GET3DGS: Generate 3D Gaussians Based on Points Deformation Fields

Haochen Yu, Weixi Gong, Jiansheng Chen, *Senior Member, IEEE,* and Huimin Ma, *Senior Member, IEEE*

***Abstract*—The 3D Gaussian Splatting method has recently shown significant advancements in rendering speed and scene composition quality, enhancing its industrial applications and boosting the demand for 3D Gaussian asset generation. However, existing mature 3D generation technologies predominantly rely on implicit representations, which often struggle to balance geometric quality with editability. The production of 3D Gaussian assets generally involves diffusion models that require a dual-stage process of reconstruction and generation, resulting in substantial training and inference costs. To overcome these challenges, we introduce GET3DGS, an innovative approach that combines 3D-aware GANs with 3D Gaussian Splatting representations. This method facilitates the manipulation of the physical attributes of 3D Gaussians, such as geometry and texture, via point deformation fields. Offering faster inference speeds and end-to-end training capabilities, our model outperforms existing diffusion model-based methods. By deriving high-quality Gaussian point cloud geometric representations from 2D images, our approach reduces material accumulation costs and produces data compatible with 3D Gaussian rendering engines. We have evaluated the generative performance of our model on ShapeNet and OmniObject3D and demonstrate competitive results in terms of image and geometric quality relative to previous methods.**

***Index Terms*—3D Gaussian Splatting, 3D generation, Differentiable rendering.**

I. Introduction

GENERATING 3D assets is crucial in constructing the digital world, and it is essential to minimize the cost of acquiring these assets. Currently, the acquisition of 3D assets primarily involves reconstruction techniques based on foreground-background disentangling methods, physical and mathematical approaches, and 3D generation methods.

Recent advancements in scene synthesis primarily depend on methods that separate the reconstruction of dynamic and static objects, facilitating scene editing and synthesis from perspectives not visible in the original dataset [1]–[8]. However, these approaches often fail to acquire assets beyond those in the dataset, which limits diversity and controllability. Alternatively, some methods integrate with traditional rendering engines [9] and employ mathematical techniques alongside large language models [10] to organize assets. These methods typically require substantial asset collections, resulting in prohibitive costs.

Previous studies [1]–[3], [11]–[14] have predominantly utilized neural radiance fields (NeRF) [15] for 3D shape generation due to their impressive representation capabilities. However, NeRF fully relies on implicit representations, necessitating the use of explicit geometric representations through Marching Cubes algorithms [16]. This additional step not only complicates the editing process but also affects the accuracy and precision of the geometry. Other researchers have explored the use of pre-trained 2D diffusion models [17]–[20] for 3D shape generation, incorporating score distillation sampling (SDS) [17] to extend the knowledge of diffusion models to 3D spaces. While these methods benefit from rich scene knowledge in diffusion models, they are limited by computational inefficiencies arising from the time-consuming nature of diffusion sampling and subsequent reconstruction.

3D Gaussian Splatting [21] (3DGS) employs rasterization, resulting in an explicit geometric representation akin to a point cloud. This allows for direct manipulation using existing tools and offers rendering speeds on par with those of mesh models. The technique effectively bridges the gap between rapid scene construction and robust editing capabilities. Consequently, it is imperative to investigate methodologies for the rapid and high-fidelity generation of 3DGS assets.

Current research predominantly integrates diffusion models with 3DGS to divide the generation task into a two-stage pipeline: generation and reconstruction. Most studies employ 3DGS during the reconstruction phase [22], [23], with minimal emphasis on the intrinsic generation capabilities of 3D Gaussians. Our work focuses on developing high-fidelity methods for generating 3D Gaussians, aiming to create intelligent generation plugins compatible with 3D Gaussian rendering engines. This research endeavor seeks to address the lack of 3D Gaussian asset generation and expedite the development and deployment of intelligent simulation engines utilizing 3DGS.

Generating high-fidelity 3D Gaussians presents several challenges. The inherent design of 3DGS often omits additional networks, relying instead on the direct optimization of specific attributes. This limitation restricts the generation capabilities by preventing the direct addition of noise. To overcome this, we introduce GET3DGS, a novel approach that utilizes points deformation fields for 3D Gaussians. This method incorporates geometry collapse fields and points SH fields, offering a controlled interface for generating physical attributes from both geometric and textural perspectives. It permits indepen-

This work was supported by National Science and Technology Major Project under Grant 2022ZD0116305; by the National Nature Science Foundation of China under Grant 62227801; (Corresponding author: Huimin Ma.)

H. Yu, W. Gong, J. Chen and H. Ma are with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China. (e-mails: haochen.yu@xs.ustb.edu.cn, M202210639@xs.ustb.edu.cn, jschen@ustb.edu.cn, mhmpub@ustb.edu.cn)

dent manipulation of geometry and texture noise. Moreover, to mitigate instability during GAN training, we propose a progressive densification training strategy that integrates the progressive training approach of GANs with the densification of 3D Gaussians.

Our model efficiently learns high-quality three-dimensional geometric representations from two-dimensional images, thereby significantly reducing asset accumulation costs. This approach supports direct data generation that is compatible with three-dimensional Gaussian rendering engines, enhancing data-driven scene synthesis. Our model is particularly adept at disentangling texture and shape representations from two-dimensional images, which allows for disentangled manipulation of attributes. This capability ensures substantial variability and adaptability in the outputs, effectively addressing issues of poor generalization and diversity. Utilizing a GAN architecture, our model achieves rapid sampling speeds, comparable to those reported in GET3D [24], thereby decreasing inference time and enhancing usability for real-world applications. By leveraging three-dimensional Gaussians, our model ensures the production of high-fidelity images and geometries without compromising geometric accuracy, which is critical for applications in three-dimensional graphics, virtual reality, and simulation.

We assessed our model using the ShapeNet [25] and OmniObject3D [26] datasets, achieving superior performance over prior studies in terms of geometric and image synthesis quality. Additionally, comparisons with existing point cloud generation methodologies demonstrated that the geometric quality of the outcomes from our model is on par with those dedicated solely to geometry generation.

Our contributions are summarized as follows:

- We propose a high-fidelity asset generator utilizing 3D Gaussian representations, which allows for controlled generation of shape and texture in 3D Gaussian assets.
- We introduce a template-based point cloud deformation method, named points deformation fields, that maps Gaussian point cloud templates to target dataset distributions.
- We propose a progressive densification training strategy that utilizes multi-resolution progression to enhance the densification of 3D Gaussians, resulting in rapid and stable training.

II. Related Work

3D Gaussian Splatting. The standard depiction of Gaussians as ellipsoids shows that 3D Gaussians and ellipsoids are isomorphic. Eq. (1) illustrates this normalized ellipsoid representation. This isomorphism implies that a sufficiently detailed combination of these models can construct any geometric shape in 3D space. 3D Gaussians possess inherent properties such as spatial coordinate X, anisotropic covariance Σ, opacity ρ, and spherical harmonics coefficients SHs. The anisotropic covariance includes rotation factors Ψ and scaling S, as depicted in Eq. (2), with the final ellipsoidal shape being derived through axis-specific scaling followed by rotation.

$$G(X) = e^{-\frac{1}{2}(X)^T \Sigma^{-1}(X)} \tag{1}$$

$$\Sigma = \Psi S S^T \Psi^T \tag{2}$$

Ellipsoids are projected onto a plane using a rapid rasterization rendering pipeline to generate 2D images. The projection formula, given a transformation matrix W and the Jacobian matrix J of the affine transformation for projective mapping, is described by Eq. (3).

$$\Sigma^{'} = JW\Sigma W^T J^T \tag{3}$$

Eq. (4) presents the rendering method, where c_i represents the color of the i^{th} 3D Gaussians within a pixel, as determined from the corresponding perspective using spherical harmonic coefficients (SHs). N denotes the number of points currently covered by the pixel. These points, organized by depth, are accumulated through a pointwise process.

$$C = \sum_{i \in N} c_i \rho_i \prod_{j=1}^{i-1} (1 - \rho_j), \tag{4}$$

The 3D Gaussian model employs the attributes of its distribution as directly trainable parameters, omitting any neural network layers. It undergoes training using a reconstruction loss function and is densified at a predetermined frequency.

Our study utilizes 3DGS for 3D representation, enhancing this approach with supplementary networks to produce 3DGS assets. By employing differentiable rendering coupled with Gaussian rasterization, we refine the physical attributes of Gaussian point clouds. Additionally, our methodology includes a densification process for 3D Gaussians, which establishes a robust training pipeline.

