GVGEN: Text-to-3D Generation with Volumetric Representation

Xianglong He[1,2*], Junyi Chen[1,3*], Sida Peng[4], Di Huang[1], Yangguang Li[5], Xiaoshui Huang[1], Chun Yuan[2†], Wanli Ouyang[1], and Tong He[1†]

[1] Shanghai AI Lab
[2] Tsinghua Shenzhen International Graduate School
[3] Shanghai Jiao Tong University
[4] Zhejiang University
[5] VAST

https://GVGEN.github.io

Abstract. In recent years, 3D Gaussian splatting has emerged as a powerful technique for 3D reconstruction and generation, known for its fast and high-quality rendering capabilities. Nevertheless, these methods often come with limitations, either lacking the ability to produce diverse samples or requiring prolonged inference times. To address these shortcomings, this paper introduces a novel diffusion-based framework, GVGEN, designed to efficiently generate 3D Gaussian representations from text input. We propose two innovative techniques: (1) *Structured Volumetric Representation*. We first arrange disorganized 3D Gaussian points as a structured form GaussianVolume. This transformation allows the capture of intricate texture details within a volume composed of a fixed number of Gaussians. To better optimize the representation of these details, we propose a unique pruning and densifying method named the Candidate Pool Strategy, enhancing detail fidelity through selective optimization. (2) *Coarse-to-fine Generation Pipeline*. To simplify the generation of GaussianVolume and empower the model to generate instances with detailed 3D geometry, we propose a coarse-to-fine pipeline. It initially constructs a basic geometric structure, followed by the prediction of complete Gaussian attributes. Our framework, GVGEN, demonstrates superior performance in qualitative and quantitative assessments compared to existing 3D generation methods. Simultaneously, it maintains a fast generation speed (∼7 seconds), effectively striking a balance between quality and efficiency.

Keywords: Text-to-3D Generation · Feed-forward Generation · 3D Gaussians

1 Introduction

The development of 3D models is a pivotal task in computer graphics, garnering increased attention across various industries, including video game design,

∗ Equal Contribution.
† Corresponding Authors.

film production, and AR/VR technologies. Among the different aspects of 3D modeling, generating 3D models from text descriptions has emerged as a particularly intriguing area of research due to its accessibility and ease of use. Various methods [16, 31, 33] have been proposed to handle the task. Still, it continues to present difficulties owing to the ambiguity of texts and intrinsic domain gap between text description and corresponding 3D assets.

Previous text-to-3D approaches can be broadly classified into two categories: optimization-based generation [8, 33, 41, 48] and feed-forward generation [19, 31, 32, 40]. Optimization-based methods have become rather popular recently, due to the rapid development of text-to-image diffusion models [12, 36]. These methods usually optimize 3D objects conditioned on texts or images through Score Distillation Sampling (SDS) [33], distilling rich knowledge from 2D image generation models. Despite yielding impressive results, optimization-based methods face the Janus problem [33], manifesting as multiple faces or over-saturation problems. Additionally, the optimization of a single object can be prohibitively time-consuming, requiring extensive computational effort. Contrarily, feed-forward approaches strive to generate 3D assets directly from text descriptions, thus sidestepping the Janus problem and significantly hastening the generation process. Our work is closely related to feed-forward-based methods. However, feed-forward methods that utilize multi-view generation models often create lower-resolution 3D assets than their multi-view image counterparts. Moreover, models directly generating 3D objects from texts often encounter difficulties with semantics when complex prompts are used.

Different from previous feed-forward-based methods like [19] that follow a text-2D-3D framework, our method proposes to generate 3D assets via directly learning 3D representation. In this study, we introduce an innovative and streamlined coarse-to-fine generation pipeline, GVGEN, for generating 3D Gaussians directly from text descriptions. Leveraging the highly expressive and fast-rendering capabilities of 3D Gaussians, our method achieves not only promising results but also maintains rapid text-to-3D generation and rendering. As shown in Fig. 1, our method consists of two stages: GaussianVolume fitting and text-to-3D generation. In the first stage, we introduce GaussianVolume, a structured volumetric form composed of 3D Gaussians. Achieving this is challenging due to the sparse and unstructured nature of optimizing original 3D Gaussians. To address this, we introduce a novel Candidate Pool Strategy for pruning and densification. This approach allows for fitting high-quality volumetric representation of Gaussians, rather than unordered Gaussian points, making the generation process more conducive for a diffusion-based framework, as is utilized in the following step.

Despite the GaussianVolume establishing a structured volumetric framework that integrates seamlessly with existing diffusion pipelines, the intrinsic complexity of rich features of 3D Gaussians presents significant challenges. Specifically, capturing the distribution of a vast amount of training data effectively becomes difficult, resulting in hard convergence for the diffusion model. Addressing these challenges, we partition the text-to-3D generation into two steps: coarse geom-

etry generation and Gaussian attributes prediction. To be more specific, in the first step, we employ a diffusion model to generate the coarse geometry of objects, termed the Gaussian Distance Field (GDF) - an isotropic representation outlining the proximity of each grid point to the nearest Gaussian point's center. Following this, the generated GDF, in conjunction with text inputs, is processed through a 3D U-Net-based model to predict the attributes of GaussianVolumes, ensuring enhanced control and model convergence.

To the best of our knowledge, this is the first study to directly feed-forward generate 3D Gaussians from texts, exploring new avenues for rapid 3D content creation and applications. Our main contributions are summarized as follows:

- We introduce GaussianVolume, a structured, volumetric form consisting of 3D Gaussians. Through the innovative Candidate Pool Strategy for pruning and cloning, we accommodate high-quality GaussianVolume fitting within a fixed volumetric resolution. This framework seamlessly integrates with existing generative networks, leveraging the inherent advantages of 3D Gaussians for explicit and efficient representation.
- We propose GVGEN, an efficient text-to-3D coarse-to-fine generation pipeline that first generates geometry volume and then predicts detailed 3D Gaussian attributes, better controlling diverse geometry and appearances of generated assets. GVGEN achieves a fast generation speed (∼7 seconds) compared with baseline methods, effectively striking a balance between quality and efficiency.
- Compared with existing baselines, GVGEN demonstrates competitive capabilities in both quantitative and qualitative aspects.

2 Related Works

2.1 Text-to-3D Generation

Generating 3D objects conditioned on texts has become a challenging yet prominent research area in recent years. Previous approaches [16,29] utilize CLIP [34] as a prior for 3D asset optimization but lacked realism and fidelity. With the rise of text-to-image generative models [12,36], Dreamfusion [33] leverages Score Distillation Sampling (SDS) to generate diverse 3D objects, drawing on the rich prior knowledge embedded in these models. Subsequently, some works [6, 21, 24, 25] focus on modeling and learning multi-modal attributes of objects (e.g. colors, albedo, normals, and depths), to enhance consistency. ProlificDreamer [41] and LucidDreamer [22] introduce novel losses to improve the quality. Some studies [15, 23, 37, 38] explore predicting multi-views for objects, followed by using feed-forward methods or SDS-based optimization to generate 3D objects. Moreover, integrating the recently proposed 3D Gaussian Splatting [18], works [8,39, 46] lead to improvements in convergence speed for text-to-3D object generation. However, these methods still require significant generation time (∼hours) per object or face 3D inconsistency of generated objects.

