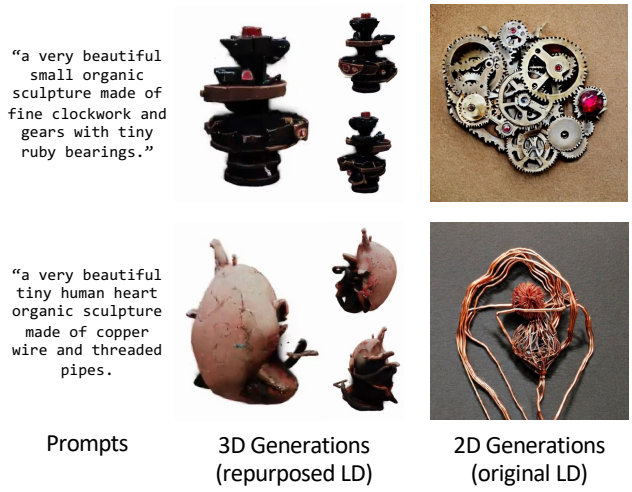

Figure 15. Similar to the original LD, our repurposed LD is not good at capturing long text prompts for 3D generations.

Future Directions. The purpose of this work is to introduce a novel way to represent 3D contents as 2D and make attempts to unify both 2D and 3D generation, allowing advancements in either paradigm to benefit both. There are two directions particularly worth pursuing by following the pathway presented in this work. The first is to experiment with more advanced diffusion models, such as transformer-based architectures [32] or latent diffusion models for Gaussians [36]. The second is to integrate plug-ins that are originaly designed for 2D diffusion models, such as IP-adapter [56] and DreamBooth [38], into 3D generations, given the unified diffusion framework.

Figure 16. **More text-to-3D generation results.** Our model is able to generate 2D Gaussian atlases to resemble high-quality 3D Gaussians from various prompts. We present the generated Gaussian atlases in the order from top left to bottom right: 3D location $\mathbf{x}$, albedo $\mathbf{c}$, color-coded opacity $\mathbf{o}$, normalized scale $\mathbf{s}$, and the last three channels of normalized quaternion $\mathbf{r}$.

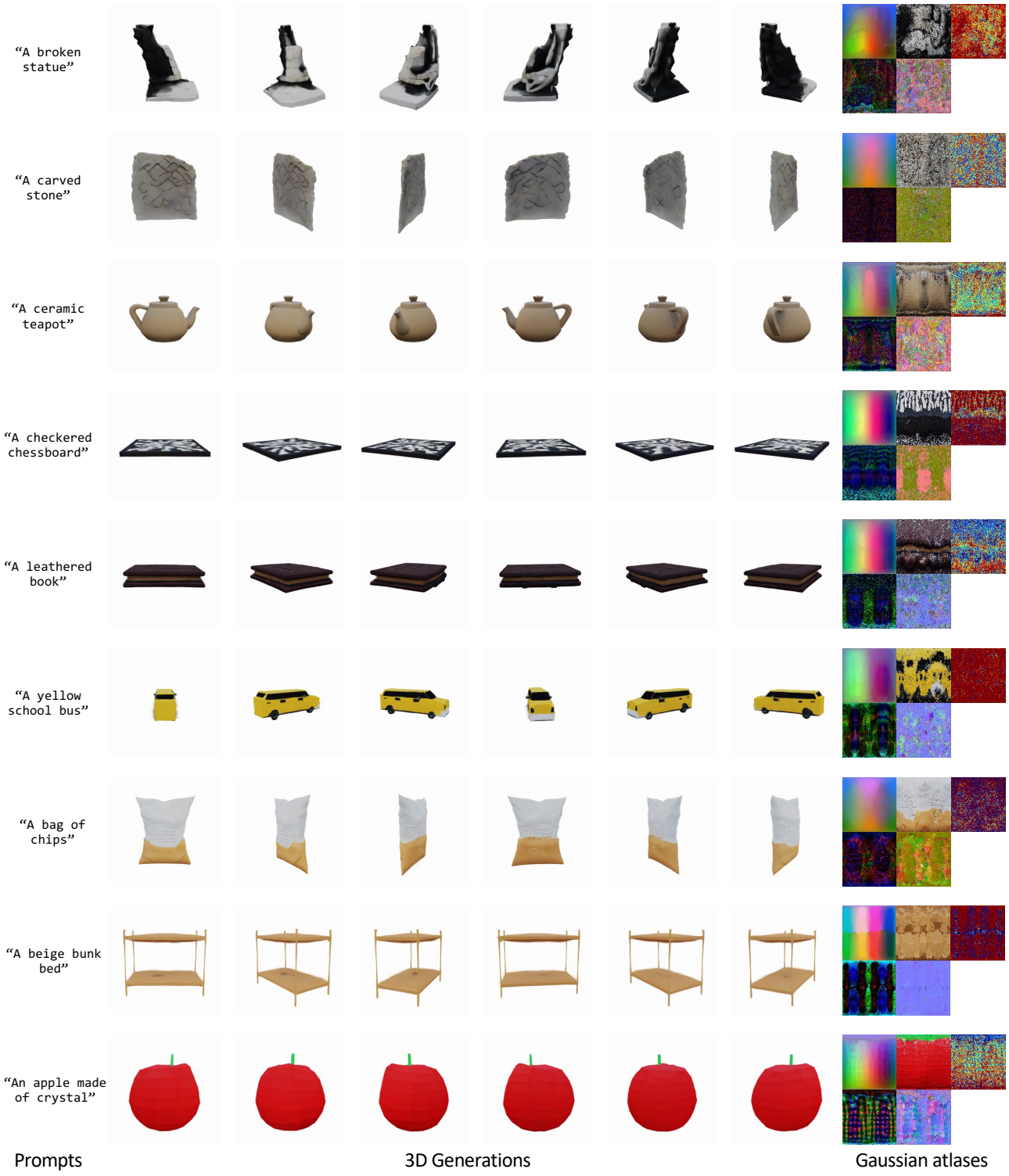

Figure 17. **More text-to-3D generation results (cont.).** Our model is able to generate 2D Gaussian atlases to resemble high-quality 3D Gaussians from various prompts. We present the generated Gaussian atlases in the order from top left to bottom right: 3D location $\mathbf{x}$, albedo $\mathbf{c}$, color-coded opacity $\mathbf{o}$, normalized scale $\mathbf{s}$, and the last three channels of normalized quaternion $\mathbf{r}$.