3D Generation. Classified by the mode of representation, the task of generating 3D models is primarily divided into four categories: mesh generation [24], [27]–[29], 3D point clouds generation [30]–[41], surface generation through SDF [42]–[45], and implicit methods epitomized by NeRF [2], [3], [13], [14], [46]–[48]. Two main approaches facilitate the 3D generation tasks: 3D-aware GANs or the integration of reconstruction and diffusion models. References [22], [23], [49] pioneer the integration of 3DGS with diffusion models to generate 3D models from text prompts. Works utilizing differentiable rendering, categorized as 3D-aware generation [2], [3], [13], [14], [47], aim to learn 3D implicit representations from 2D images [50]. Studies [24], [51], [52] use 2D images to learn explicit mesh structures. Investigations such as [24], [39], [47], [48] have achieved controllable generation by exploring the latent space of models.

Our research aligns with these models and focuses on utilizing a texture-geometry disentangled latent space for controllable and rapid generation of 3DGS assets. This is achieved through the use of generative adversarial networks that are self-supervised via 2D rendering.

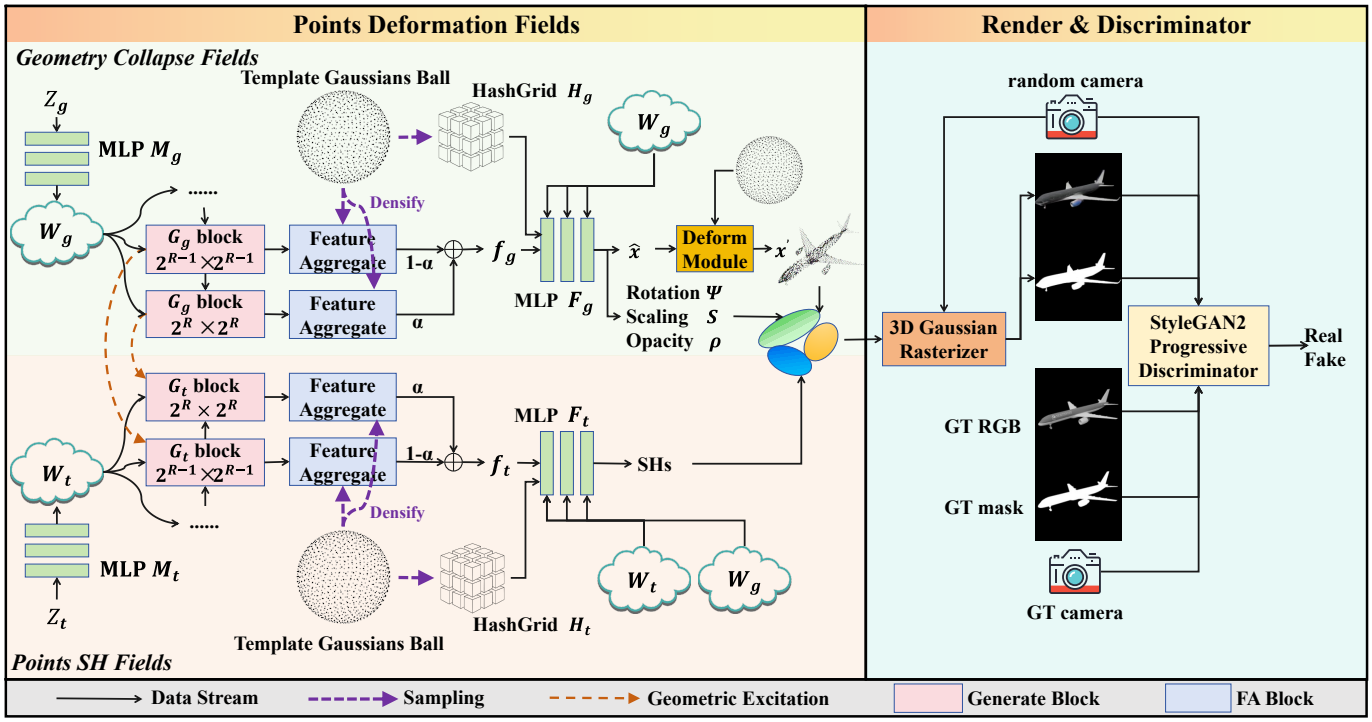

Fig. 1. The main architecture diagram of our model GET3DGS, which is a 3D-aware GAN structure. The network includes geometry collapse fields and points SH fields. These two fields jointly constitute the points deformation fields. Both fields have generator blocks, feature aggregate modules, and an MLP decoder.

III. Method

The primary challenge in generating 3D Gaussians lies in accurately producing their physical properties. For unconditional generation, a pragmatic solution involves enhancing the 3D Gaussian framework with an additional network. This network converts a random noise distribution into a structured distribution that represents physical attributes. This method of network augmentation, previously applied in the field of 4D Gaussians [53], [54], begins with the training of a 3D Gaussian model, followed by a Multilayer Perceptron (MLP) network. The MLP is tasked with learning and adapting to the temporal dynamics of the physical attributes of point clouds, thereby facilitating the generation of 4D scenes.

Nevertheless, this methodology centers on understanding temporal deformations within established 3D Gaussians. In generative tasks, the final distribution of point clouds remains unpredictable. Although pre-trained 3D Gaussian point clouds do not exist, utilizing an initial point cloud template and learning the deformation mapping to the desired target is feasible, inspired by the concept of 4D Gaussians. Introducing noise can facilitate the production of diverse outputs, thereby supporting varied generative processes. Similarly, for point cloud colors, employing a technique akin to shape processing that maps noise to their Spherical Harmonics (SH) distribution can achieve detailed color representations.

Similar to other explicit and implicit 3D representations, the attributes of 3D Gaussians are divided into geometric and texture color attributes. Geometric attributes in our model encompass coordinates, rotation, scaling, and opacity, whereas texture color attributes are defined exclusively by spherical harmonic coefficients. Previous studies, including references [2], [3] and [13], have demonstrated that texture color is influenced by geometric attributes.

Feature fields are essential for linking physical attribute distributions with noise distributions in 3D generation tasks. Consequently, providing detailed 3DGS features on a point-by-point basis is crucial. Furthermore, to comply with parallelization requirements, the number of 3DGS points must be consistent within each batch. To address this, Gaussian template points have been designed to meet these fixed-number requirements.

To address the requirements of point-wise feature assignment and accommodate the geometric and texture attributes of 3D Gaussians, we propose a model that incorporates geometry collapse fields (Section III-A) and points SH fields (Section III-B). The geometry collapse fields reposition the template Gaussian point cloud and derive parameters for scaling, rotation, and opacity adjustments. The points SH fields predict the spherical harmonic coefficients following template deformation. Furthermore, to improve training speed and stability, we introduce a progressive densification training strategy (Section III-C). The overall architecture of our model is illustrated in Fig. 1.

We treat the points deformation fields as the generator. During training, we utilize adversarial loss to optimize the generator and the discriminator in sequence. Specifically, two noises are randomly sampled from a normal distribution and simultaneously input into the points deformation fields, which outputs five attributes of 3DGS. These attributes are then combined to form a 3DGS model, and images are rendered

using the 3DGS differentiable rasterizer that supports batch rendering. The discriminator receives the images rendered by the renderer and the sampled images from the dataset until the generator achieves distribution fitting. During inference, it is possible to simultaneously sample two noise inputs for unconditional 3DGS asset generation, or to fix one of the noises while sampling the other for disentangled control.

A. Geometry Collapse Fields

The geometry collapse fields are designed to map the template point cloud to a specific distribution. The geometry deformation network accepts point coordinate $\vec{x}$ as inputs and outputs the mapped positions $\vec{x}'$, denoted as CF. Consequently, the mapping process is formulated as shown in Eq. (5).

$$\vec{x}' = CF(\vec{x}) \tag{5}$$

However, it is an undeniable fact that training a simple network, such as na MLP, requires mapping a large number of points to their appropriate positions, thereby creating an extremely large solution space. Given the unordered nature of point cloud data, deformations do not adhere to a bijective mapping. A point may be present both at the front and the rear of a vehicle; however, this does not alter the final configuration of the point cloud. Once the network achieves sufficient robustness, it can autonomously compensate for any missing elements. Such a vast solution space negatively impacts network training, potentially resulting in significant failures during the training of GANs.