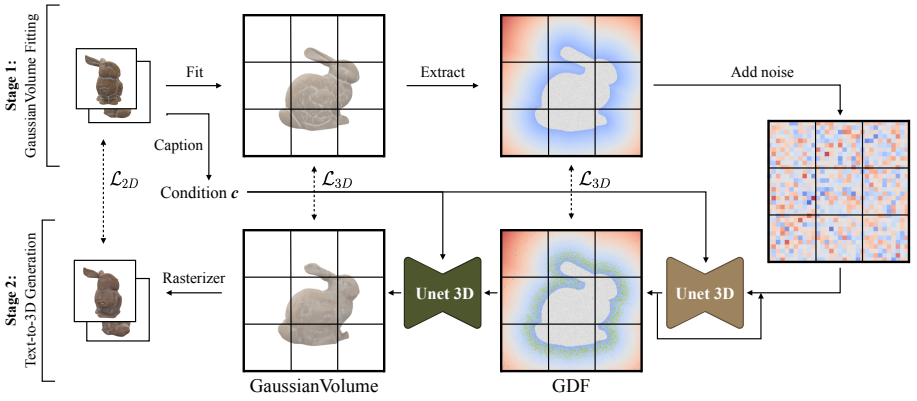


Fig. 1: Overview of GVGEN. Our framework comprises two stages. In the data pre-processing phase, we fit GaussianVolumes (Sec. 3.1) and extract coarse geometry Gaussian Distance Field (GDF) as training data. For the generation stage (Sec. 3.2), we first generate GDF via a diffusion model, and then send it into a 3D U-Net to predict attributes of GaussianVolumes.

As the emergence of large-scale 3D datasets [10,11], Feed-forward models are trained to reduce the generation time. Point-E [31] and Shap-E [17], trained on millions of 3D assets, generate point clouds and neural radiance fields respectively. 3D VADER [32] trains auto-decoders to fit latent volumes and uses them to train the diffusion model. Similarly, VolumeDiffusion [40] trains an efficient volumetric encoder to produce training data for the diffusion model. Despite these advancements, text-to-3D Gaussian generation remains largely unexplored, primarily due to the complexity of organizing disorganized Gaussians. Our work introduces a coarse-to-fine pipeline to feed-forward generate 3D Gaussians from texts, by first generating the object's coarse geometry and then predicting its explicit attributes, pioneering in the direct generation of 3D Gaussians from text descriptions.

2.2 Differentiable 3D Representation

Since the proposal of Neural Radiance Field (NeRF) [28], various differentiable neural rendering methods [2,3,5,30] have emerged, which demonstrate remarkable capabilities in scene reconstruction and novel view synthesis. These methods are also widely used in 3D generation tasks. Instant-NGP [30] employs feature volumes for accelerations by querying features only at the corresponding spatial positions. Works [3,5] decompose features into lower dimensions for faster training and less memory storage. However, they still have some shortcomings compared with point-based rendering methods [4,45] in the aspects of rendering speed and explicit manipulability.

In recent research, 3D Gaussian Splatting [18] has received widespread attention. It adopts anisotropic Gaussians to represent scenes, achieving real-time

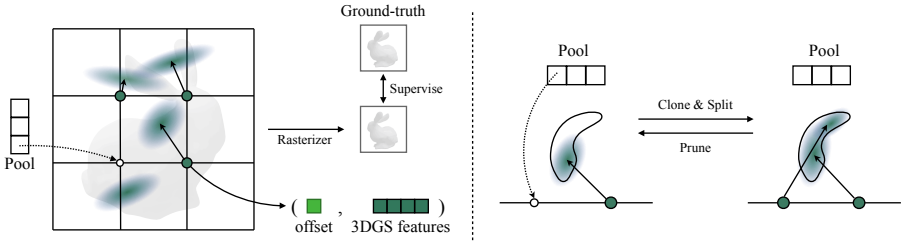


Fig. 2: Illustration of GaussianVolume Fitting. We organize a fixed number of 3D Gaussians in a volumetric form, termed GaussianVolume. By using position offsets to express slight movements from grid points to Gaussian centers, we can capture the details of objects. The proposed Candidate Pool Strategy (CPS) (Sec. 3.1) enables effective pruning and densification with a pool storing pruned points.

rendering and facilitating downstream tasks like 3D generation [35,44,47], scene editing [7] and dynamic scene rendering [42]. We introduce a strategy to fit 3D Gaussians as a structured volumetric form called GaussianVolume, which allows for good integration with the existing generation pipeline, leveraging the advantages of 3D Gaussians and achieving fast generation and rendering speeds.

3 Methodology

Our text-to-3D generation framework (Fig. 1), GVGEN, is delineated into two pivotal stages: GaussianVolume fitting (Sec. 3.1) and text-to-3D generation (Sec. 3.2). Initially, in the GaussianVolume fitting stage, we propose a structured, volumetric form of 3D Gaussians, termed GaussianVolume. We fit the GaussianVolume as training data for the generation stage. This stage is crucial for arranging disorganized point-cloud-like Gaussian points as a format more amenable to neural network processing. To address this, We use a fixed number (i.e. fixed volume resolution for the volume) of 3D Gaussians to form our GaussianVolume (Sec. 3.1), thereby facilitating ease of processing. Furthermore, we introduce the Candidate Pool Strategy (CPS) for dynamic pruning, cloning, and splitting of Gaussian points to enhance the fidelity of fitted assets. Our GaussianVolume maintains high rendering quality with only a small number (32,768 points) of Gaussians.

In the phase of generation Sec. 3.2, we first use a diffusion model conditioned on input texts to generate the coarse geometry volume, termed Gaussian Distance Field (GDF), which represents the geometry of generated objects. Subsequently, a 3D U-Net-based reconstruction model utilizes the GDF and text inputs to predict the attributes of the final GaussianVolume, thus achieving the generation of detailed 3D objects from text descriptions.

3.1 Stage1: GaussianVolume Fitting

The original 3D Gaussian Splatting technique offers fast optimization and real-time rendering. However, its lack of structure makes it challenging for 3D neural

Algorithm 1: GaussianVolume Fitting with Candidate Pool Strategy

Input: The number of Gaussian points N with assigned positions p, total iteration times T, multi-view images I with camera poses V, prune and densify conditions τ_p, τ_d

Output: GaussianVolume G, a volumetric form composed of a fixed number of 3D Gaussians

$t \leftarrow 0$; `// Iteration Time`
$G \leftarrow \text{Initialize}(N, p)$; `// Initialize Gaussian Points`
$P \leftarrow \emptyset$; `// Initialize Candidate Pool`
while $t < T$ **do**
  $\hat{I} \leftarrow \text{Rasterize}(G, V)$; `// Rendering`
  $\mathcal{L} \leftarrow \text{Loss}(I, \hat{I})$; `// Loss`
  $G \leftarrow \text{Optimize}(\nabla\mathcal{L})$; `// Update Gaussian Attributes`
  if *IsRefinementIteration(t)* **then**
    $G_p \leftarrow \text{PrunePoints}(G, \tau_p)$; `// Determine Pruned Points`
    $G, P \leftarrow \text{AddPointsToPool}(G_p, G, P)$; `// Make` G_p `"Deactivated"`
    $G_d \leftarrow \text{DensifyPoints}(G, \tau_d)$; `// Determine Densified Points in` G
    $G_{new} \leftarrow \text{FindPointsInPool}(G_d, P)$; `// Find Added Points in` P
    $G, P \leftarrow \text{RemovePointsFromPool}(G_{new}, G, P)$; `// Make` G_{new} `"Activated"`
  end
  if *EndRefinementIteration(t)* **then**
    $\text{ReleasePool}(G, P)$; `// Make All Points "Activated"`
  end
end

networks to process effectively. Directly generating 3D Gaussians through networks involves integrating existing point cloud networks, treating Gaussians as carriers of semantically rich features. However, previous work [43] has pointed out the difficulty of directly generating point clouds with only 3-dimensional color features. Extending such methods, especially for high-dimensional Gaussians, may make the learning process very challenging. We also attempted to extend previous work [27] using point cloud diffusion to generate 3D Gaussians, while the results were disappointing.

