Dragen3D: Multiview Geometry Consistent 3D Gaussian Generation with Drag-Based Control

JINBO YAN, Tencent, China
ALAN ZHAO, Tencent, China
YIXIN HU, Tencent America, USA

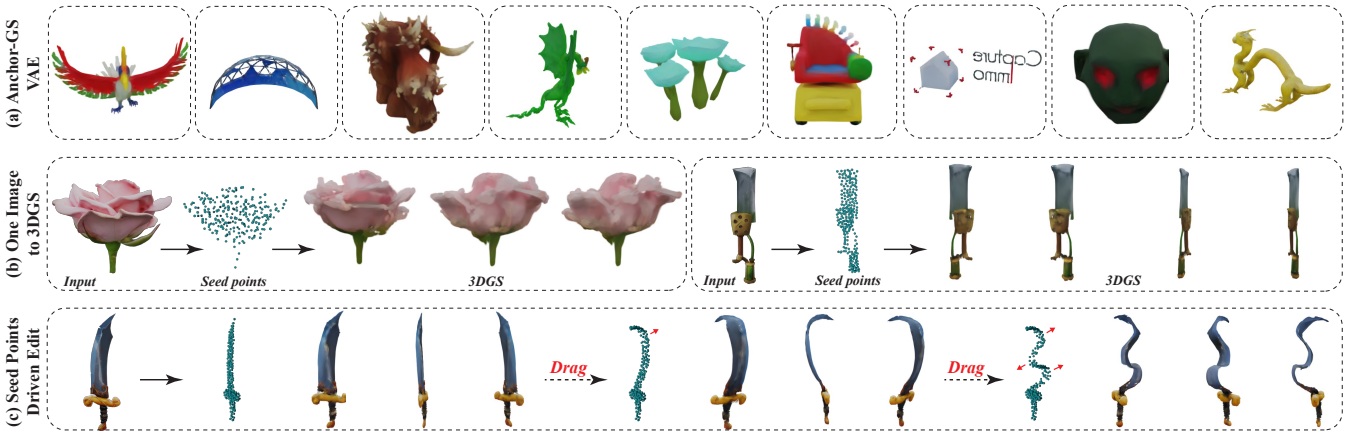

Fig. 1. We propose the Anchor-GS VAE and Seed-Point-Driven Generation strategy, which achieve high-quality 3DGS reconstructions (a) and multi-view geometry consistent single-image 3DGS generation (b). Additionally, we enable controllable 3DGS generation by allowing drag-based editing of the seed points (c).

Single-image 3D generation has emerged as a prominent research topic, playing a vital role in virtual reality, 3D modeling, and digital content creation. However, existing methods face challenges such as a lack of multi-view geometric consistency and limited controllability during the generation process, which significantly restrict their usability. To tackle these challenges, we introduce Dragen3D, a novel approach that achieves geometrically consistent and controllable 3D generation leveraging 3D Gaussian Splatting (3DGS). We introduce the Anchor-Gaussian Variational Autoencoder (Anchor-GS VAE), which encodes a point cloud and a single image into anchor latents and decode these latents into 3DGS, enabling efficient latent-space generation. To enable multi-view geometry consistent and controllable generation, we propose a Seed-Point-Driven strategy: first generate sparse seed points as a coarse geometry representation, then map them to anchor latents via the Seed-Anchor Mapping Module. Geometric consistency is ensured by the easily learned sparse seed points, and users can intuitively drag the seed points to deform the final 3DGS geometry, with changes propagated through the anchor latents. To the best of our knowledge, we are the first to achieve geometrically controllable 3D Gaussian generation and editing without relying on 2D diffusion priors, delivering comparable 3D generation quality to state-of-the-art methods.

1 INTRODUCTION

The field of 3D generation is highly popular at present and enjoys a wide range of applications in research and industry scenarios. However, compared to the traditional 3D modeling process where artists can directly interact and edit high-quality 3D models, achieving high geometric fidelity and direct editing within the 3D generation process is still an area awaiting in-depth study.

This challenge becomes even more pronounced in the context of 3D model generation from single-view images. For parts of the model not visible in the input image, the generated results may exhibit significant stylistic discrepancies from the visible regions, fail to achieve multi-view geometric consistency, or even appear unrealistic. To align with the creative aspirations and modeling requirements of artists, some studies, as discussed in Sec.2.4, have explored user control through input image modifications or predefined editing operations, these methods do not effectively address the aforementioned issues. To enhance the practical usability and quality of generated 3D models, we aim to develop a method that enables multi-view geometry consistent 3D generation, while allowing users to directly adjust and control the 3D shape during the generation process.

To this end, we propose an innovative approach, Dragen3D, utilizing sparse seed points for manipulating the object shape represented by 3D Gaussians (3DGS) and enhancing the multi-view geometry consistency within the 3D generation framework. To accomplish this, we train a Variational Autoencoder (VAE) that encodes the complex 3D information of an object into a compact latent space and accurately decodes it back into the 3D domain, while also supporting subsequent 3D generation in the latent space. Then, we introduce a module tasked with generating 3D seed points corresponding to the objects depicted in the input image. This ensures the geometric consistency of the seed points, thanks to the easy learning of their sparse distribution. Furthermore, a mapping

Authors' addresses: Jinbo Yan, Tencent, China, yanjbcool@gmail.com; Alan Zhao, Tencent, China, alantzhao@tencent.com; Yixin Hu, Tencent America, USA, yixinhu.yh@gmail.com.

module is incorporated to associate the information of seed points with the latent space of the VAE.

Our experiments show that DRAGEN3D produces multi-view geometry consistent 3D results as shown in Fig. 9. When the seed points undergo deformation, the corresponding latent codes are updated accordingly, enabling the generation of the final deformed 3D output upon decoding, as shown in Fig. 8

Our contributions can be summarized as follows:

- We propose the Anchor-GS VAE, which encodes 3D geometry and appearance into anchor latents and decodes them into 3DGS, making it easy to build while enabling efficient latent-space generation.
- We introduce a Seed-Driven Strategy that generates sparse seed points from a single image for geometric consistency and maps them to anchor latents via the Seed-Anchor Mapping Module.
- We design a Seed-Points-Driven Deformation module, enabling user-friendly geometric editing of 3DGS through drag operations on seed points.

We will open-source the implementation of our method and the trained models.

2 RELATED WORK

2.1 Neural Rendering and Gaussian Splatting

Radiance fields have become a popular research topic in 3D representation due to their powerful potential in 3D reconstruction and view synthesis. NeRF[Mildenhall et al. 2021], as a milestone work, made high-quality view synthesis possible. Its variants focus on improving rendering quality[Barron et al. 2021, 2022, 2023], training and inference speed [Fridovich-Keil et al. 2022; Hedman et al. 2021; Müller et al. 2022; Sun et al. 2022], and generalization ability [Chen et al. 2021; Johari et al. 2022; Wang et al. 2021; Yu et al. 2021]. Among them, 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] adopts a point-based radiance field, using 3D Gaussian primitives to represent scenes. Through anisotropic splatting and advanced rendering techniques, it enables high-quality reconstruction and real-time rendering. Some variants further enhance rendering quality and geometry [Huang et al. 2024; Lu et al. 2024; Yu et al. 2024a,b; Zhang et al. 2024b], offering the ability to represent both high-quality geometry and textures, which provides solutions for various tasks and applications, including 3D generation.