To address the vast solution spaces, one possible strategy is to restrict morphological changes to the radial direction, requiring points on the template to move only toward the origin. This method reduces the volume of the solution space and enhances the stability of the network. However, empirical evidence from our experiments suggests that this approach does not guarantee stability during the training process and adversely affects the rendering quality. This is primarily due to difficulties in capturing high-frequency details. The distribution of frequencies in an object's texture and volume is highly uneven, and the template lacks the capability to predict which regions need a denser point cloud to maintain texture fidelity. Consequently, areas with intricate details, such as car tires and patterns, often fail to replicate textures accurately due to insufficient point density.

In optimize the volume of the solution space and the density of the point cloud, we introduce a strategy involving density correction followed by radial collapse, as illustrated in Fig. 3. This approach entails computing the radial and rotational angle offsets for a template within spherical coordinates. The deformation function is segmented into radial and rotational components, as delineated in Eq. (6).

$$CF(\vec{x}) = T \circ R(\vec{x}) \tag{6}$$

$$\begin{aligned} CF(r,\theta,\varphi) &= T \circ R(r,\theta,\varphi) = T(r,\theta+\delta\theta,\varphi+\delta\varphi) \\ &= r-\delta r,\theta+\delta\theta,\varphi+\delta\varphi \\ &= r-\hat{r}r,\theta+\hat{\theta}\cdot 2\pi,\varphi+\hat{\varphi}\cdot\pi \end{aligned} \tag{7}$$

We employ spherical coordinate representation to simplify the expression, as indicated in Eq. (7). The rotation function introduces an offset $\delta\varphi$ and $\delta\theta$ to the elevation angle φ and azimuth angle θ, respectively. Simultaneously, the radial function decreases the radius r by a specific offset δr. Where, $\delta\varphi \in [0,\pi]$, $\delta\theta \in [0,2\pi]$, $\delta r \in [0,r]$. To align the network outputs, we use normalized values of three offsets, where $\hat{\theta} \in [0,1]$, $\hat{\varphi} \in [0,1]$, $\hat{r} \in [0,1]$. Normalized outputs enhance network stability and can be readily constrained by activation functions.

To generate diverse outputs, we add noise as input to the geometry collapse fields, as Eq. (8) shows, where $\hat{r}_{\mathbf{z}_g}, \hat{\theta}_{\mathbf{z}_g}, \hat{\varphi}_{\mathbf{z}_g}$ are part of the geometry collapse network outputs.

$$\begin{aligned} &CF([r,\theta,\varphi],\mathbf{z}_g) \\ &\quad= (r-\hat{r}_{\mathbf{z}_g}r,\theta+\hat{\theta}_{\mathbf{z}_g}\cdot 2\pi,\varphi+\hat{\varphi}_{\mathbf{z}_g}\cdot\pi) \\ &\quad= (r,\theta,\varphi)+(-r,2\pi,\pi)\cdot diag(\hat{r}_{\mathbf{z}_g},\hat{\theta}_{\mathbf{z}_g},\hat{\varphi}_{\mathbf{z}_g}) \end{aligned} \tag{8}$$

Utilizing traditional models such as dense MLPs as geometry collapse networks is typically inadvisable due to their insufficient three-dimensional perceptual capabilities. The triplane approach proposed by EG3D [14] may lead to data redundancy through unsampled regions on the planes. To address this, we introduce the feature aggregate module as depicted in Fig. 2. We have developed a bi-plane structure tailored for spherical point cloud templates. This structure involves positioning plane features around the spherical surface, thereby wrapping the planes around it. Subsequent sampling of template point clouds on the spherical surface ensures efficient utilization of plane features. Features derived from sampling across two spherical surfaces are concatenated and processed through a shallow MLP network to decode the deformation variables. Feature sampling involves interpolation operations, which enhance feature smoothness. To compensate for potential losses, we employ position encoding integrated with plane sampling features.

Specifically, the geometric noise $\mathbf{z}_g$ is encoded as a styled vector $\mathbf{w}_g$ of the geometric in W-space using an MLP M_g, and the generator block of StyleGAN2 [55] $G_g(\cdot)$ is used to synthesize two planes covered with features. Then the coordinate position of the template $\vec{x} = (r,\theta,\varphi)$ is used for grid sampling and feature aggregating $FA(\cdot,\cdot,\cdot)$, outputs the geometric perception features $\mathbf{f}_g$. The MLP network $F(\cdot,\cdot,\cdot)$ accepts style vectors $\mathbf{w}_g$, geometric perception features $\mathbf{f}_g$ and hash position encode $H_g(\vec{x})$ as inputs, decodes them into specific offsets $\hat{r}_{\mathbf{z}_g}, \hat{\theta}_{\mathbf{z}_g}, \hat{\varphi}_{\mathbf{z}_g}$ and calculates the specific position of the deformed point cloud using deformation module DM ($x' = DM(\hat{x})$). By integrating the remaining attributes of 3D Gaussians into the equation, the final mathematical representation of geometric collapse fields is presented in Eq. (9). Where $[\cdot,\cdot,\cdots,\cdot]$ denotes the operation of constructing a horizontal

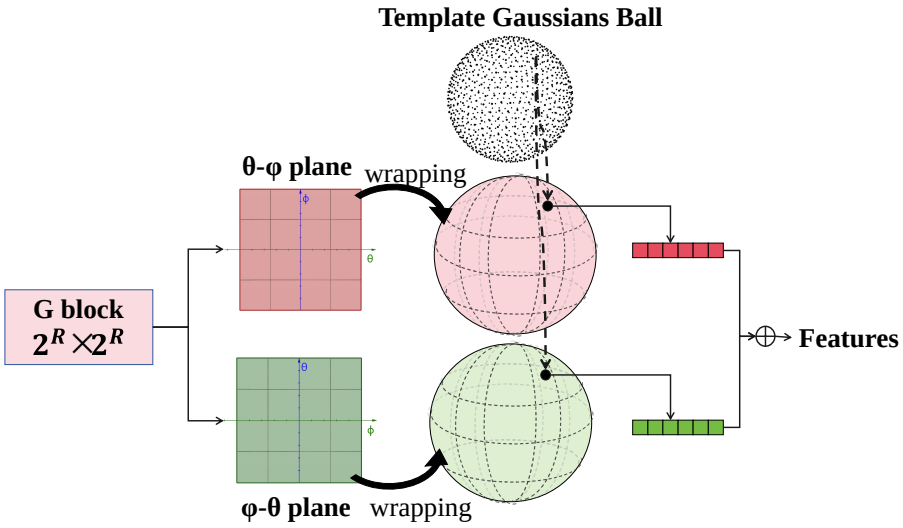

Fig. 2. The feature aggregate module (FA). Using templates to sample and aggregate the wrapping features.

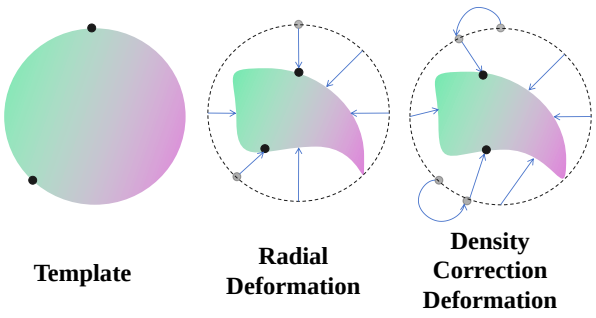

Fig. 3. (Left) The direct rendering result of the Gaussian template is a colored sphere. (Middle) Morphological changes are restricted to the radial direction. (Right) The strategy involves density correction followed by radial collapse.

vector by element flattening, Ψ,S,ρ denote the rotation, scaling and opacity of 3D Gaussians.

$$\begin{aligned}&[\vec{x}'_{\mathbf{z}_g}, \Psi_{\mathbf{z}_g}, S_{\mathbf{z}_g}, \rho_{\mathbf{z}_g}] = CF(\vec{x}, \mathbf{z}_g) = CF([r, \theta, \varphi], \mathbf{z}_g) \\ &= [DM([\hat{r}_{\mathbf{z}_g}, \hat{\theta}_{\mathbf{z}_g}, \hat{\varphi}_{\mathbf{z}_g}]), \Psi_{\mathbf{z}_g}, S_{\mathbf{z}_g}, \rho_{\mathbf{z}_g}] \\ &= [r, \theta, \varphi, \vec{0}] + [-r, 2\pi, \pi, \vec{1}] \cdot J \\ &where \quad J = diag([\hat{r}_{\mathbf{z}_g}, \hat{\theta}_{\mathbf{z}_g}, \hat{\varphi}_{\mathbf{z}_g}, \Psi_{\mathbf{z}_g}, S_{\mathbf{z}_g}, \rho_{\mathbf{z}_g}]) \\ &= diag(F_g(FA(G_g(M_g(\mathbf{z}_g)), \theta, \varphi), M_g(\mathbf{z}_g), H_g(\vec{x})))\end{aligned} \tag{9}$$

We employ various activation functions to constrain the physical attributes output by the geometry collapse fields. Specifically, for opacity and radius offset, we use the sigmoid function to constrain them within the range [0, 1]. For rotation quaternions, we utilize the tanh activation function to confine them within the range [-1, 1]. Regarding scaling, we set a scaling threshold multiplied by the sigmoid function to prevent large-scale ellipsoidal obstructions of the view during network initialization. This is to avoid gradient instability that could result from excessively high scaling factors.