Recent works [44,49] propose hybrid representations combining 3D Gaussians with triplane-NeRF, utilizing point clouds for geometry and triplane features for texture querying, to finally decode 3D Gaussians. While these methods demonstrate effectiveness, the reconstructed results are heavily constrained by the accuracy of predicted point clouds and lose the direct operability advantage of 3D Gaussians. In light of these challenges, we introduce GaussianVolume, which is a volume composed of a fixed number of 3D Gaussians. This innovation facilitates the processing of 3D Gaussians by existing generation and reconstruction models, retaining the efficiency benefits of Gaussians. The illustration of the fitting process can be found in Fig. 2.

GaussianVolume In 3D Gaussian Splatting [18], features of 3D Gaussians G include the following attributes: a center position $\mu \in \mathbb{R}^3$, covariance matrix Σ, color information $c \in \mathbb{R}^3$ (when SH order=0) and opacity $\alpha \in \mathbb{R}$. To better optimize the covariance matrix, Σ is analogous to describing the configuration of an ellipsoid via a scaling matrix $S \in \mathbb{R}^3$ and a rotation matrix $R \in \mathbb{R}^{3\times3}$, satisfying:

$$\Sigma = RSS^TR^T. \tag{1}$$

In the implementation, S, R could be stored as a 3D vector $s \in \mathbb{R}^3$ and a quaternion $q \in \mathbb{R}^4$, respectively. With these differentiable features, 3D Gaussians can be easily projected to 2D splats and employ neural point-based α-blending technique to render 2D-pixel colors. An efficient tile-based rasterizer is used for fast rendering and backpropagation.

In this work, we represent each object with a volume $V \in \mathbb{R}^{C\times N\times N\times N}$ composed of 3D Gaussians, which is trained from multi-view images of the object. Here, C, N represent the number of feature channels and volume resolution respectively, and the number of 3D Gaussians is N^3. We follow the convention and rendering methods in original works, but use position offsets $\Delta\mu$ to express slight movements between the position p of each grid point in the volume and the center μ of the Gaussian point it represents:

$$\mu = p + \Delta\mu. \tag{2}$$

During the fitting phase, we only apply the backward operation on the offsets $\Delta\mu$, which allows the expression of more fine-grained 3D assets and also imposes a restriction to form better structures. Due to the fixed number of Gaussians, we cannot directly apply the original strategy for densification control. Instead, we propose the Candidate Pool Strategy for effectively pruning and cloning.

Candidate Pool Strategy We can not directly apply the original interleaved optimization/density strategy to move 3D Gaussians in our approach, due to the dynamical changes in the number of Gaussians not suitable for a fixed number of Gaussians. Densifying or pruning Gaussians freely is not allowed since Gaussians are bijective with assigned grid points. A naive way to avoid the problem is to only adjust the offsets $\Delta\mu$ relying on gradient back-propagation. Unfortunately, we experimentally found that the movement range of Gaussian centers becomes largely limited without a pruning and densification strategy, which leads to lower rendering quality and inaccurate geometry (see Fig. 7(a)). For this reason, we propose a novel strategy (Algorithm 1) to densify and prune the fixed number of Gaussians. The key point to our strategy is storing pruned points in a candidate pool P for later densification.

We initially align Gaussian centers, μ, with assigned positions, p and set offsets $\Delta\mu = 0$ (Line 2 in Algorithm 1). The candidate pool P is initialized as an empty set (Line 3). This pool stores "deactivated" points, which refer to pruned points during optimization – they are not involved in the forward and backward process. We optimize each asset for T iterations. Following original 3D Gaussian

Splatting [18], thresholds τ_p, τ_d on view-space position gradients are used for pruning and densifying every fixed iterations (*IsRefinementIteration(t)* in Line 8).

During the refining process, once Gaussian points G_p are pruned with predefined threshold τ_p, they are added into candidate pool P, making them "deactivated" for rendering and optimization (Lines 9-10). For densification, we first determine points G_d that should be densified in the "activated" set of Gaussians G. Then newly added points G_{new} are selected from candidate pool P via some criterion (e.g., we use the nearest point within a certain distance $\epsilon_{offsets}$ to the densified points G_d). The corresponding coordinate offsets for the added points are calculated. They become "activated" for both forward and backward processes and are subsequently removed from the candidate pool P (Lines 11-13). At the end of optimization/density, all points in the candidate pool are reintroduced into the optimization for further refinement (Lines 15-17).

This strategy ensures that Gaussian points can adaptively move during the optimization process, to represent more intricate object shapes. Simultaneously, the resulting structured volumetric form maintains its physical meaning, demonstrating a balance between adaptability and well-defined structure.

Training Loss The final loss for supervision is the original loss used in 3D Gaussian Splatting adding a regularization loss:

$$\mathcal{L}_{offsets} = Mean(\text{ReLU}(|\Delta\mu - \epsilon_{offsets}|)), \tag{3}$$

where $\epsilon_{offsets}$ is a hyper-parameter, to restrict the center of 3D Gaussians not too far from their corresponding grid points,

$$\mathcal{L}_{fitting} = \lambda_1\mathcal{L}_1 + \lambda_2\mathcal{L}_{SSIM} + \lambda_3\mathcal{L}_{offsets} \tag{4}$$

After training, the points are sorted in spatial order according to volume coordinates as training data for the generation stage. Once the training is over, 2D images of target objects could be rendered at an ultra-fast speed since each GaussianVolume is rather lightweight. More implementation details can be found in the supplemental materials.

3.2 Stage2: Text-to-3D Generation

As discussed in work [40], 3D volume space is high-dimensional, which potentially poses challenges in training the diffusion model. Empirically, we found that even though we form 3D Gaussians as a volumetric structure, it still takes too long for model convergence when training with a large amount of data. Moreover, learning the data distribution of GaussianVolumes from diverse categories via a single diffusion model causes it to collapse to the single-step reconstruction, leading to the degradation of generation diversity. To overcome these challenges, we introduce a coarse-to-fine pipeline that first generates coarse geometry volumes (Gaussian Distance Field, GDF) and then predicts the attributes of the

Fig. 3: Visualization of GaussianVolume Fitting. The rendering results demonstrate excellent reconstruction performance of GaussianVolumes.

GaussianVolume. In the first step, we adopt a diffusion model conditioned on input texts to generate GDF. In the second step, we employ a 3D U-Net-based model to predict the attributes of Gaussians by inputting the GDF along with the text condition, leading to the final GaussianVolume.