2.2 2D Diffusion Priors Based 3D Generation

Leveraging the high-quality generation capabilities of text-to-image diffusion models [Betker et al. 2023; Rombach et al. 2022; Saharia et al. 2022], some multiview diffusion models [Li et al. 2023a; Liu et al. 2023; Long et al. 2024; Shi et al. 2023a,b; Wang and Shi 2023] enable view synthesis based on text/image and view conditions, facilitating 3D generation from 2D diffusion priors. Some methods optimize 3D representations from these 2D priors using an SDS-loss-based approach [Liang et al. 2024; Poole et al. 2022; Shi et al. 2023b; Tang et al. 2023; Wang et al. 2024]or direct optimization [Tang et al. 2025b] from generated images. However, these methods are computationally expensive due to scene-by-scene optimization. Alternatively, other methods adopt a feed-forward [Chen et al. 2025; Li et al. 2023b; Liu et al. 2024c; Tang et al. 2025a; Xu et al. 2024a,b] or denoising process [Liu et al. 2024b; Wang et al. 2025; Xu et al. 2023] for 3D generation from 2D priors. For instance, LGM generates four views through a multiview diffusion model and then infers the corresponding 3DGS. These 2D-prior-based methods are constrained by inconsistencies in the multiview diffusion model, leading to misaligned 3D geometry and textures, and due to the stochastic nature of multiview image generation, they lack controlled generation.

2.3 End-to-end 3D Generative Models

Some methods [Hong et al. 2023; Tochilkin et al. 2024; Zou et al. 2024] directly generate 3D representations from a single image without relying on 2D diffusion priors. For example, TriplaneGaussian [Zou et al. 2024] creates a point cloud from a single image, combines it with triplane fusion for texture, and produces the final 3DGS, achieving state-of-the-art single-image 3D results. Other approaches [Gupta et al. 2023; Müller et al. 2023; Nichol et al. 2022; Zhang et al. 2023, 2024c; Zhao et al. 2024] use 3D diffusion models, like 3DShape2VecSet [Zhang et al. 2023], which encodes 3D information into a latent set and decodes it into a mesh, with diffusion models generating the latent set. Some approaches [He et al. 2025; Xiang et al. 2024; Zhang et al. 2024a; Zhou et al. 2024] also explore diffusion-based generation with Gaussian Splatting, such as GaussianCube [Zhang et al. 2024a], which constructs structured Gaussian representations and uses a 3D U-Net-based diffusion model to generate Gaussians from noise. While these methods model 3D data distribution well, they lack user-friendly control for generation and editing. In contrast, our model leverages a diffusion model to learn 3D information distribution without needing a 3DGS dataset, offering controllable generation through 3D space manipulation.

2.4 Editing in 3D Generative Models

To enable controllable 3D generation and editing, SketchDream [Liu et al. 2024a] allows users to modify the sketch and achieve edits using SDS optimization for vivid results. However, its controllability is limited as user modifications are made in 2D space, which may not produce the desired effect for unselected viewpoints. Interactive3D [Dong et al. 2024] directly edits 3DGS in 3D space using SDS optimization and predefined operations, converting the 3DGS representation into InstantNGP [Müller et al. 2022] with further refine. MVDrag3D [Chen et al. 2024] projects 3D-space drag operations onto multiview images, using 2D diffusion editing capabilities, and infers the edited 3DGS through LGM [Tang et al. 2025a], followed by SDS refinement. These methods offer a more user-friendly experience. However, all of these methods rely on 2D generative priors, which may lead to geometric inconsistencies (as discussed in Sec. 2.2), and require time-consuming optimization. In contrast, our method enables interactive manipulation of sparse seed points in 3D space, applying seed-point-driven deformation to modify the 3DGS without 2D priors or additional optimization, offering a more user-friendly editing experience.

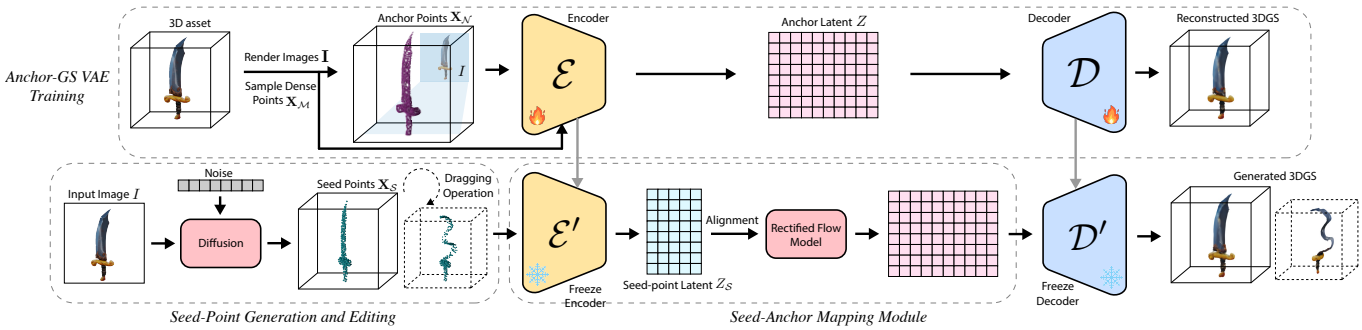

Fig. 2. Overview of the framework.

3 METHOD

3.1 Overview

Our method, DRANGEN3D, takes an image as input and generate a 3D object represented by 3D Guassians with multi-view geometric consistency, allowing user interaction of editing the geometry during the process. As illustrated in Fig. 2, we first train an Anchor-Gaussian (Anchor-GS) VAE that encodes complex 3D information into a latent space and decodes it into 3DGS, enabling subsequent generation in the latent space (Sec. 4). Then, we propose Seed-Point-Driven Controllable Generation module for 3D generation from a single image. This module starts with the generation of the rough initial geometry represented by a set of sparse surface points, named seed points, where we can apply the editing by deforming the seed points. After that, a mapping module is designed to map the (edited) seed point information to the latent space, which can be decode to 3DGS subsequently (Sec. 5).

3.2 Background

Gaussian Splatting. Gaussian splatting represents scenes as a collection of anisotropic 3D Gaussians. Each Gaussian primitive $\mathcal{G}_i$ is parameterized by a center $\mu \in \mathbb{R}^3$, opacity $\alpha \in \mathbb{R}$, color $c \in \mathbb{R}^{3(n+1)^2}$ which is represented by n-degree SH coefficients and 3D covariance matrix $\Sigma \in \mathbb{R}^{3\times3}$,which can be represented by scaling $s \in \mathbb{R}^3$ and rotation $r \in \mathbb{R}^4$.