B. Points SH Fields

To generate the texture of 3D Gaussians, we use the same structure as the geometry collapse fields. The texture noise $\mathbf{z}_t$ is encoded as a styled vector $\mathbf{w}_t$ using an MLP M_t, and feed to the textural generator block $G_t(\cdot)$. We also utilize two planes to generate SH features and subsequently sample these features to facilitate shape coloring. Given that texture is influenced by shape, we employ geometric excitation for the points SH fields at each point. Specifically, the features generated by the module at each geometric resolution are input into the points SH fields generation module at the corresponding resolution. These are then combined with the color features, as illustrated in Fig. 4.

Geometric excitation involves injecting the intermediate features from the convolution of the geometry collapse fields into the points SH fields and performing element-wise multiplication. This approach is akin to the style modulation used in StyleGAN. We believe that this style modulation will ensure that the feature content of the texture branch is not excessively influenced by the excitation, but is instead modulated to enable the texture to adapt to the geometric distribution.

The geometry and texture stylized vectors are concatenated to guide the MLP decoder to output texture attributes of 3D Gaussians, where $\oplus$ denotes the concatenate operator. As shown in Eq. (10), we adopt the same sampling and aggregation methods as used for geometry collapse fields. These are then input into a lightweight MLP F_t along with shape features, to facilitate the learning of SH coefficients, denoted as SHs.

$$\begin{aligned}SHs &= SF(\vec{x}, \mathbf{z}_t, \mathbf{z}_g) \\ &= F_t(FA(G_t(M_t(\mathbf{z}_t), G_g(M_g(\mathbf{z}_g))), \theta, \varphi), \\ &\quad M_t(\mathbf{z}_t) \oplus M_g(\mathbf{z}_g), H_t(\vec{x}))\end{aligned} \tag{10}$$

C. Progressive Densification Training

Directly learning the geometry and color mapping of a large number of point clouds is inherently complex. When

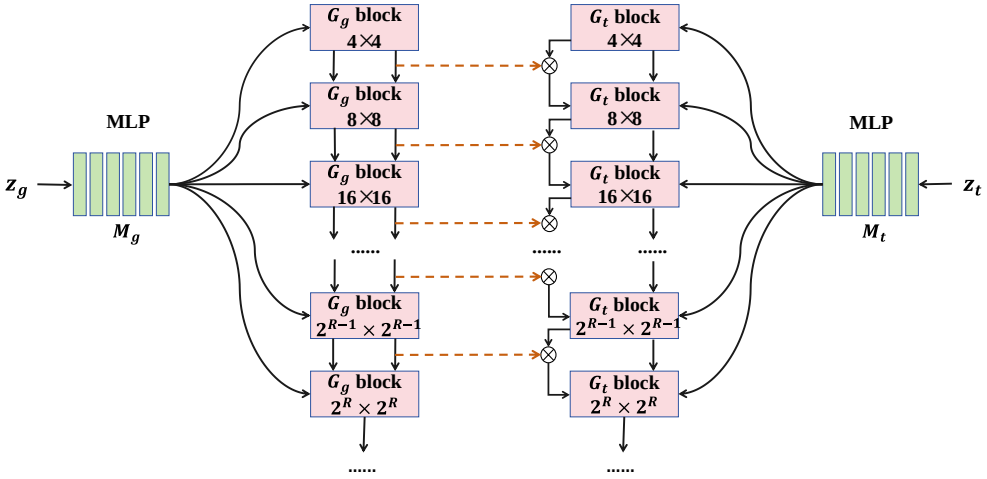

Fig. 4. A schematic diagram of geometric exception, where geometric features are injected into texture features and fully fused for convolution.

employing GANs, learning the distributions of geometry and texture can be slow and unstable. During the training of 3D Gaussian models, the point count increases in each iteration through a process known as splitting. To tackle this issue, we propose a progressive densification training strategy for point clouds that merges the concepts of progressive generator resolution with the splitting of 3D Gaussians. Initially, a resolution R_{init} is selected, which sets the dimensions for geometry collapse fields, points SH fields, and the resolution for 3D Gaussian Splatting rendering.

As the number of iterations increases, the resolution is doubled in the splitting process of 3D Gaussians. The inherent capabilities of Gaussian point clouds for learnable rotation and scaling obviate the need for reinitialization based on point cloud distances during splitting. This prevents network fluctuations and ensures training stability. Consequently, our model progressively densifies by splitting coordinate positions, thereby aligning point clouds with pixel resolution. At lower resolutions, fewer 3D Gaussians are needed. In our self-supervision approach, downsampled authentic images that lack excessive details enable 3D Gaussians to approximate the fundamental structure and color scheme of the object. High-resolution training further refines these details.

The progressive densification progress is described by Eq. (11). Where R represents the base 2 logarithm of the current training resolution, $\vec{x}$ represents the coordinates of the template point clouds, and α represents the proportion of training completed progressively, which uniformly increases from 0 to 1 throughout the training at the current resolution.

$$\begin{aligned}
\mathbf{f}_g^R(\vec{x}) &= FA(G_g^R(M_g(\mathbf{z}_g), \theta, \varphi) \\
\mathbf{f}_t^R(\vec{x}) &= FA(G_t^R(M_t(\mathbf{z}_t), G_g^R(M_g(\mathbf{z}_g))), \theta, \varphi) \\
\mathbf{f}_g^R(\vec{x}) &= \mathbf{f}_g^R(\vec{x}) \cdot \alpha + \mathbf{f}_g^{R-1}(\vec{x}) \cdot (1-\alpha) \\
\mathbf{f}_t^R(\vec{x}) &= \mathbf{f}_t^R(\vec{x}) \cdot \alpha + \mathbf{f}_t^{R-1}(\vec{x}) \cdot (1-\alpha)
\end{aligned} \tag{11}$$

D. Gaussian Template

In our methodology, we utilize a mathematical model to achieve uniform point distribution across the surface of a sphere. This model calculates coordinates to ensure even spacing throughout the sphere.

The process is initiates by calculating the total number of points based on the resolution of the 3D Gaussian template. For each point, we compute two essential spherical coordinates: the polar angle ϕ and the azimuthal angle θ. These calculations are as shown in Eq. (12).

$$\begin{aligned}
\phi_i &= \arccos\left(-1 + \frac{2i+1}{N_{temp}}\right) \\
\theta_i &= \sqrt{N_{temp}\pi} \cdot \phi_i
\end{aligned} \tag{12}$$

where i is the index of each point, and N_{temp} is the total number of points. To ensure the points wrap correctly around the sphere, we apply modulo operations on the angles, with θ adjusted by 2π and ϕ by π. This mathematical model enables the uniform distribution of points across the surface of a sphere.

TABLE I
DIFFERENT TEMPLATE DENSITIES AT VARIOUS RESOLUTIONS.

resolution	N_{temp}
128	(128 × 128) 16384
256	(160 × 160) 25600
512	(200 × 200) 40000

Within our experimental framework, point cloud templates with varying densities are generated at different resolutions by progressively increasing the number of points as the resolution rises. The detailed experimental configurations are presented in Table I.