Gaussian Distance Field Generation The Gaussian Distance Field (GDF) $F \in \mathbb{R}_0^{+1\times N\times N\times N}$ stores an isotropic feature representing the fundamental geometry of a GaussianVolume. It measures the distance between each grid coordinate and its nearest Gaussian point, similar to the definition of Unsigned Distance Field. This attribute can be easily extracted from fitted GaussianVolumes via sorting algorithms. After obtaining the ground truth GDF, we train a diffusion model conditioned on texts to generate GDF, creating the coarse geometry of objects. The model is supervised by minimizing Mean Square Error (MSE) loss $\mathcal{L}_{3D}$ between the ground truth GDF and the generated GDF, which is equivalent to predicting added noises in the diffusion process. Using diffusion models, the generation diversity with respect to object shape is introduced.

GaussianVolume Prediction After generating GDF, we send it into a U-Net-based model modified from SDFusion [9], along with text descriptions, to predict all attributes of the GaussianVolume. The reason why a reconstruction model is adopted is that we empirically found that the single-step model produces comparable predictions of Gaussian attributes with a diffusion model.

We use two kinds of losses in this phase: MSE loss $\mathcal{L}_{3D}$ between ground truth GaussianVolume and the predicted one, and rendering loss $\mathcal{L}_{2D}$:

$$\mathcal{L}_{2D} = \lambda\mathcal{L}_1 + (1-\lambda)\mathcal{L}_{SSIM} \tag{5}$$

for rendered images, which are composed of L1 and SSIM loss. The total loss is,

$$\mathcal{L} = \lambda_{3D}\mathcal{L}_{3D} + \lambda_{2D}\mathcal{L}_{2D} \tag{6}$$

Using a multi-modal loss balances global semantic and local details, and keeps the training process more stable. More implementation details about model architectures and data processing can be found in the supplemental materials.

Fig. 4: Comparisons with State-of-the-art Text-to-3D Methods. Our method achieves competitive visual results with better alignments with text conditions.

4 Experiments

4.1 Baseline Methods and Dataset

Baseline Methods We compare with both feed-forward-based methods and optimization-based methods. In the feed-forward category, we evaluate Shap-E [17] and VolumeDiffusion [40], both of which directly generate 3D assets. For optimization-based methods, we consider DreamGaussian [39] as the baseline, where coarse 3D Gaussians undergo optimization with SDS loss [33] and are subsequently converted into meshes for further refinement. For VolumeDiffusion, we report results generated in the feed-forward process, without post-optimization using SDS via pretrained text-to-image models, to ensure fair comparisons with other methods.

Dataset Our training dataset comprises the Objaverse-LVIS dataset [11], which contains $\sim$ 46,000 3D models in 1,156 categories. For text prompts for training,

Fig. 5: Text-to-3D Generation Results by GVGEN.

we use captions from Cap3D [26], which leverages BLIP-2 [20] to caption multi-view images of objects and consolidates them into single captions through GPT-4 [1]. For evaluation, we generate 100 assets and render 8 views for each asset. The camera poses for these views are uniformly sampled around the object. Our method excels in both visualization and semantic alignment.

4.2 Qualitative and Quantitative Results

GaussianVolume Fitting We present the reconstruction results of GaussianVolume Fitting in Fig. 3. These results demonstrate that the proposed GaussianVolume can express high-quality 3D assets with a small number of Gaussian points under the volume resolution $N = 32$. Higher resolutions yield better rendering effects but require more computational resources for the generation stage. Additional ablation studies about fitting GaussianVolume could be found in Sec. 4.3 and the supplemental materials.

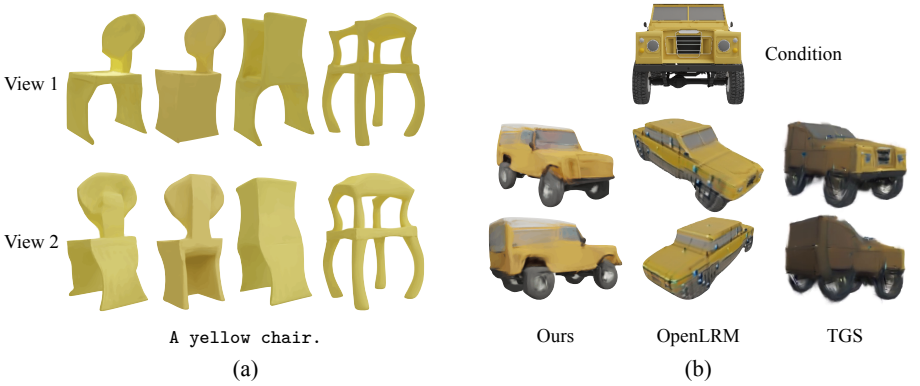


Fig. 6: Display of Generation Diversity. (a) displays diverse generated assets according to the same text prompt via GVGEN. (b) shows the comparisons among our image-conditioned GVGEN, OpenLRM [13] and TGS [49]. The single-view reconstruction models suffer from the problem of average patterns, leading to implausible shapes and textures in unseen regions, while GVGEN produces reasonable appearances and geometries.

Text-to-3D Generation We provide visual comparisons and quantitative analyses of our method and existing baselines in Fig. 4 and Tab. 1, respectively. As depicted in the figure, our model generates reasonable geometry and plausible textures. Shap-E [17] can produce rough shapes but sometimes result in misaligned appearances with respect to the input texts. VolumeDiffusion [40] tends to generate unrealistic textures. And DreamGaussian [39], which adopts an optimization-based approach, always produces over-saturated results. For quantitative results, we compare the CLIP score between rendered images and the corresponding texts, and also the generation time of these methods. We randomly select 300 text prompts for evaluation from the 40 LVIS categories containing the highest number of objects. For each object, we render 8 views in uniformly sampled camera poses. To measure generation time, we set the sampling steps to 100 for diffusion-based methods. The configuration of DreamGaussian [39] follows the default settings. Refer to Fig. 5 for more generation results.

Generation Diversity As depicted in Fig. 6(a), GVGEN can generate diverse assets conditioned on the same prompt. The generative diversity of our method not only differentiates it from reconstruction approaches but also heightens the imaginative capabilities of users. Furthermore, we develop an image-to-3D model conditioned on CLIP image embeddings and compare the results with the recently popular single-image reconstruction models TGS [49] and LRM [14] (See Fig. 6(b)). Since we do not have access to the close-sourced LRM, we use a re-implementation of it, namely OpenLRM [13]. The single-view reconstruction models suffer from the problem of average patterns, leading to implausible shapes and textures in unseen regions, while GVGEN produces reasonable appearances

Table 1: Quantitative Comparisons against Baseline Methods on CLIP Score and Inference Speed. The metrics demonstrate the superior performance of our method.

Metrics	CLIP score ↑	Time ↓
Ours	**28.53**	**7 sec**
Shap-E	28.48	11 sec
VolumeDiffusion	25.09	7 sec
DreamGaussian	23.60	∼3 min

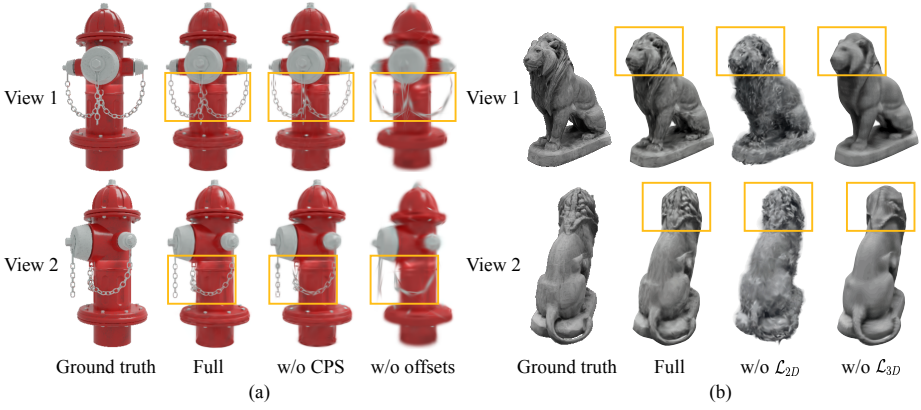


Fig. 7: Qualitative Results for Ablation Studies. (a) represents visual comparisons of different GaussianVolume fitting methods. (b) stands for results using different losses to train the GaussianVolume prediction model.

and geometries. Such comparisons accentuate the critical difference between our GVGEN and the reconstruction methods.