During rendering, the 3D Gaussian is first projected onto 2D space. Given a view transformation matrix W, the 2D covariance matrix Σ' can be computed as : $\Sigma' = JW\Sigma W^T J^T$, where J is the Jacobian of the affine approximation of the projective transformation. Subsequently, the Gaussians covering a pixel are sorted based on depth. The color of the pixel is obtained using point-based alpha blending rendering:

$$c = \sum_{i=1}^{n} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_i) \tag{1}$$

Rectified Flow Model. The Rectified Flow Model [Lipman et al. 2022; Liu et al. 2022] has the capability to establish a mapping between two distributions, π_0 and π_1, making it well-suited for our task of mapping seed point latents to anchor latents. Given $x_0 \sim \pi_0$ and the corresponding $x_1 \sim \pi_1$, we can obtain $x(t) = (1-t)x_0 + tx_1$ at timestamp $t \in [0, 1]$ through linear interpolation. A vector field v_θ parameterized by a neural network is used to drive the flow from the source distribution π_0 to the target distribution π_1 by minimizing the conditional flow matching objective:

$$L(\theta) = E_{t,x_0,x_1,y} ||v_\theta(x_t, t, y) - (x_1 - x_0)|| \tag{2}$$

Here, $v_\theta(x_t, t, y)$ is the predicted flow at time t for a given point x_t, y refers to the image condition that guides the flow matching.

4 ANCHOR-BASED 3DGS VAE

We adopt an anchor-based approach to obtain 3D Gaussians, where the "anchor" refers to anchor points that are surface points capturing the main geometry of the object. We design and train an Anchor-Gaussian VAE that utilize Geometry-Texture Encoder $\mathcal{E}$ to encode geometry and appearance information of a 3D object into a set of fixed length latents, called anchor latents Z (Sec. 4.1). Subsequently, the Decoder $\mathcal{D}$ decodes these anchor latents into Gaussian primitives in a coarse-to-fine manner (Sec. 4.2). The encoder and decoder are trained together in an end-to-end manner, with the loss function (Sec. 4.3).

4.1 Geometry-Texture Encoder

The Geometry-Texture Encoder encodes the anchor points, the surface point cloud, and a set of rendered images of an object into a latent space. We obtain the anchor points $\mathbf{X}_\mathcal{N}$ of an object by sampling from the surface point cloud $\mathbf{X}_\mathcal{M} \in \mathbb{R}^{M\times3}$ of a 3D object using Farthest Point Sampling (FPS) method, which is similar to [Zhang et al. 2022, 2023]. Here $\mathcal{N} \subsetneq \mathcal{M}$, represents the index set of point clouds, with default settings of $|\mathcal{N}| = 2048$ and $|\mathcal{M}| = 4096$, and $\mathbf{X}_\mathcal{N} \in \mathbb{R}^{N\times3}$ denotes the sampled anchor points.

To encode the appearance information, we then project these anchor points onto the image feature plane $P_I \in \mathbb{R}^{H\times W\times C}$, which is encoded from the rendered image I of a known viewpoint: $\forall i \in \mathcal{N}$, $f_i = \Psi(\Pi_I(x_i), P_I)$, where $\Pi_I(x_i)$ represents the projection of x_i onto the image plane of I using the camera parameters of I, and Ψ denotes bilinear interpolation. The f_i and positional encoding of x_i represents the texture information and geometric of the i-th anchor.

To allow each anchor to capture more global information, we then input these features into two layers of Transformer blocks, which utilize point clouds $\mathbf{X}_\mathcal{M}$ and image tokens extracted from the

input image I to perform cross-attention:

$$\begin{aligned} Z^{'} &= \textbf{Transformer}_1(\{(\textbf{PE}(x_i); f_i)\}_{i\in\mathcal{N}} | \{\textbf{PE}(x_i)\}_{i\in\mathcal{M}}) \\ Z &= \textbf{Transformer}_2(Z^{'} | F_I) \end{aligned} \tag{3}$$

where Z represents the anchor latents obtained through encoding, **PE** represents the positional encoding, and (;) denotes concatenation along the channel dimension. $F_I \in \mathbb{R}^{N\times C}$ refers to the image feature tokens extracted by the Image Encoder from the input image I, where we use DINOv2[Oquab et al. 2023] for the feature extraction. And **Transformer**(|) denotes a Transformer block with cross-attention. All these encoding processes can be collectively represented by $\mathcal{E}$:

$$Z = \mathcal{E}(\mathbf{X}_\mathcal{N} \mid \mathbf{X}_\mathcal{M}, I) \tag{4}$$

After passing through the encoder $\mathcal{E}$, the anchor feature Z simultaneously encodes both geometric and texture information.

4.2 Decoder

In the Decoder, we adopt a coarse-to-fine approach to progressively obtain the Gaussian primitives, which enables higher-quality and more complete geometry. In the Encoder $\mathcal{E}$, both geometry and texture information are consolidated into a set of anchor latents Z, which is first decoded into a coarse geometry and then refined to recover more detailed geometry and corresponding textures.

Specifically, we apply Transformer with self-attention to Z:

$$Z^L = \textbf{Transformer}(Z) \tag{5}$$

Here, the Transformer block consists of L layers, and $\{Z^j\}_{j=1..L}$ represents the output at j-th layer of the Transformer with Z^L being the final output. We select the output from the k-th layer ($k = 2$ and $L = 8$ in default) as Z^{coarse}, and the output from the last layer as Z^{fine}. We first pass Z^{coarse} through a linear layer to reconstruct the anchors' spatial positions:

$$\hat{\mathbf{X}}_\mathcal{N} = \textbf{Linear}(Z^{\text{coarse}})$$

The symbol $\hat{\mathbf{X}}_\mathcal{N} \in \mathbb{R}^{N\times 3}$ represents the reconstructed positions of anchor points, which approximates the coarse geometry. Then, we assign m ($m = 8$ in default) Gaussian points to each anchor point. The positions of these Gaussian points are determined based on the anchor points' positions and a set of offsets derived from Z^{fine}. For the i-th anchor point, we have:

$$\begin{aligned} \{O_i^1, \ldots, O_i^m\} &= \textbf{Linear}(z_i^{\text{fine}}) \\ \{\mu_i^1, \ldots, \mu_i^m\} &= \hat{x}_i + \{O_i^1, \ldots, O_i^m\} \end{aligned} \tag{6}$$

where z_i^{fine} is the fine feature of i-th anchor and $\hat{x}_i$ represents the coarse position decoded for the i-th anchor point. Here, $\{O_i^j\}_{j=1..m}$ denotes the offsets of the j-th Gaussian point relative to the anchor position, and $\{\mu_i^j\}_{j=1..m}$ represents the final positions of the Gaussian points. This way, we obtain a set of Gaussian point positions with dimensions $\mathbb{R}^{N'\times 3}$, where $N' = N \times m$, representing the final fine-grained geometry.

For each Gaussian point, we can assign its other attributes by interpolating from its k ($k = 8$ in default) nearest anchors in the neighborhood:

$$z_i = \frac{\sum_{k\in\mathcal{N}(\mu_i)} e^{-d_k} z_k^{fine}}{\sum_{k\in\mathcal{N}(\mu_i)} e^{-d_k}} \tag{7}$$

where $\mathcal{N}(\mu_i)$ represents the set of neighboring anchor points of Gaussian point position μ_i, and d_k represents the Euclidean distance from μ_i to the reconstructed position of z_k^{fine}. Then we can use a linear layer to decode the attributes color c_i, opacity o_i, scale $scale_i$, and rotation rot_i of a Gaussian primitive z_i: $\{c_i, o_i, scale_i, rot_i\} = \textbf{Linear}(z_i)$.