E. Training Loss

In this study, we adopted an adversarial loss function consistent with GET3D. Moreover, no additional regularization was necessary. The equation is presented as Eq. (13).

$$L(D_x, G) = \mathbb{E}_{\mathbf{z}\in\mathcal{N}, c\in\mathcal{C}}[g(D_x(R_{gau}(Ge(\mathbf{z}), c)))] + \mathbb{E}_{I_x\in p_x, c\in p_c}[g(-D_x(I_x, c)) + \lambda||\nabla D_x(I_x, c)||_2^2] \quad (13)$$

Where $\mathcal{N}$ represents the normal distribution, $\mathcal{C}$ signifies the uniform distribution over a specified interval, Ge denotes the entire generator, and R_{gau} represents the rasterization rendering of 3DGS. The activation function is represented by $g(u) = -log(1 + e^{-u})$ while p_x and p_c correspond to the data distributions of real images and real camera labels, respectively. The coefficient λ is the penalty term for the discriminator. The overall loss function is shown in Eq. (14).

$$L = L(D_x, Ge) + L(D_{mask}, Ge) \quad (14)$$

IV. Experiments

A. Basic Settings

Baselines. We select 2D-to-3D methods, including GRAF [1], Pi-GAN [11], PartNeRF [12], DiffTF [46], GET3D [24] and GaussianCube [49] as the baselines for both image and 3D shape generation. Additionally, we choose 3D methods such as PointFlow [33], DiT-3D [36], DPM [56], PSF [57] and MeshDiffusion [58] for comparing point cloud generation quality.

Datasets. ShapeNet [25] is a widely recognized dataset used for the study and analysis of 3D shapes. This comprehensive database contains a wide range of 3D models across various categories. In our study, we adopted similar preprocessing, rendering techniques, data splitting methods, and camera viewpoints as those used in GET3D to create a 2D image dataset focused on four categories: cars, chairs, motorbikes, and airplanes.

The OmniObject3D [26] dataset is a large-scale collection of real-world 3D models, comprising approximately 6000 models across 216 distinct categories. These models have undergone precise scanning and detailed annotation. Compared to ShapeNet, OmniObject3D includes fewer objects per category.

Metrics. We followed the experimental setup of GET3D and calculated FID and KID for 2D image quality, along with the geometric metrics CD-COV and CD-MMD for point clouds. Using ShapeNet, we generated 50K images for FID and KID evaluation, and synthesized point clouds at five times the test set size for geometric metrics. For OmniObject3D, we also generated 50K images for FID and KID, and synthesized 5000 point clouds for comparison with 1000 real point clouds in metric evaluation.

It is important to note that our model generates 3D Gaussians instead of meshes, which contrasts with the traditional approach of uniformly sampling point clouds across mesh surfaces, as employed by GET3D. To ensure fair experimental comparisons, we applied the Farthest Point Sampling (FPS) technique to uniformly extract points from the 3D Gaussians. We adhered to the standard of sampling 2048 points for metric computation, consistent with GET3D and other benchmark methods. These points were evaluated using the Coverage Score (COV) and Minimum Matching Distance (MMD) metrics, with Chamfer Distance (CD) serving as the comparison metric. The calculation methods for CD, COV, and MMD are shown in Eq. (15).

TABLE II
Values of Gamma Across Different Datasets and Resolutions.

Datasets	resolution	λ	Datasets	resolution	λ
Car	128	3200	Chair	128	3200
	256	3200		256	3200
	512	3200		512	3200
Motorbike	128	3200	Plane	128	10
	256	3200		256	3200
	512	3200		512	-
OmniObject3D	128	10			
	256	10			
	512	40			

$$\begin{aligned} CD(X,Y) &= \sum_{x\in X}\min_{y\in Y}||x-y||_2^2 + \sum_{y\in Y}\min_{x\in X}||x-y||_2^2, \\ COV\left(S_g S_r\right) &= \frac{|\{\arg\min_{Y\in S_r} CD(X,Y) \mid X\in S_g\}|}{|S_r|}, \\ MMD\left(S_g, S_r\right) &= \frac{1}{|S_r|}\sum_{Y\in S_r}\min_{X\in S_g} CD(X,Y) \end{aligned} \quad (15)$$

Hyper Parameters. The regularization parameter, λ, plays a crucial role in tuning the generalization ability of the model by penalizing complexity. For most datasets, λ is consistently set to 3200 across all training resolutions to mitigate overfitting. However, specific adjustments were made to accommodate the unique characteristics of certain datasets. As Table II shows, in the OmniObject3D dataset, λ was adjusted to 10 for resolutions 128×128 and 256×256, and 40 for the 512×512 resolution. In our experiments , we set the dimensions of $\mathbf{z}_g, \mathbf{z}_t, \mathbf{w}_g, \mathbf{w}_t$ to 512. We use a mapping network with 8 layers to encode the noise $\mathbf{z}_g, \mathbf{z}_t$ into $\mathbf{w}_g, \mathbf{w}_t$ space. For the hyperparameter of rendering, we adopt the experimental settings of GET3D. We train our model with $batchsize = 64$. In our experimental setup, cameras were positioned on the surface of a sphere with a radius of 1.2. For the ShapeNet and GAME datasets, the azimuth angle ranged from 0 to 2π, and the elevation angle was set from 0 to $\pi/6$. For the OmniObject3D dataset, the elevation angle was adjusted to range from 0 to $\pi/2$ radians. The FOV of the camera was set to 50 degrees. In all experiments, the learning rate was set to 0.002.

B. Comparison with Baselines

The comparison experiments are categorized based on the resolution of the datasets used for the baselines.

In Table III, a quantitative comparison between our model and other baselines at a resolution of 128 × 128 is presented. Our model outperforms others across four metrics: FID, KID, COV, and MMD. It demonstrates clear advantages in both low-resolution 2D image generation and 3D shape generation. GRAF, PiGAN and PartNeRF rely on NeRF's implicit rendering capabilities, employing the marching cubes algorithm to extract explicit geometric structures. GET3D shows suboptimal performance at lower resolutions, suggesting that our method can produce higher-quality geometric structures even at low resolutions.