4.3 Ablation Studies

GaussianVolume Fitting We first study the effects of strategies fitting GaussianVolume via visual comparisons. In these ablation experiments, 3D assets are trained with 72 images, with camera poses uniformly distributed at a distance of 2.4 from the world center, and evaluated with 24 images posed uniformly at a distance of 1.6. We set volume resolution $N = 32$, i.e. number of Gaussian points $N^3 = 32,768$. The full method, as used in the data preparation stage, performs the best in terms of rendering results, even with only $32,768$ Gaussians. "w/o CPS" refers to optimizing coordinate offsets $\Delta\mu$ only via backpropagation, while "w/o offsets" means fixing Gaussian coordinates $\mu = p$ without optimizing point positions. Fig. 7(a) shows that using the full method produces the best rendering results. Tab. 2 reports the average PSNR, SSIM, and LPIPS scores, supporting

Table 2: Quantitative Metrics for Ablation Studies. The left table analyzes GaussianVolume fitting strategies, while the right table compares different losses training the GaussianVolume attributes prediction model. For qualitative comparisons, see Fig. 7.

Metrics	PSNR ↑	SSIM ↑	LPIPS ↓
Full	**30.122**	**0.963**	**0.038**
w/o CPS	29.677	0.958	0.049
w/o offsets	27.140	0.936	0.084

Metrics	PSNR ↑	SSIM ↑	LPIPS ↓
Full	35.03	**0.9872**	**0.0236**
w/o $\mathcal{L}_{3D}$	**35.21**	0.9846	0.0268
w/o $\mathcal{L}_{2D}$	29.55	0.9654	0.0444

the effectiveness of the proposed strategy quantitatively. More results can be found in the supplemental materials.

Text-to-3D Generation Due to the computational resources required, it is impractical to evaluate each part of our design with models trained on the full Objaverse-LVIS dataset for our ablation study. Therefore, we train the models on a small subset of Objaverse-LVIS to validate the effectiveness of losses in the stage of predicting GaussianVolume attributes. Specifically, we pick 1,336 assets themed animals from 24 LVIS categories for training. We randomly select ∼20% of the data from the training data and evaluate quantitative metrics with rendered novel views of each asset. Fig. 7(b) demonstrates that the proposed multi-modal losses produce plausible results with fine-grained details, and quantitative results are listed in Tab. 2. The combination of two types of losses leads to more detailed textures while keeping the entire geometry smooth.

4.4 Limitations

GVGEN has shown encouraging results in generating 3D objects. However, its performance is constrained when dealing with input texts significantly divergent from the domain of training data, as illustrated in Fig. 12 in the supplementary. Since we need to fit the GaussianVolume per object to prepare training data, it is time-consuming to scale up to millions of object data for better diversity. Additionally, the volume resolution N is set as 32 (For 3D Gaussians, the point number is only N^3=32,768) to save computational resources, which limits the rendering effects of 3D assets with very complex textures. In the future, we will further explore how to generate higher-quality 3D assets in more challenging scenarios.

5 Conclusions

In conclusion, this paper explores the feed-forward generation of explicit 3D Gaussians conditioned on texts. We innovatively organize disorganized 3D Gaussian points into a structured volumetric form, termed GaussianVolume, enabling feed-forward generation of 3D Gaussians via a coarse-to-fine generation pipeline.

To facilitate the fitting of high-quality GaussianVolumes for training, we propose a novel pruning and densifying strategy, i.e. Candidate Pool Strategy. Our proposed framework, GVGEN, demonstrates remarkable efficiency in generating 3D Gaussians from texts. Experimental results demonstrate the competitive capabilities of GVGEN in both qualitative and quantitative terms. This progress suggests potential extensions of our approach in tackling a broader spectrum of challenges within the field.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
2. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5855–5864 (2021)
3. Cao, A., Johnson, J.: Hexplane: A fast representation for dynamic scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 130–141 (2023)
4. Chang, J.H.R., Chen, W.Y., Ranjan, A., Yi, K.M., Tuzel, O.: Pointersect: Neural rendering with cloud-ray intersection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8359–8369 (2023)
5. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision. pp. 333–350. Springer (2022)
6. Chen, R., Chen, Y., Jiao, N., Jia, K.: Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. arXiv preprint arXiv:2303.13873 (2023)
7. 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)
8. Chen, Z., Wang, F., Liu, H.: Text-to-3d using gaussian splatting. arXiv preprint arXiv:2309.16585 (2023)
9. Cheng, Y.C., Lee, H.Y., Tulyakov, S., Schwing, A.G., Gui, L.Y.: Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4456–4465 (2023)
10. Deitke, M., Liu, R., Wallingford, M., Ngo, H., Michel, O., Kusupati, A., Fan, A., Laforte, C., Voleti, V., Gadre, S.Y., et al.: Objaverse-xl: A universe of 10m+ 3d objects. Advances in Neural Information Processing Systems **36** (2024)
11. 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)
12. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. Advances in neural information processing systems **34**, 8780–8794 (2021)
13. He, Z., Wang, T.: Openlrm: Open-source large reconstruction models. https://github.com/3DTopia/OpenLRM (2023)