4.3 Loss Function

Thanks to our efficient anchor-based representation design, the training of our VAE does not require pre-constructing a large-scale 3DGS dataset. Instead, we supervise the entire network using the rendering loss between the predicted rendered images and the ground truth images:

$$\mathcal{L}_{\text{rgb}} = \mathcal{L}_{\text{MSE}} + \lambda_s \mathcal{L}_{\text{SSIM}} + \lambda_l \mathcal{L}_{\text{lpips}} \tag{8}$$

where $\mathcal{L}_{\text{MSE}}$represents the pixel-wise Mean Squared Error (MSE) loss, $\mathcal{L}_{\text{SSIM}}$ represents the Structural Similarity Index (SSIM) loss and $\mathcal{L}_{\text{lpips}}$ represents the perceptual loss.

In addition, to obtain better geometry, we apply 3D point cloud supervision to both the reconstructed anchor point positions and the Gaussian point positions, comparing them with ground-truth points sampled from the 3D assets:

$$\mathcal{L}_{\text{points}} = \lambda_c \mathcal{L}_{\text{cd}} + \lambda_e \mathcal{L}_{\text{emd}} \tag{9}$$

Here, $\mathcal{L}_{\text{cd}}$ denotes the Chamfer Distance (CD), and $\mathcal{L}_{\text{emd}}$ represents the Earth Mover's Distance (EMD).

Finally, incorporating KL divergence regularization on the anchor latents produced by the encoder, the total loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{rgb}} + \mathcal{L}_{\text{points}} + \lambda_{\text{kl}} \mathcal{L}_{\text{KL}} \tag{10}$$

5 SEED-POINT-DRIVEN ANCHOR LATENT GENERATION AND EDITING

We adopt a Seed-Point-Driven generation approach to progressively obtain the anchor latents Z. First, we generate a sparse set of seed points $\mathbf{X}_\mathcal{S} \in \mathbb{R}^{S\times 3}$, which can be viewed as a rough representation of the geometry(Sec. 5.1). And then, through the Seed-Anchor Mapping module, we transform the sparse distribution of seed points into a dense distribution of anchor latents (Sec. 5.2). The Seed-Point-Driven strategy enables interactive geometric control of the generated 3DGS by simply *dragging* the seed points (Sec. 5.3).

This approach has the following advantages: (1)**Geometrically Consistent Generation**:We first learn a sparse set of seed points $\mathbf{X}_\mathcal{S}$ ($S = 256$), which ensures geometrically consistent 3D results due to the sparse nature of seed points and the ease of learning their distribution. (2) **Support for Geometric Editing**: By constructing the Seed-Anchor Mapping Module, we map seed points to their corresponding anchor latents. This decoupled design naturally supports geometric editing—modifying the seed points results in different anchor latents, enabling deformation of the 3DGS.

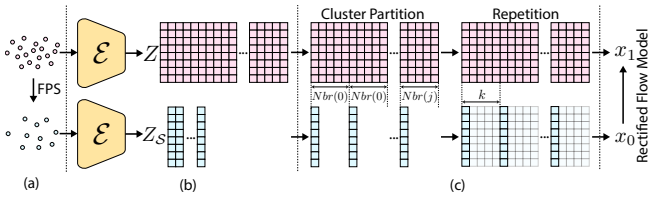

Fig. 3. Seed-Anchor Mapping Module: (a) We use FPS to establish a correspondence between Z and $\mathbf{X}_{\mathcal{S}}$. (b) Dimension Alignment: Encoding the seed points $\mathbf{X}_{\mathcal{S}}$ to obtain $Z_{\mathcal{S}}$, ensuring dimensional alignment with Z. (c) Token Alignment: Each token in the seed latent is treated as a center to partition the tokens of Z into $|\mathcal{S}|$ clusters. A repeat operation is then applied to the seed latents, achieving semantic and token count alignment between $Z_{\mathcal{S}}$ and Z.

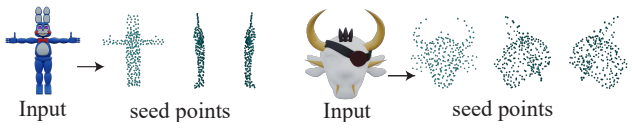

Fig. 4. Generation of Seed Points with Multiview Geometry Consistency

5.1 Seed Points Generation Module

Our goal is to generate a sparse set of seed points $\mathbf{X}_{\mathcal{S}}$ as a rough representation of the geometry from a single image input. To achieve this, we employ a diffusion model conditioned on the image to learn the distribution of $\mathbf{X}_{\mathcal{S}}$. Given the sparse nature of the seed points $\mathbf{X}_{\mathcal{S}}$, where $|\mathcal{S}| = 256$ in our settings, their distribution is relatively simple to learn directly, without the need for projection into a latent space. The results can be seen in Fig. 4. Specifically, we utilize the Rectified Flow model to map Gaussian noise to the seed point distribution π_S, treating the noise ϵ as x_0 and the data sample x_S as x_1.

5.2 Seed-Anchor Mapping Module

To use an input image I and a set of (deformed) seed points to control the generation of 3DGS, we need to derive the corresponding anchor latents Z. We model this task as a flow matching problem between two distributions and aim to solve it using the Rectified Flow Model, as shown in Fig. 3.

First, we need to establish a one-to-one correspondence between known samples from these two distributions. Specifically, for each anchor point set $\mathbf{X}_{\mathcal{N}}$, we apply Furthest Point Sampling (FPS) to downsample the anchor points to obtain the seed points $\mathbf{X}_{\mathcal{S}}$. That ensures for each Z, we can find a corresponding $\mathbf{X}_{\mathcal{S}}$.

Dimension Alignment. To construct the Rectified Flow model, the starting and targets must share the same dimensionality. Thus, we encode the seed points to align with the dimension of Z by passing $\mathbf{X}_{\mathcal{S}}$ through the freeze encoder $\mathcal{E}$ of Anchor-GS VAE:

$$Z_{\mathcal{S}} = \mathcal{E}(\mathbf{X}_{\mathcal{S}}|\mathbf{X}_{\mathcal{S}}, I) \tag{11}$$

Here, $Z_{\mathcal{S}}$ represents the encoded latents of the seed points. This allows us to simplify the problem into a mapping from $Z_{\mathcal{S}}$ to corresponding Z. Using the same pretrained encoder with Z, the $Z_{\mathcal{S}}$ distribution becomes better aligned with the target anchor latents Z, providing valuable information about the alignment between the points and the image.

Token Alignment. To establish flow matching between $Z_{\mathcal{S}}$and Z, we need to ensure that both samples contain the same number of tokens, and each token in the two samples corresponds semantically. Unlike [Fischer et al. 2023], which performs 2D grid-based upsampling on images, we cannot simply upsample seed points to match the target size while maintaining semantic correspondence between the points, as our latents are unordered.