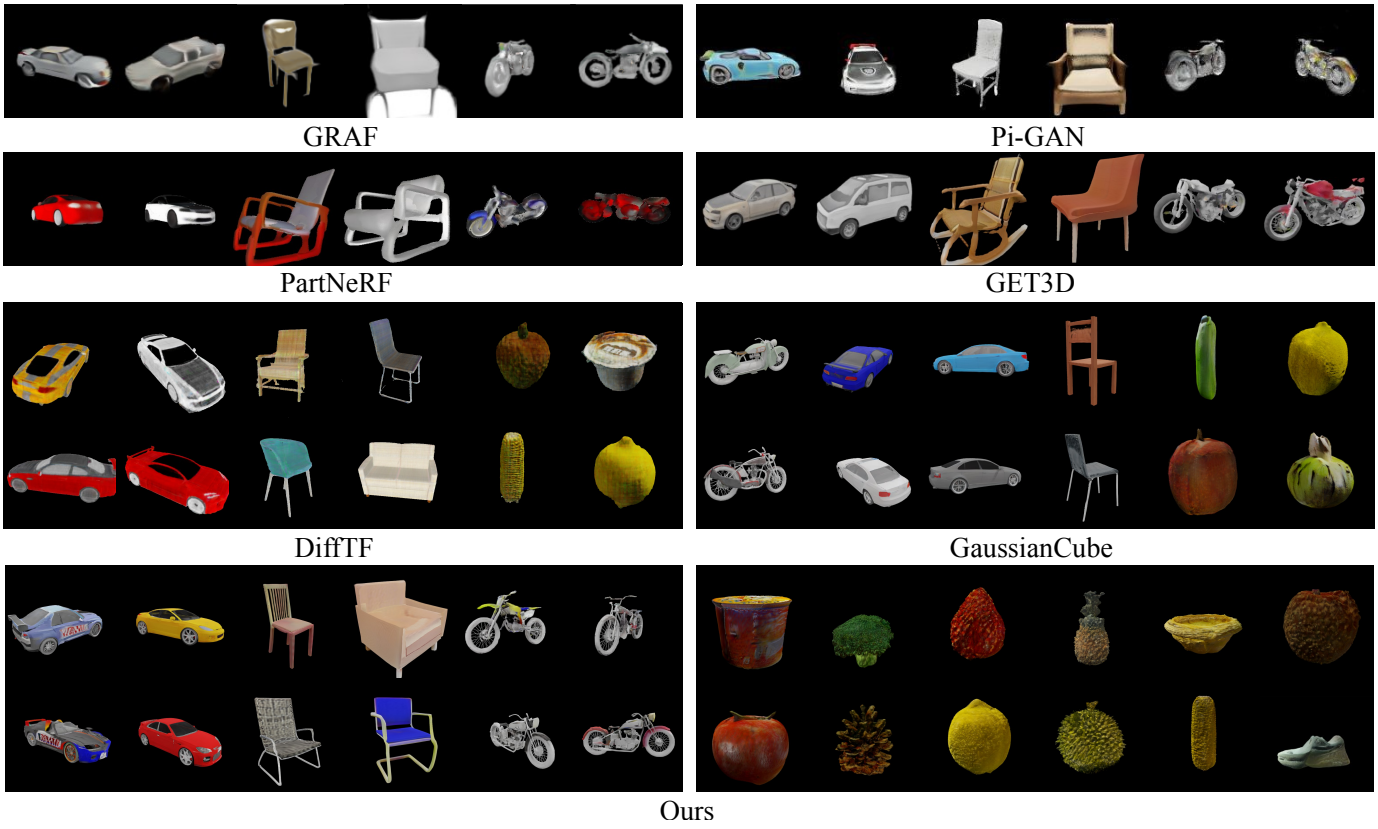

Fig. 5. Qualitative comparison of rendering effects between our model and related works. Our model produces high-quality and more natural images. DiffTF shows significant texture stripes, while other baselines fail to capture fine details, like car tires.

TABLE III
THE RESULTS OF THE EVALUATION OF 2D IMAGE QUALITY AND 3D POINT CLOUD GEOMETRY QUALITY UNDER 128 × 128 RESOLUTION.

Dataset	model	FID↓	KID↓ (‰)	COV↑(%)	MMD↓
Car	GRAF	100.08	90.28	36.01	1.23
	Pi-GAN	73.67	56.17	31.39	1.11
	GET3D	36.21	22.50	8.25	1.30
	PartNeRF	98.71	85.00	42.70	1.33
	GET3DGS(ours)	**18.31**	**8.38**	**53.21**	**1.09**
Chair	GRAF	54.56	22.19	30.41	9.73
	Pi-GAN	107.29	82.27	42.16	5.31
	GET3D	68.17	41.22	33.98	5.90
	PartNeRF	123.76	110.23	51.26	5.87
	GET3DGS(ours)	**23.79**	**14.39**	**72.86**	**3.07**
Plane	GRAF	43.90	26.55	38.40	6.89
	Pi-GAN	78.68	60.21	34.02	5.98
	GET3D	43.29	29.20	36.54	4.06
	PartNeRF	100.21	81.27	43.24	3.21
	GET3DGS(ours)	**13.58**	**6.54**	**62.00**	**1.58**
MotorBike	GRAF	75.80	49.10	67.86	1.26
	Pi-GAN	148.77	131.68	63.01	1.56
	GET3D	81.25	46.74	56.72	2.11
	PartNeRF	141.68	132.27	56.16	1.73
	GET3DGS(ours)	**66.69**	**39.33**	**73.75**	**1.03**

TABLE IV
THE RESULTS OF THE EVALUATION OF 2D IMAGE QUALITY AND 3D POINT CLOUD GEOMETRY QUALITY UNDER 256 × 256 RESOLUTION.

Dataset	model	FID↓	KID↓ (‰)	COV↑(%)	MMD↓
Car	GRAF	194.70	167.00	28.91	1.34
	Pi-GAN	62.60	41.12	32.34	1.31
	GET3D	24.03	13.23	21.45	1.12
	PartNeRF	145.41	117.49	49.53	1.52
	GET3DGS(ours)	**14.54**	**6.15**	**62.36**	**1.07**
Chair	GRAF	100.20	74.71	26.62	10.98
	Pi-GAN	90.28	70.02	41.12	6.01
	GET3D	37.12	22.18	51.27	4.20
	PartNeRF	129.09	108.67	70.20	4.22
	GET3DGS(ours)	**24.40**	**14.33**	**78.17**	**3.69**
MotorBike	GRAF	169.80	140.60	49.82	2.01
	Pi-GAN	89.87	69.82	63.27	1.92
	GET3D	76.21	48.76	63.72	1.80
	PartNeRF	145.43	135.06	64.38	1.45
	GET3DGS(ours)	**75.11**	**45.28**	**76.25**	**0.93**
OmniObject3D	GRAF	111.75	93.12	13.43	9.01
	Pi-GAN	101.54	87.11	23.17	7.65
	GET3D	43.32	21.77	31.23	5.28
	GET3DGS(ours)	**27.40**	**10.27**	**34.27**	**4.46**

Table IV presents a quantitative comparison of the metrics of our model at a 256 × 256 resolution against other baselines. Our model outperforms all baselines across all four metrics.

Table V presents the results obtained at a resolution of 512 × 512. In this study, we introduce DiffTF and GaussianCube, and retrain them on our experiment setting. The table also includes a comparison with selected works on 3D unconditional point cloud generation to evaluate the quality differences between our model and purely geometric generation models. The

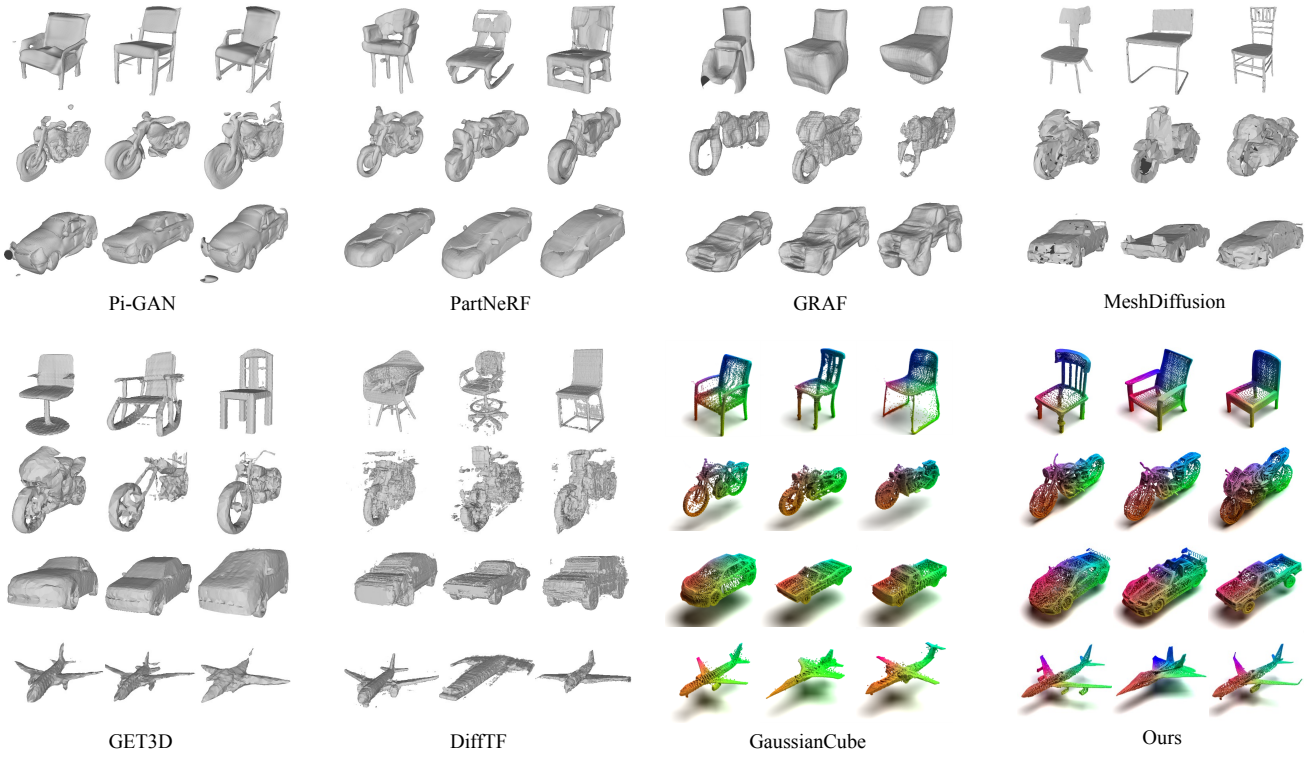

Fig. 6. Qualitative comparison of 3D shape generation between our model and baselines. The baselines either lack geometric detail or exhibit floating artifacts. In view of the representation form of 3DGS, we demonstrate the point cloud geometry of our model, and the point cloud rendering method remains consistent with PointFlow [33].

experimental results demonstrate that our model has better FID and KID metrics on car, chair and OmniObject3D datasets. Regarding geometric quality, our model surpasses DiffTF on the MotorBike and Car datasets. Although the geometric quality of our model is slightly lower than DiffTF in conditional generation tasks, it exhibits superior image rendering quality. In comparison to GaussianCube, our approach demonstrates superior performance in terms of image quality on the car, chair and omniobject3d datasets. Although the image quality on the motor datasets is inferior to GaussianCube, the geometric quality is superior. GaussianCubes generally have better geometric quality, but have poorer geometric diversity on the Shapenet dataset. Comparisons with 3D baselines indicate that the geometric quality of our model closely aligns with their performance under 3D data supervision, and in some cases even exceeds it.