14. Hong, Y., Zhang, K., Gu, J., Bi, S., Zhou, Y., Liu, D., Liu, F., Sunkavalli, K., Bui, T., Tan, H.: Lrm: Large reconstruction model for single image to 3d. arXiv preprint arXiv:2311.04400 (2023)
15. Huang, Z., Wen, H., Dong, J., Wang, Y., Li, Y., Chen, X., Cao, Y.P., Liang, D., Qiao, Y., Dai, B., et al.: Epidiff: Enhancing multi-view synthesis via localized epipolar-constrained diffusion. arXiv preprint arXiv:2312.06725 (2023)
16. Jain, A., Mildenhall, B., Barron, J.T., Abbeel, P., Poole, B.: Zero-shot text-guided object generation with dream fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 867–876 (2022)
17. Jun, H., Nichol, A.: Shap-e: Generating conditional 3d implicit functions. arXiv preprint arXiv:2305.02463 (2023)
18. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
19. Li, J., Tan, H., Zhang, K., Xu, Z., Luan, F., Xu, Y., Hong, Y., Sunkavalli, K., Shakhnarovich, G., Bi, S.: Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. arXiv preprint arXiv:2311.06214 (2023)
20. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. arXiv preprint arXiv:2301.12597 (2023)
21. Li, W., Chen, R., Chen, X., Tan, P.: Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d. arXiv preprint arXiv:2310.02596 (2023)
22. Liang, Y., Yang, X., Lin, J., Li, H., Xu, X., Chen, Y.: Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching (2023)
23. Liu, Y., Lin, C., Zeng, Z., Long, X., Liu, L., Komura, T., Wang, W.: Syncdreamer: Generating multiview-consistent images from a single-view image. arXiv preprint arXiv:2309.03453 (2023)
24. Liu, Z., Li, Y., Lin, Y., Yu, X., Peng, S., Cao, Y.P., Qi, X., Huang, X., Liang, D., Ouyang, W.: Unidream: Unifying diffusion priors for relightable text-to-3d generation. arXiv preprint arXiv:2312.08754 (2023)
25. Long, X., Guo, Y.C., Lin, C., Liu, Y., Dou, Z., Liu, L., Ma, Y., Zhang, S.H., Habermann, M., Theobalt, C., et al.: Wonder3d: Single image to 3d using cross-domain diffusion. arXiv preprint arXiv:2310.15008 (2023)
26. Luo, T., Rockwell, C., Lee, H., Johnson, J.: Scalable 3d captioning with pretrained models. Advances in Neural Information Processing Systems **36** (2024)
27. Melas-Kyriazi, L., Rupprecht, C., Vedaldi, A.: Pc2: Projection-conditioned point cloud diffusion for single-image 3d reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12923–12932 (2023)
28. 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)
29. Mohammad Khalid, N., Xie, T., Belilovsky, E., Popa, T.: Clip-mesh: Generating textured meshes from text using pretrained image-text models. In: SIGGRAPH Asia 2022 conference papers. pp. 1–8 (2022)
30. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
31. Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., Chen, M.: Point-e: A system for generating 3d point clouds from complex prompts. arXiv preprint arXiv:2212.08751 (2022)

32. Ntavelis, E., Siarohin, A., Olszewski, K., Wang, C., Gool, L.V., Tulyakov, S.: Autodecoding latent 3d diffusion models. Advances in Neural Information Processing Systems **36** (2024)
33. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988 (2022)
34. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
35. Ren, J., Pan, L., Tang, J., Zhang, C., Cao, A., Zeng, G., Liu, Z.: Dreamgaussian4d: Generative 4d gaussian splatting. arXiv preprint arXiv:2312.17142 (2023)
36. 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)
37. Shi, R., Chen, H., Zhang, Z., Liu, M., Xu, C., Wei, X., Chen, L., Zeng, C., Su, H.: Zero123++: a single image to consistent multi-view diffusion base model. arXiv preprint arXiv:2310.15110 (2023)
38. Shi, Y., Wang, P., Ye, J., Long, M., Li, K., Yang, X.: Mvdream: Multi-view diffusion for 3d generation. arXiv preprint arXiv:2308.16512 (2023)
39. Tang, J., Ren, J., Zhou, H., Liu, Z., Zeng, G.: Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. arXiv preprint arXiv:2309.16653 (2023)
40. Tang, Z., Gu, S., Wang, C., Zhang, T., Bao, J., Chen, D., Guo, B.: Volumediffusion: Flexible text-to-3d generation with efficient volumetric encoder. arXiv preprint arXiv:2312.11459 (2023)
41. Wang, Z., Lu, C., Wang, Y., Bao, F., Li, C., Su, H., Zhu, J.: Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. Advances in Neural Information Processing Systems **36** (2024)
42. Wu, G., Yi, T., Fang, J., Xie, L., Zhang, X., Wei, W., Liu, W., Tian, Q., Wang, X.: 4d gaussian splatting for real-time dynamic scene rendering. arXiv preprint arXiv:2310.08528 (2023)
43. Wu, Z., Wang, Y., Feng, M., Xie, H., Mian, A.: Sketch and text guided diffusion model for colored point cloud generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8929–8939 (2023)
44. Xu, D., Yuan, Y., Mardani, M., Liu, S., Song, J., Wang, Z., Vahdat, A.: Agg: Amortized generative 3d gaussians for single image to 3d. arXiv preprint arXiv:2401.04099 (2024)
45. Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., Neumann, U.: Point-nerf: Point-based neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5438–5448 (2022)
46. Yi, T., Fang, J., Wu, G., Xie, L., Zhang, X., Liu, W., Tian, Q., Wang, X.: Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. arXiv preprint arXiv:2310.08529 (2023)
47. Yin, Y., Xu, D., Wang, Z., Zhao, Y., Wei, Y.: 4dgen: Grounded 4d content generation with spatial-temporal consistency. arXiv preprint arXiv:2312.17225 (2023)
48. Yu, X., Guo, Y.C., Li, Y., Liang, D., Zhang, S.H., Qi, X.: Text-to-3d with classifier score distillation. arXiv preprint arXiv:2310.19415 (2023)
49. Zou, Z.X., Yu, Z., Guo, Y.C., Li, Y., Liang, D., Cao, Y.P., Zhang, S.H.: Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. arXiv preprint arXiv:2312.09147 (2023)

6 Implemantation Details

6.1 Stage 1: GaussianVolume Fitting

In this stage, we precisely fit each object using 96 uniformly rendered images, capturing various poses. The initial 72 images are rendered with camera poses uniformly distributed around a camera-to-world-center distance of 2.4 units. The remaining 24 images are rendered from a closer distance of 1.6 units to provide a more detailed view. We set the volume resolution at $N = 32$ and the spherical harmonics (SH) order at 0, resulting in each Gaussian point having a feature channel number $C = 14$.

The fitting volume is assumed to be a cube with a side length of 1 world unit, centered within the global coordinate system. Inside this cube, Gaussian points are uniformly distributed to ensure a comprehensive coverage of the volume. Initially, the offset for each Gaussian point $\Delta\mu$ is set to zero. To optimize each instance, we conduct training against a white background for 20,000 iterations. A densification strategy is employed from iteration 500 to 15,000, with subsequent operations executed every 100 iterations to incrementally enhance the model's density. After this densification phase, we periodically clip features to predefined ranges every 100 iterations to maintain consistency and prevent outliers. We refrain from resetting the opacity of each Gaussian point during the training process. This decision is made to avoid introducing instability into the model's learning and adaptation phases.

For weights selection in $\mathcal{L}_{fitting}$, we set $\lambda_1 = 0.8, \lambda_2 = 0.2, \lambda_3 = 20.0$, and the offsets threshold as 1.5 voxel distances. Mathematically, this is expressed as $\epsilon_{offsets} = \frac{1.5}{32-1}$, signifying the calculated distance between adjacent grid points in our defined volume space. For other parameters, we adhere to the default configurations in 3D Gaussian Splatting [18].

6.2 Stage 2: Text-to-3D Generation

Data Pre-processing In the generation phase, we first preprocess GaussianVolume data to improve the stability and convergence of the learning process.

First, we standardize the decomposition of the covariance matrix by imposing an ordering constraint on the scaling vector $s \in \mathbb{R}^3$ of each Gaussian, ensuring they are arranged in ascending order. This organization, while maintaining the orthogonality of the eigenvectors, does not modify the resultant covariance matrix. The rotation vector $q \in \mathbb{R}^4$ is also adjusted to align with the sequence of transformations introduced by scaling. Secondly, we represent quaternions using spherical coordinates. This conversion reduces the rotation vector q to a three-dimensional representation, enhancing the uniqueness and consistency of each Gaussian's parameters. By normalizing quaternions in this manner, we ensure that identical shapes are represented by identical parameters. Finally, we normalize each feature channel individually based on its mean and variance. This normalization process equilibrates the scales across different channels.