To address this, we propose a cluster-based token alignment strategy. Each token in the latents retains the geometric information of the encoded points, allowing us to partition the latents into clusters based on their spatial positions. Specifically, for each token in the seed latents, we identify its neighborhood in the anchor latents using:

$$\forall i \in \mathcal{S}, \quad \text{KNN}(x_i) = \{x_j\}_{j \in \text{Nbr}(i)} \tag{12}$$

Here, x_j, x_i represent the encoded position of $z_j \in Z$ and $z_i \in Z_{\mathcal{S}}$, respectively, while $\text{Nbr}(i)$ denotes the index set of the k-nearest neighbors around z_i. This partitions Z partition into $|\mathcal{S}|$ clusters, where the tokens in each cluster $\text{Nbr}(i)$ are semantically similar to z_i , leveraging their spatial proximity. After establishing the semantic similarity between each token in $Z_{\mathcal{S}}$ and the corresponding cluster of tokens in Z, we simply repeat tokens in $Z_{\mathcal{S}}$ to ensure numerical equivalence. At this point, the alignment results between Z and $Z_{\mathcal{S}}$ can serve as the start point $x_0$and target x_1 for the Rectified Flow model, with the same number of tokens and semantic correspondence.

Model Architecture and Details. We implement the model using Transformer blocks, with image conditions serving as the key and value in the cross-attention, and inject timestamps via the adaptive layer norm (adaLN) block as described in [Peebles and Xie 2023]. With token alignment, the input tokens x_t are clustered and exhibit spatial similarity within each cluster. Therefore, we can downsample and then upsample within each cluster to reduce computational complexity, skipping connections to transfer detailed information. Similar to [Deng et al. 2024; Fischer et al. 2023], we apply noise augmentation to the start point x_0, which enhances the stability of our training process. Additionally, since the seed points used during inference are generated and subject to various edits, this noise augmentation helps our model generalize to a wider variety of seed points. We apply a cosine schedule for noise augmentation at 150 timesteps during both training and evaluation.

5.3 Seed-Points-Driven Deformation

Thanks to our Seed-Anchor Mapping module, the mapping begins with seed points—a sparse set of points that guide and control the overall 3D geometry generation. By adjusting the positions of these seed points, we can intuitively generate various desired geometries, making the editing process flexible and precise. The discrete nature of point clouds enables effective application of drag-style editing. Additionally, mature 3D tools like Blender[Community 2018] already support such operations on point clouds in 3D space, providing an intuitive and user-friendly editing experience.

Specifically, for the initial seed points $X_{\mathcal{S}}$ and their corresponding $Z_{\mathcal{S}}$, we apply drag-style editing to the seed points, resulting in $X_{\hat{\mathcal{S}}}$. We then encode $X_{\hat{\mathcal{S}}}$ to obtain $\hat{Z}_{\mathcal{S}}$, using Eq. 11. During encoding, we continue to use the projected features obtained from the projection of $X_{\mathcal{S}}$ onto the input image to preserve the correspondence between geometry and texture.

To preserve the consistency of these unedited regions, we introduce a mask to ensure their invariance:

$$Z^*_{\mathcal{S}} = mask \odot Z_{\mathcal{S}} + (1 - mask) \odot \hat{Z}_{\mathcal{S}} \tag{13}$$

In this equation, the mask is a Boolean vector indicating whether a point remains unchanged. With this, $Z^*_{\mathcal{S}}$ serves as the new seed latents. By applying the same alignment operation and Seed-Anchor Mapping module, we derive the corresponding anchor latents from the dragged seed points, which are then decoded into the deformed 3DGS. This process ensures that the dragged points remain aligned with their original texture while maintaining consistency in the unedited regions.

6 EXPERIMENTS

6.1 Implementation Details

Datasets. We use the Multiview Rendering Dataset [Qiu et al. 2023; Zuo et al. 2024] based on Objaverse [Deitke et al. 2022] for training. The dataset includes 260K objects, with 38 views rendered for each object, with a resolution of 512×512. To obtain the surface point clouds, we transform the 3D models according to the rendering settings, filter out those that are not aligned with the rendered images, and use Poisson sampling method[Yuksel 2015] to sample the surface. We randomly split the final processed data into training and testing sets, with the training dataset consisting of 200K objects. We conduct our in-domain evaluation using the test set from Objaverse, which includes 2,000 objects. To assess our model's cross-domain capabilities, we evaluate it on the Google Scanned Objects (GSO) [Downs et al. 2022]dataset, which contains 1,030 real-world scanned 3D models, with 32 views rendered for each model on a spherical surface.

Network. In our implementation, the anchor latents have a fixed length of 2048 and a dimension of 8. The model dimension in our transformer blocks is 512, with each transformer block comprising two attention layers and a feed-forward layer, following the design in [Zou et al. 2024]. The Anchor-GS VAE consists of two transformer blocks in the encoder and eight transformer blocks in the decoder. The Seed-Anchor Mapping Module is implemented using 24 transformer blocks, with 4 blocks for downsampling and 4 blocks for upsampling. Similarly, the Seed Points Generation Module is implemented with 24 transformer blocks. Leveraging the sparsity of seed points, we directly learn their distribution without requiring a VAE. The image conditioning in our model is extracted using DINOv2[Oquab et al. 2023].

Training Details. For training the Anchor-GS VAE, we randomly select 8 views per object, using one view as the input and all 8 views as ground truth images for supervision. The loss weights are set as $\lambda_s = 1$, $\lambda_l = 1$, $\lambda_c = 1$, $\lambda_e = 1$, and $\lambda_{KL} = 0.001$. We train the Anchor-GS VAE on a subset of our collected dataset containing approximately 40K objects, using a batch size of 128 on 8 A100 GPUs (40GB) for 24K steps. The Seed-Anchor Mapping Module is trained on the full dataset with a batch size of 1280 on 64 V100 GPUs(32GB) for 20K steps. The Seed Points Generation Module is trained on 48 V100 GPUs (32GB) for 54K steps. We use the AdamW optimizer with an initial learning rate of 4×10^{-4}, which is gradually reduced to zero using cosine annealing during training. The sampling steps for both the Seed-Anchor Mapping Module and the Seed Points Generation Module are set to 50 during inference.

Baseline. We compared our method with previous SOTA 3DGS generation models in the single-image input setting. LGM and LaRA rely on 2D multi-view diffusion priors to obtain multi-view images, which are then used to generate the output 3DGS in a feed-forward manner, as described in 2.2. TriplaneGS [Zou et al. 2024] does not require a 2D diffusion prior, directly generating 3DGS from a single input image, as outlined in 2.3. Both of them achieving SOTA performance. For each compared method, we use the official models and provided weights and ensure careful alignment of the camera parameters.

6.2 Results of VAE Reconstruction

In Fig. 7, we present the results of our Anchor-GS VAE. Given point clouds and a single image, our Anchor-GS VAE achieves high-quality reconstructions with detailed geometry and textures.