The qualitative comparisons between our model and the baselines across different datasets demonstrate that our model excels in generating clear and detailed synthesized images for 2D image synthesis tasks, as shown in Fig. 5. Furthermore, our model exhibits a strong ability to generate complex point cloud structures for 3D geometry generation tasks, capturing finer details compared to the mesh surfaces produced by the baselines, as illustrated in Fig. 6. Additionally, our model effectively learns geometric information within the interiors of objects.

Fig. 8 and Fig. 9 qualitatively demonstrate the impact of varying texture noise and shape noise on image synthesis. Our model effectively decouples shape and texture, enabling controllable generation of 3D Gaussians.

The inference speed of our model and the baselines was evaluated on a single RTX 4090. The results presented in Fig. 7 show that our model requires only 28ms on average to generate an object, which is significantly faster than the diffusion-based generation methods. With a comparable inference speed to GET3D, our model is characterized as a rapid 3DGS assets generator.

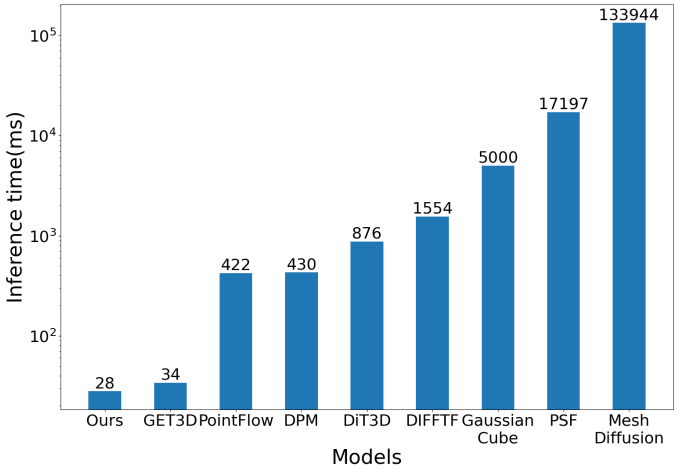

Fig. 7. Comparison of inference speed between our model and each baseline model on a single RTX 4090.

Fig. 8. The result obtained by changing texture noise $\mathbf{z}_t$. The texture of the object changes as the texture noise varies, while the shape remains constant.

Fig. 9. The result obtained by changing geometry noise $\mathbf{z}_g$. The shape of the object changes as the geometry noise varies, while the color remains constant.

TABLE V
THE EVALUATION OF 2D IMAGE QUALITY AND 3D POINT CLOUD GEOMETRY QUALITY UNDER 512 × 512 RESOLUTION. THE UNDERLINES REPRESENTS BETTER GEOMETRIC PERFORMANCE OF THE 3D BASELINE.

Dataset	Type	Model	FID↓	KID↓ (‰)	COV↑(%)	MMD↓
Car	3D	PointFlow	-	-	60.57	0.85
	3D	DiT-3D	-	-	31.70	2.98
	3D	DPM	-	-	60.24	0.95
	3D	PSF	-	-	70.73	1.03
	3D	MeshDiffusion	-	-	66.55	1.12
	2D→3D	GET3D	13.19	5.66	44.13	**0.80**
	2D→3D	DiffTF	73.21	58.72	49.53	1.36
	2D→3D	GaussianCube	16.77	5.22	61.48	0.90
	2D→3D	GET3DGS(ours)	**11.03**	**4.17**	**66.97**	1.04
Chair	3D	PointFlow	-	-	72.94	2.67
	3D	DiT-3D	-	-	76.83	2.82
	3D	DPM	-	-	81.73	2.87
	3D	PSF	-	-	81.92	2.65
	3D	MeshDiffusion	-	-	82.26	2.65
	2D→3D	GET3D	30.16	18.32	69.92	3.90
	2D→3D	DiffTF	80.56	65.34	**76.22**	3.64
	2D→3D	GaussianCube	32.17	15.06	63.39	**3.60**
	2D→3D	GET3DGS(ours)	**24.41**	**13.68**	75.44	3.90
MotorBike	3D	PointFlow	-	-	64.57	0.88
	3D	DiT-3D	-	-	75.34	0.81
	3D	DPM	-	-	80.67	0.99
	3D	PSF	-	-	74.46	1.25
	3D	MeshDiffusion	-	-	89.04	1.08
	2D→3D	GET3D	74.04	45.66	65.75	1.74
	2D→3D	DiffTF	102.46	88.95	41.02	6.05
	2D→3D	GaussianCube	**58.95**	**30.03**	68.60	0.95
	2D→3D	GET3DGS(ours)	61.15	33.98	**81.25**	**0.92**
OmniObject3D	2D→3D	GET3D	28.92	12.01	31.21	6.34
	2D→3D	DiffTF	93.85	51.88	34.59	**6.05**
	2D→3D	GaussianCube	26.45	11.15	**35.10**	6.26
	2D→3D	GET3DGS(ours)	**26.26**	**11.13**	32.59	6.46

C. Ablations

Ablation of Triplane and Bi-plane. Features are sampled on three planes (Triplane) using Gaussian templates, as well as on two planes with our proposed Bi-plane method. A comparison of generation results on the Car and Chair datasets at a resolution of 128 × 128 provides valuable insights. Notably, our Bi-plane method has a similar number of parameters to the Triplane method but is slightly more compact. As shown in Table VI, the Bi-plane method achieves significant improvements in generation performance, potentially due to higher feature utilization. The number of interpolation points on the plane grid used by the sampling points was calculated, along with the feature utilization. The results indicate that our proposed method achieves a feature utilization rate of 93.9%, which is 13% higher than that of the Triplane method.

Ablation of Deformation Method. A comparison was conducted to assess the impact of different deformation methods on the generation effect. Specifically, the direct radial deformation method and the radial plus rotation deformation method were compared in this experiment. The experimental results presented in Table VII suggest that our deformation method results in better generation performance. This superiority can be attributed to the direct radial deformation method's limitation in handling the uneven distribution of frequency details within the object.

Ablation of Single and Double Fields. A comparative analysis was conducted to evaluate the effects of employing a geometric texture dual branch versus a separate coupling branch on the generated results. Specifically, one noise was used to jointly control object properties, while two noises were used independently to control geometric and textural properties. As shown in Table VIII, the implementation of a dual-branch structure may cause a slight decrease in image synthesis performance for the car dataset but enhances the geometric quality and diversity of the point cloud. For the chair dataset, no significant impact on image quality was observed, but there was an improvement in geometric quality. In conclusion, while the dual-branch structure may not significantly affect overall performance, it greatly enhance user controllability.