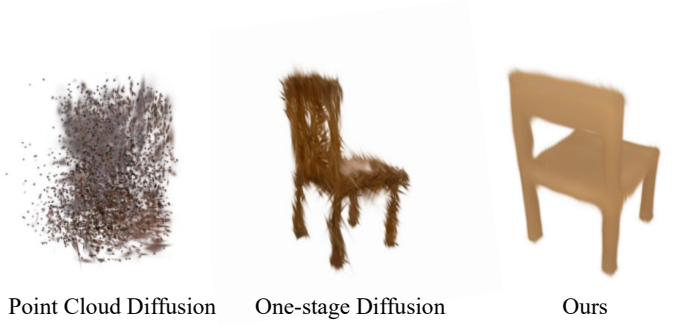


Fig. 8: Visual Results of Generated Assets in Explorational Experiments.

Model Architecture and Training In the generation phase of Gaussian Density Fields (GDF), we utilize a 3D U-Net modified from [9]. The text condition is 77×768 embeddings extracted with CLIP ViT-L/14 [34] text encoder, and it is injected into the 3D U-Net using cross-attention mechanisms. An additional MLP layer is employed to bridge the inter-modal gap. The architecture of the prediction model mirrors the 3D U-Net, with adjustments including a modification in the input channel number and the omission of timestep blocks.

When training the generative model predicting GDF, we only use MSE loss $\mathcal{L}_{3D}$ as the supervision. For training the prediction model, we set $\lambda = 0.2$ for the rendering loss $\mathcal{L}_{2D}$ through the whole process. Then, for the total loss $\mathcal{L} = \lambda_{3D}\mathcal{L}_{3D} + \lambda_{2D}\mathcal{L}_{2D}$, we set $\lambda_{3D} = 1.0, \lambda_{2D} = 0$ for the first 100 epochs, providing a robust initialization that focuses solely on 3D consistency. Subsequently, the weights are adjusted to $\lambda_{3D} = 0.2$ and $\lambda_{2D} = 0.8$, shifting the focus towards optimizing 2D rendering and further refining.

7 Explorational Experiments

In our early exploration, we investigated the efficacy of various generative models, initially training three unconditional models: a point cloud diffusion model, and a primal 3D U-Net-based diffusion model, and compared them with our full model. We specifically utilized 8 fitted GaussianVolumes belonging to the chair LVIS-category as our training data and trained each model for 4,000 epochs. The qualitative results are shown in Fig. 8.

Our initial foray involved adapting the point cloud diffusion model, inspired by prior work [27], to facilitate the generation of 3D Gaussians. Unfortunately, this approach struggled with convergence issues, yielding unsatisfactory results. The inherent limitations of point cloud diffusion models in capturing and generating the nuanced structures of 3D Gaussians became apparent, prompting us to explore alternative strategies. Employing a primal one-stage diffusion model offered a glimpse into generating basic 3D structures. However, this model predominantly produced coarse outputs characterized by spiny shapes, highlighting

Fig. 9: GSGEN [8] Results Initialized with Our Method and Point-E [31]. The left column represents rendering results initialized with different methods, and the right column stands for rendering results after optimization with GSGEN.

the need for a more nuanced generation technique to achieve plausible 3D geometry. Our full model, incorporating a coarse-to-fine generation pipeline, demonstrated significant improvements in generating 3D Gaussians. This approach not only simplified the generation process but also empowered the model to produce instances with more accurate and realistic 3D geometry. By sequentially refining the generated outputs, the coarse-to-fine pipeline effectively addresses the limitations observed in the earlier models, showcasing its superiority in generating complex 3D structures.

8 Application

To demonstrate the superiority of our method, we show GVGEN's capability to integrate with optimization-based methods, like GSGEN [8], for further refinement (see Fig. 9). Using the generated GaussianVolume[6] as initialization to

[6] We filter low-opacity Gaussian points for GSGEN initialization.

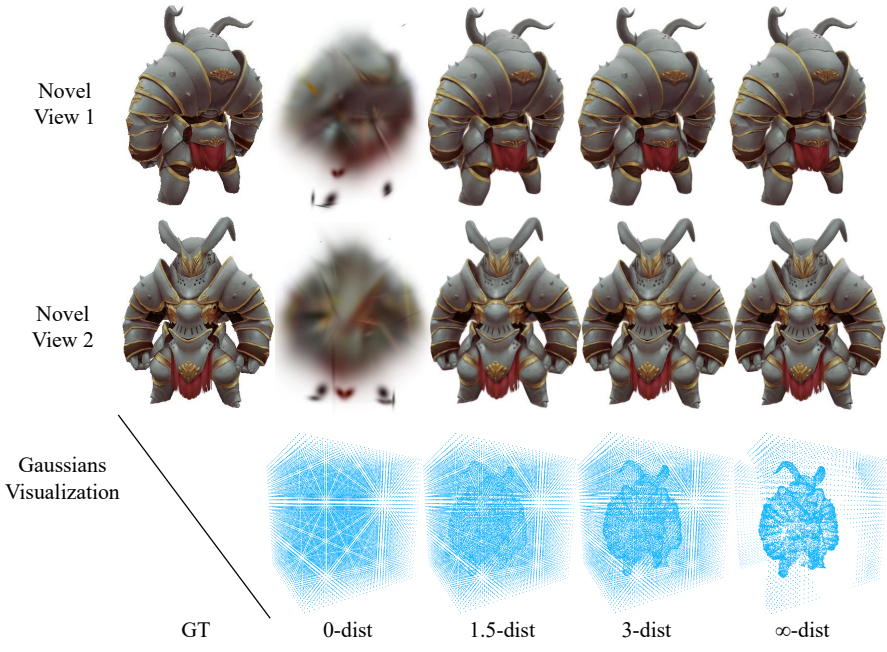


Fig. 10: Visual Results of Rendered Images and Positions of The Gaussian Center. The fitted assets are optimized with different offsets threshold $\epsilon_{offsets}$.

replace point clouds from Point-E with random colors. This shift enables GSGEN to further optimize 3D Gaussians, achieving better alignment with the text descriptions in both texture and geometry. This enhancement stems from avoiding the adverse impact of color attributes from Point-E on GSGEN's utilization, as the features produced by GVGEN are more compatible and beneficial for the optimization process.

9 Additional Ablation Studies on GaussianVolume Fitting Stage

9.1 Effects of offsets threshold $\epsilon_{offsets}$

In Tab. 3 and Fig. 10, we present the effects of varying the offset threshold, $\epsilon_{offsets}$ during GaussianVolume fitting. The term "0-dist" indicates the absence of an offsets term, whereas "∞-dist" denotes the omission of offsets regularization. Our observations reveal that minimal regularization leads to improved rendering performance. However, this comes at the cost of Gaussian points within the volume becoming more unstructured and deviating from grid points. Without this regularization, the offset term becomes overly flexible, making it challenging

Table 3: Quantitative results for different offsets threshold selection.

Metrics	PSNR ↑	SSIM ↑	LPIPS ↓
0-dist	18.189	0.859	0.176
1.5-dist	**30.437**	0.966	0.046
3-dist	30.395	0.969	0.038
∞-dist	30.007	**0.969**	**0.034**

Table 4: Quantitative Results for Different Settings of GaussianVolume Resolution.