6.3 Results of 3D Generation

Metrics. Following previous works [Chen et al. 2025; Zou et al. 2024], we use peak signal-to-noise ratio (PSNR), perceptual quality measure LPIPS, and structural similarity index (SSIM) as evaluation metrics to assess different aspects of image similarity between the predicted and ground truth. Additionally, we report the time required to infer a single 3DGS. We use a single image as input and evaluate the 3D generation quality using all available views as testing views to compare our method with others, all renderings are performed at a resolution of 512.

Tab. 1 presents the quantitative evaluation results of our method compared to previous SOTA methods on the Objaverse and GSO datasets, along with qualitative results shown in Fig. 9. The multi-view diffusion model used in LGM tend to produce more diverse but uncontrollable results, and lacks precise camera pose control. As a result, it fails in our dense viewpoints evaluation, achieving PSNR scores of 12.76 and 13.81 on the Objaverse and GSO test sets, respectively.

As shown in Tab. 1, LGM and LaRa, influenced by the multi-view inconsistency of 2D diffusion models, achieve relatively lower scores in our dense viewpoint evaluation. In contrast, our method achieves the best results across both datasets, with only a slight overhead in inference time.

Fig. 9 presents the first six rows from the Objaverse dataset and the last three rows from the GSO dataset. All methods are compared using the same camera viewpoints. For the Objaverse dataset, the rendering viewpoints are the left and rear views relative to the input viewpoint, while for the GSO dataset, the views are selected to showcase the object as completely as possible. Compared to methods using 2D diffusion priors, such as LGM and LaRa, our method

demonstrates better multi-view geometric consistency, while the former tends to generate artifacts or unrealistic results in our displayed views. Compared to TGS, our method learns the 3D object distribution more effectively, resulting in more geometrically consistent multi-view results, such as the sharp feature in the left view in the first knife case.

6.4 Editing Results Based on Drag

As shown in Fig. 8, our method enables Seed-Points-Driven Deformation. Starting with generated seed points from the input image, the sparse nature of the seed points allows for easy editing using 3D tools (e.g., Blender[Community 2018]) with a few drag operations. The edited 3DGS can then be obtained within 2 seconds.

6.5 Ablation Study

Seed Points Generation. We employ a Rectified Flow model to generate seed points conditioned on a single input image. Owing to the sparsity of the seed points, the flow model is easier to train and effectively learns the distribution of the seed points. However, we also explored an alternative implementation using a transformer-based feed-forward approach, where point clouds are generated from learnable embeddings in a single pass, as in [Zou et al. 2024]. As demonstrated in Fig. 5, the feed-forward approach struggles to capture the true distribution of seed points and fails to produce satisfactory results in regions not visible in the input image.

Dimension Alignment. To match the dimension of the starting and target points in the Seed-Anchor Mapping Module, we encode the seed points using the Anchor-GS VAE encoder (Eq. 11). This process brings their distributions closer, reducing learning difficulty and reliance on image conditions. To validate this method, we conducted experiments by replacing the encoding approach with positional encoding . When using positional encoding, the Seed-Anchor Mapping overly relied on the image condition, neglecting the contribution of the seed points and failing to enable seed-driven 3DGS deformation.

Token Alignment. We ensure token alignment in Flow Matching by organizing tokens around seed points, followed by cluster-based partition and repetition. To evaluate its effectiveness, we conducted two ablation experiments, as shown in Tab. 2. In the *No-cluster+No-repetition* setting, we omitted the clustering step, aligning only the corresponding seed and anchor latents while filling unmatched portions with noise. This also prevented cluster-based downsampling in the Flow Model, resulting in doubled memory consumption. In the *No-cluster* setting, we repeated the seed latents to match the number of anchor latents but left them unordered, leading to disorganized token matching. As shown in Tab. 2, on a 40K subset with the same number of training steps, the absence of token alignment significantly degraded Flow Matching performance, resulting in inaccurate correspondences.

7 CONCLUSIONS

In this paper, we present Dragen3D, a framework for multi-view geometry consistent single-image 3DGS generation with drag-based editing. We propose the Anchor-GS VAE, which encodes 3D geometry and texture into anchor latents and decodes them into 3DGS.

Table 1. Quantitative evaluation of our method compared to previous work. † achieves relatively lower PSNR values in the evaluation, so we display the results in Sec. 6.3.

Method	Objaverse[Deitke et al. 2022]			GSO[Downs et al. 2022]			Time(s)
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS ↓	
LGM†[Tang et al. 2025a]	-	0.836	0.211	-	0.833	0.21	4.82
LaRa[Chen et al. 2025]	16.57	0.860	0.174	15.98	9.869	0.162	9.50
TriplaneGS[Zou et al. 2024]	18.80	0.883	0.143	19.84	0.900	0.120	0.70
Ours	20.92	0.896	0.120	20.52	0.904	0.1122	4.71

Table 2. Ablation about token alignment

	PSNR↑	SSIM↑	LPIPS↓
No-cluster+No-repetition	18.84	0.877	0.141
No-cluster	19.20	0.876	0.142
ours-full	19.94	0.881	0.134

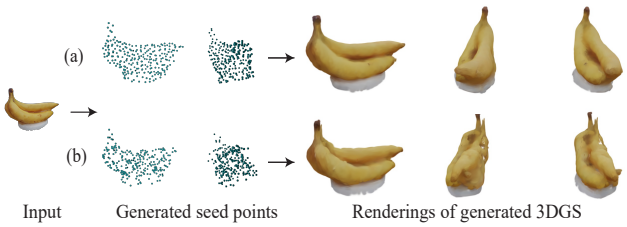

Fig. 5. Ablation study about different seed points geneartion methods: (a) using our method, (b) using Transformer.

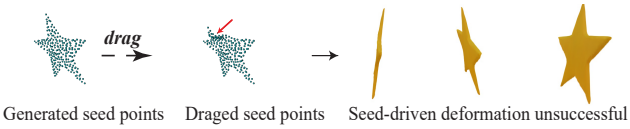

Fig. 6. Without Dimension Alignment, seed-points-driven deformation fails

Combining seed-point generation from single image, user-interacted seed-point editing, and seed-to-anchor-latent mapping, we are able to generate and control the final output 3DGS. Evaluations across multiple datasets demonstrate that Dragen3D achieves state-of-the-art 3DGS quality from single images. However, our method has room for improvement. First, incorporating 3D appearance editing based on prompts could be an interesting direction to explore, especially when integrated with existing multimodal large models. Additionally, the quality and quantity of training data limit our model's capabilities, which can be further improved with more computational resources.

REFERENCES

Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5855–5864.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of*

the IEEE/CVF conference on computer vision and pattern recognition. 5470–5479.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. 2023. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 19697–19705.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. 2023. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf* 2, 3 (2023), 8.

Anpei Chen, Haofei Xu, Stefano Esposito, Siyu Tang, and Andreas Geiger. 2025. Lara: Efficient large-baseline radiance fields. In *European Conference on Computer Vision*. Springer, 338–355.

Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. 2021. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF international conference on computer vision*. 14124–14133.

Honghua Chen, Yushi Lan, Yongwei Chen, Yifan Zhou, and Xingang Pan. 2024. Mv-Drag3D: Drag-based Creative 3D Editing via Multi-view Generation-Reconstruction Priors. *arXiv preprint arXiv:2410.16272* (2024).

Blender Online Community. 2018. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam. http://www.blender.org

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. 2022. Objaverse: A Universe of Annotated 3D Objects. *arXiv preprint arXiv:2212.08051* (2022).

Ken Deng, Yuanchen Guo, Jingxiang Sun, Zixin Zou, Yangguang Li, Xin Cai, Yanpei Cao, Yebin Liu, and Ding Liang. 2024. DetailGen3D: Generative 3D Geometry Enhancement via Data-Dependent Flow. *arXiv preprint arXiv:2411.16820* (2024).

Shaocong Dong, Lihe Ding, Zhanpeng Huang, Zibin Wang, Tianfan Xue, and Dan Xu. 2024. Interactive3D: Create What You Want by Interactive 3D Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4999–5008.

Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. 2022. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2553–2560.

Johannes S Fischer, Ming Gui, Pingchuan Ma, Nick Stracke, Stefan A Baumann, and Björn Ommer. 2023. Boosting Latent Diffusion with Flow Matching. *arXiv preprint arXiv:2312.07360* (2023).

Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. 2022. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5501–5510.

Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 2023. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371* (2023).

Xianglong He, Junyi Chen, Sida Peng, Di Huang, Yangguang Li, Xiaoshui Huang, Chun Yuan, Wanli Ouyang, and Tong He. 2025. Gvgen: Text-to-3d generation with volumetric representation. In *European Conference on Computer Vision*. Springer, 463–479.

Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. 2021. Baking Neural Radiance Fields for Real-Time View Synthesis. *ICCV* (2021).

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. 2023. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400* (2023).

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2024. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 Conference Papers*. 1–11.

Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. 2022. Geonerf: Generalizing nerf with geometry priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18365–18375.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023). https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. 2023b. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. *arXiv preprint arXiv:2311.06214* (2023).

Weiyu Li, Rui Chen, Xuelin Chen, and Ping Tan. 2023a. Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d. *arXiv preprint arXiv:2310.02596* (2023).

Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. 2024. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6517–6526.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. 2022. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747* (2022).

Feng-Lin Liu, Hongbo Fu, Yu-Kun Lai, and Lin Gao. 2024a. Sketchdream: Sketch-based text-to-3d generation and editing. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–13.

Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. 2024b. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10072–10083.

Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. 2024c. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems* 36 (2024).

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. 2023. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*. 9298–9309.

Xingchao Liu, Chengyue Gong, and Qiang Liu. 2022. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003* (2022).

Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. 2024. Wonder3d: Single image to 3d using cross-domain diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9970–9980.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. 2024. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20654–20664.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulo, Peter Kontschieder, and Matthias Nießner. 2023. Diffrf: Rendering-guided 3d radiance field diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4328–4338.

Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant neural graphics primitives with a multiresolution hash encoding. *ACM transactions on graphics (TOG)* 41, 4 (2022), 1–15.

Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. 2022. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751* (2022).

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. 2023. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193* (2023).

William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4195–4205.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. 2022. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988* (2022).

Lingteng Qiu, Guanying Chen, Xiaodong Gu, Qi zuo, Mutian Xu, Yushuang Wu, Weihao Yuan, Zilong Dong, Liefeng Bo, and Xiaoguang Han. 2023. RichDreamer: A Generalizable Normal-Depth Diffusion Model for Detail Richness in Text-to-3D. *arXiv preprint arXiv:2311.16918* (2023).

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems* 35 (2022), 36479–36494.

Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. 2023a. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110* (2023).

Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. 2023b. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512* (2023).

Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2022. Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction. In *CVPR*.

Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. 2025a. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *European Conference on Computer Vision*. Springer, 1–18.

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. 2023. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653* (2023).

Shitao Tang, Jiacheng Chen, Dilin Wang, Chengzhou Tang, Fuyang Zhang, Yuchen Fan, Vikas Chandra, Yasutaka Furukawa, and Rakesh Ranjan. 2025b. Mvdiffusion++: A dense high-resolution multi-view diffusion model for single or sparse-view 3d object

reconstruction. In *European Conference on Computer Vision*. Springer, 175–191.

Dmitry Tochilkin, David Pankratz, Zexiang Liu, Zixuan Huang, Adam Letts, Yangguang Li, Ding Liang, Christian Laforte, Varun Jampani, and Yan-Pei Cao. 2024. Triposr: Fast 3d object reconstruction from a single image. *arXiv preprint arXiv:2403.02151* (2024).

Peng Wang and Yichun Shi. 2023. Imagedream: Image-prompt multi-view diffusion for 3d generation. *arXiv preprint arXiv:2312.02201* (2023).

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. 2021. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4690–4699.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. 2024. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems* 36 (2024).

Zhengyi Wang, Yikai Wang, Yifei Chen, Chendong Xiang, Shuo Chen, Dajiang Yu, Chongxuan Li, Hang Su, and Jun Zhu. 2025. Crm: Single image to 3d textured mesh with convolutional reconstruction model. In *European Conference on Computer Vision*. Springer, 57–74.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. 2024. Structured 3D Latents for Scalable and Versatile 3D Generation. *arXiv preprint arXiv:2412.01506* (2024).

Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. 2024a. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191* (2024).

Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. 2024b. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. *arXiv preprint arXiv:2403.14621* (2024).

Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, et al. 2023. Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model. *arXiv preprint arXiv:2311.09217* (2023).

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. 2021. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4578–4587.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. 2024a. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19447–19456.

Zehao Yu, Torsten Sattler, and Andreas Geiger. 2024b. Gaussian Opacity Fields: Efficient Adaptive Surface Reconstruction in Unbounded Scenes. *ACM Transactions on Graphics* (2024).

Cem Yuksel. 2015. Sample Elimination for Generating Poisson Disk Sample Sets. *Comput. Graph. Forum* 34, 2 (May 2015), 25–32. https://doi.org/10.1111/cgf.12538

Bowen Zhang, Yiji Cheng, Jiaolong Yang, Chunyu Wang, Feng Zhao, Yansong Tang, Dong Chen, and Baining Guo. 2024a. GaussianCube: Structuring Gaussian Splatting using Optimal Transport for 3D Generative Modeling. *arXiv preprint arXiv:2403.19655* (2024).

Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiaoxiao Long, and Ping Tan. 2024b. RaDe-GS: Rasterizing Depth in Gaussian Splatting. *arXiv preprint arXiv:2406.01467* (2024).

Biao Zhang, Matthias Nießner, and Peter Wonka. 2022. 3dilg: Irregular latent grids for 3d generative modeling. *Advances in Neural Information Processing Systems* 35 (2022), 21871–21885.

Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 2023. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–16.

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. 2024c. CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–20.

Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. 2024. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. *Advances in Neural Information Processing Systems* 36 (2024).

Junsheng Zhou, Weiqi Zhang, and Yu-Shen Liu. 2024. Diffgs: Functional gaussian splatting diffusion. *arXiv preprint arXiv:2410.19657* (2024).

Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. 2024. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10324–10335.