Metrics	PSNR ↑	SSIM ↑	LPIPS ↓	Num of Gaussians	Time(s)
16-res	23.079	0.887	0.122	4,096	169
32-res	**30.437**	0.966	0.046	32,768	179
64-res	30.395	**0.972**	0.032	262,144	195
3DGS	29.789	0.969	**0.030**	175,532	191

for the network to learn effectively. To strike a balance between flexibility and maintaining a well-defined structure, we have selected $\epsilon_{offsets} = \frac{1.5}{32-1}$, equivalent to a 1.5 voxel distance, as our optimal threshold.

9.2 Effects of GaussianVolume Resolution

We present a comparison of rendering results and quantitative metrics in Tab. 4 and Fig. 11, utilizing the "Chunky knight" asset from the Objaverse dataset. An increase in resolution N-res of the GaussianVolume correlates to a higher number of Gaussian points, N^3. To accommodate the varying volume resolutions N, adjustments are made to the initialization and the offsets threshold, specifically $\epsilon_{offsets} = \frac{1.5}{N-1}$. The term "3DGS" denotes the process of fitting the object using the original 3D Gaussian Splatting [18], with iteration settings matched to ours and other parameters set to default. The analysis reveals that a greater number of Gaussian points leads to improved fitting quality under identical settings, albeit at the expense of increased memory usage. Notably, even when utilizing fewer Gaussian points, our method delivers comparable visual quality to the original 3DGS approach and achieves higher PSNR values in a structured representation. This underscores the efficiency and effectiveness of our method in reconstructing high-quality, memory-efficient representations.

10 Failure Cases

As illustrated in Fig. 12, some of the objects generated by our model suffer from blurred textures and imprecise geometry. This phenomenon largely stems from the relatively small size of our training dataset, which comprises around 46,000

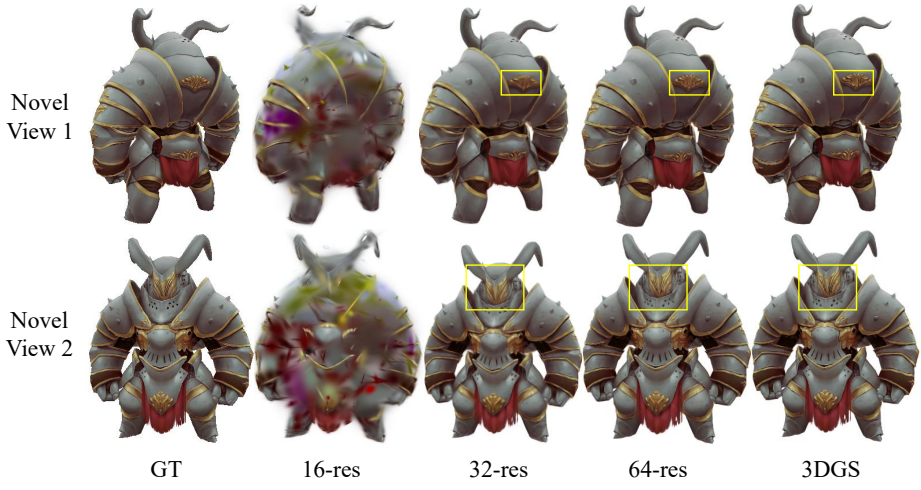


Fig. 11: Qualitative Comparisons among different GaussianVolume resolution settings and original 3DGS.

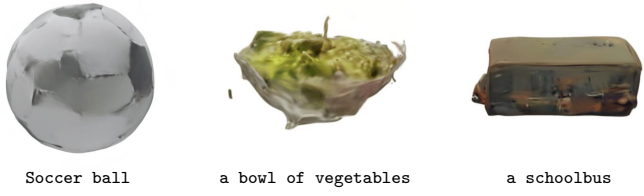


Fig. 12: Failure Cases.

instances. Such a dataset size limits the model's capacity to produce varied outputs in response to a broad spectrum of text inputs. In the future, our work will focus on two main avenues: improving the model architecture and enhancing the data quality. By addressing these aspects, we aim to scale up the model for application in large-scale scenarios, which is expected to improve the generation diversity and lead to better-rendered results.

11 Additional Qualitative Results

We provide more visual results in Fig. 13 and Fig. 14.

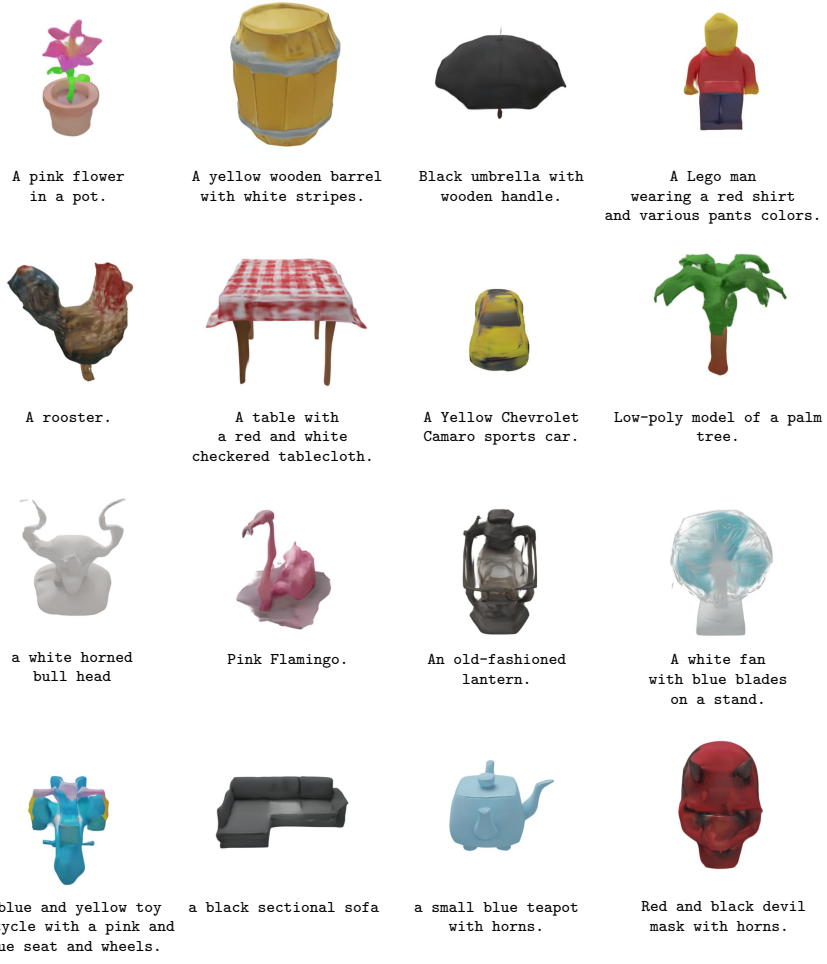


Fig. 13: More Qualitative Results.

an hourglass with blue sand

a reindeer with antlers.

a chocolate muffin on a plate.

White crane

a pink, purple, and blue children's chandelier.

Yellow rubber duck.

a light blue, plastic child's chair.

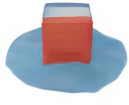
pink frosted donut with white sprinkles.

a wooden bedside side table.

a chessboard with checkered pattern.

a light blue ottoman with a metal frame.

Thor's wooden hammer



a red, blue, and white cube floating in water.

a pair of glasses.

a wooden stool with seat and legs.

a garlic bulb.

Fig. 14: More Qualitative Results.