Qi Zuo, Xiaodong Gu, Yuan Dong, Zhengyi Zhao, Weihao Yuan, Lingteng Qiu, Liefeng Bo, and Zilong Dong. 2024. High-Fidelity 3D Textured Shapes Generation by Sparse Encoding and Adversarial Decoding. In *European Conference on Computer Vision*.

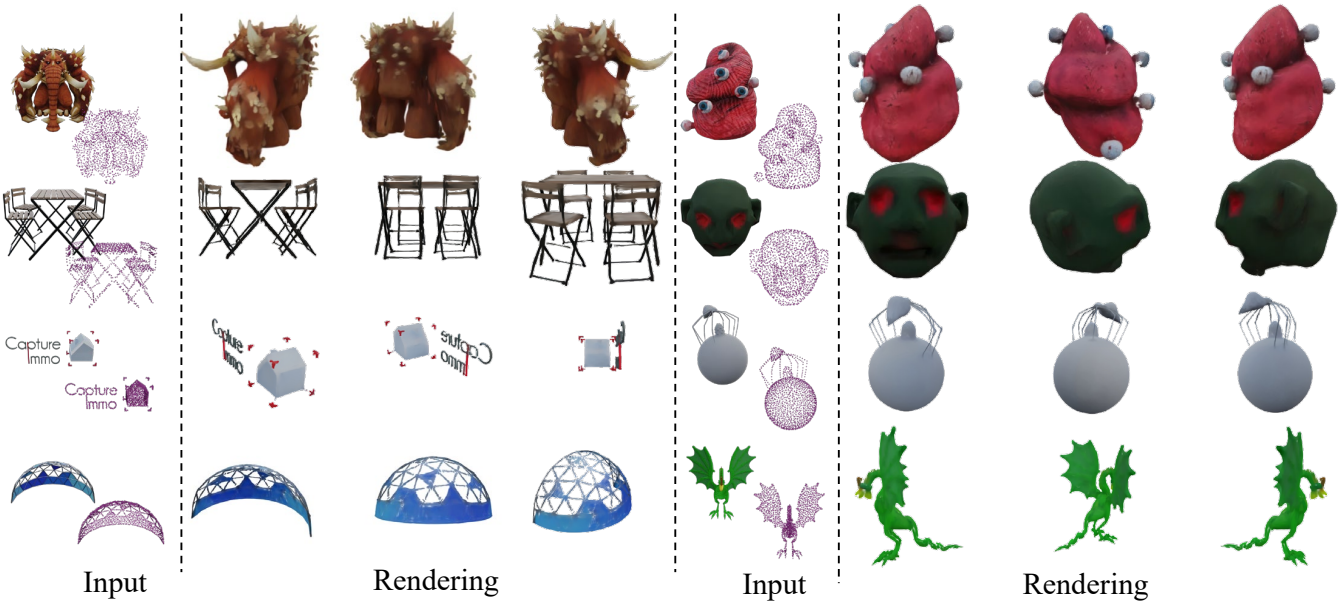

Fig. 7. The Reconstruction results of Anchor-GS VAE

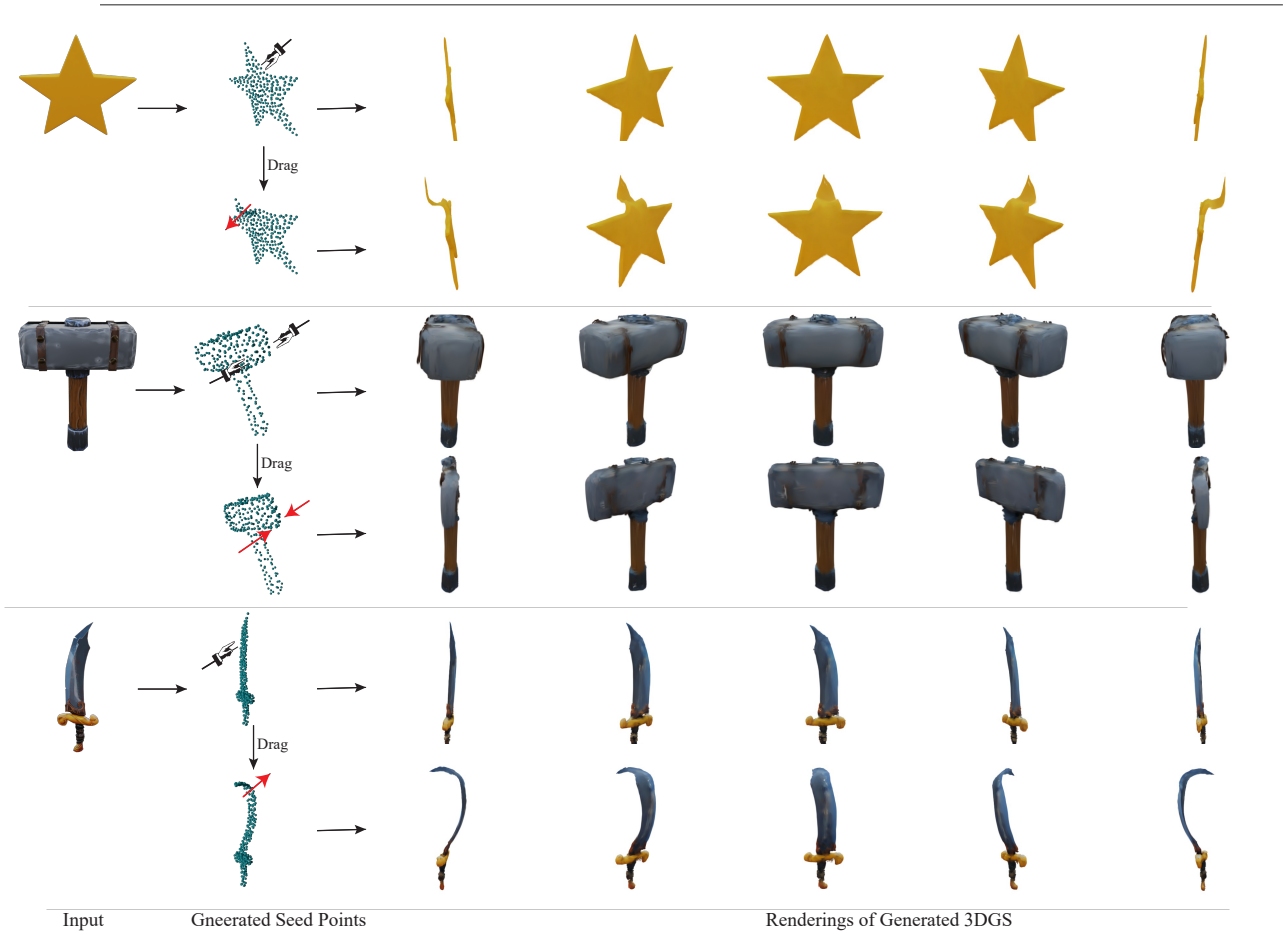

Fig. 8. Results of drag-based editing: By performing a limited number of drag edits on the seed points, we achieve finely controlled, edited results.

Fig. 9. Comparison of single-image to 3DGS generation: Our method achieves more multi-view geometrically consistent results.