Ablation of Progressive Densification Training. This experiment examines the impact of applying a progressive densification training approach on generation quality. The experiment evaluated the time required to train the model

TABLE VI
THE EFFECT OF TRIPLANE AND BI-PLANE ON IMAGE AND POINT CLOUD QUALITY. (LR=0.002, BS=96, ITER=4000K)

	Params	Rate	Car 128^2 FID↓	KID↓	COV↑	MMD↓	Chair 128^2 FID↓	KID↓	COV↑	MMD↓
Triplane	84.38M	80.99%	18.16	8.89	53.96	1.09	24.15	14.85	71.19	3.86
Bi-Plane	84.14M	**93.9%**	**17.34**	**8.01**	**54.20**	**1.08**	**23.79**	**14.39**	**72.86**	**3.07**

TABLE VII
EFFECTS OF DIFFERENT DEFORMATION METHODS ON IMAGE AND POINT CLOUD QUALITY. (LR=0.002, BS=96, ITER=4000K)

	Car 128^2 FID↓	KID↓	COV↑	MMD↓	Chair 128^2 FID↓	KID↓	COV↑	MMD↓
radius	18.71	8.75	55.60	1.30	24.34	14.86	71.33	3.95
radius+rot	**17.34**	**8.01**	**57.20**	**1.08**	**23.79**	**14.39**	**72.86**	**3.07**

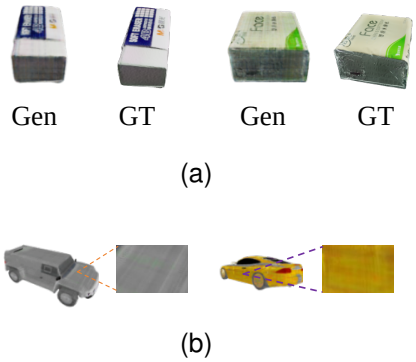

Fig. 10. Two existing problems with DiffTF. (a) DiffTF produces a serious phenomenon of copying data sets. It is easy to find the same objects in the dataset as the generated samples. (b) The object texture generated by DIFFTF has layered stripes.

until optimal performance was achieved. Experimental results presented in Table IX demonstrate that this approach can reduce the training time of the model without compromising its performance. Additionally, adopting this strategy helps reduce training failures.

V. DISCUSSION

As shown in Fig. 10b, the output samples generated by DiffTF exhibit noticeable striations, which complicate the accurate computation of FID and KID, resulting in suboptimal scores. Based on the geometric properties of DiffTF, the generated mesh structure also displays striation-like geometric patterns, leading to anomalies in the corresponding textures. Additionally, the mesh shows prominent floating artifacts.

Analysis of the synthesis images of DiffTF indicates a significant presence of textual elements and icons incorporated as textures into the model. As shown in Fig. 10a, this finding suggests that DiffTF leans more towards reconstruction rather than generation, as it nearly replicates the text and patterns from the dataset. This inclination may be attributed to the design of DiffTF, which includes a reconstruction process followed by the training of diffusion models.

GaussianCube is similar to DiffTF, as both adopt the design principle of reconstruction followed by diffusion. After

TABLE VIII
EFFECT OF SINGLE BRANCH AND DUAL-BRANCH ON THE QUALITY OF IMAGE AND POINT CLOUD. (LR=0.002, BS=96, ITER=4000K)

	Disentangle shape	tex	Car 128^2 FID↓	KID↓	COV↑	MMD↓	Chair 128^2 FID↓	KID↓	COV↑	MMD↓
union	×	×	**15.68**	**6.92**	53.21	1.11	23.80	14.91	**73.56**	3.58
split	✓	✓	17.34	8.01	**54.20**	**1.08**	**23.79**	**14.39**	72.86	**3.07**

TABLE IX
EFFECT OF PROGRESSIVE DENSIFICATION TRAINING ON THE QUALITY OF IMAGE AND POINT CLOUD. (LR=0.002, BS=96, ITER=4000K)

	Training Time	Car 128^2 FID↓	KID↓	COV↑	MMD↓	Chair 128^2 FID↓	KID↓	COV↑	MMD↓
w/o. progressive	2D	17.68	**8.00**	**54.18**	1.08	24.22	15.08	70.24	3.31
progressive	1D 10H	**17.34**	8.01	54.20	**1.08**	**23.79**	**14.39**	**72.86**	**3.07**

filtering out points with transparency values less than one thousandth, we still observe a greater number of floating objects in the point cloud. This phenomenon may be attributed to the unevenness of the 24 randomly captured perspective images. Since our experimental settings align with those of GET3D, we captured 24 images of each object from random angles within a specified range when constructing the 2D dataset. This approach may influence the reconstruction stages of both GaussianCube and DiffTF, as numerous studies have shown that sparse viewing angles can lead to significant floating artifacts in 3D graphics assets.

In contrast, the proposed model does not exhibit the aforementioned issues. It generates smoother and more natural samples compared to GET3D. The incorporation of 3D Gaussians in this approach facilitates better fitting of smooth surface contours, resulting in enhanced FID and KID scores. Conversely, the transparency attribute of 3D Gaussians enables the capture of intricate geometric structures within objects, outperforming GET3D, which focuses solely on surface learning.

The model also exhibits limitations. Variations in scaling thresholds may impact the final geometric quality, necessitating adjustments of thresholds tailored to specific datasets. Furthermore, datasets containing multiple classes with long-tail distributions, such as OmniObject3D, may present challenges, including class confusion and reduced generation quality.

The contribution of this model is the efficient and precise generation of 3DGS assets, complemented by an interface for separating texture and geometry. Although controllable generation based on text and other inputs has not yet been explored, this research provides a method for separating 3DGS attributes and highlights the potential for integration with various conditions. Future studies will focus on the alignment and control of multimodal conditions in latent spaces.

VI. CONCLUSION

This paper presents GET3DGS, a novel, fast, and high-fidelity 3D Gaussian assets generator designed to meet the increasing demand for rapid material generation in the evolving field of 3D Gaussians, thereby accelerating the development of related intelligent simulation engines. By constructing points deformation fields utilizing the proposed geometry collapse

fields and points SH fields, it is possible to synthesize shapes and textures that align with the dataset distribution while achieving disentangled control between geometry and texture. The model also demonstrated stable training through the proposed progressive dense training strategy. Experiments indicate that the model achieves high image and geometric quality, surpassing previous works across most metrics.

REFERENCES

[1] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, "Graf: Generative radiance fields for 3d-aware image synthesis," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 154–20 166, 2020.

[2] M. Niemeyer and A. Geiger, "Giraffe: Representing scenes as compositional generative neural feature fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 453–11 464.

[3] Y. Xue, Y. Li, K. K. Singh, and Y. J. Lee, "Giraffe hd: A high-resolution 3d-aware generative model," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 440–18 449.

[4] Y. Chen, F. Rong, S. Duggal, S. Wang, X. Yan, S. Manivasagam, S. Xue, E. Yumer, and R. Urtasun, "Geosim: Realistic video simulation via geometry-aware composition for self-driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 7230–7240.

[5] Y. Xu, M. Chai, Z. Shi, S. Peng, I. Skorokhodov, A. Siarohin, C. Yang, Y. Shen, H.-Y. Lee, B. Zhou *et al.*, "Discoscene: Spatially disentangled generative radiance fields for controllable 3d-aware scene synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4402–4412.

[6] Z. Wu, T. Liu, L. Luo, Z. Zhong, J. Chen, H. Xiao, C. Hou, H. Lou, Y. Chen, R. Yang, Y. Huang, X. Ye, Z. Yan, Y. Shi, Y. Liao, and H. Zhao, "Mars: An instance-aware, modular and realistic simulator for autonomous driving," *CICAI*, 2023.

[7] Z. Yang, Y. Chen, J. Wang, S. Manivasagam, W.-C. Ma, A. J. Yang, and R. Urtasun, "Unisim: A neural closed-loop sensor simulator," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 1389–1399.

[8] J. Yang, B. Ivanovic, O. Litany, X. Weng, S. W. Kim, B. Li, T. Che, D. Xu, S. Fidler, M. Pavone *et al.*, "Emernerf: Emergent spatial-temporal scene decomposition via self-supervision," *arXiv preprint arXiv:2311.02077*, 2023.

[9] A. Raistrick, L. Lipson, Z. Ma, L. Mei, M. Wang, Y. Zuo, K. Kayan, H. Wen, B. Han, Y. Wang *et al.*, "Infinite photorealistic worlds using procedural generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 630–12 641.

[10] C. Sun, J. Han, W. Deng, X. Wang, Z. Qin, and S. Gould, "3d-gpt: Procedural 3d modeling with large language models," *arXiv preprint arXiv:2310.12945*, 2023.

[11] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein, "pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5799–5809.

[12] K. Tertikas, D. Paschalidou, B. Pan, J. J. Park, M. A. Uy, I. Emiris, Y. Avrithis, and L. Guibas, "Generating part-aware editable 3d shapes without 3d supervision," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[13] J. Gu, L. Liu, P. Wang, and C. Theobalt, "StyleneRF: A style-based 3d aware generator for high-resolution image synthesis," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=iUuzzTMUw9K

[14] E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis *et al.*, "Efficient geometry-aware 3d generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 123–16 133.

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

[16] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987. [Online]. Available: https://api.semanticscholar.org/CorpusID:15545924

[17] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=FjNys5c7VyY

[18] Y. Shi, P. Wang, J. Ye, L. Mai, K. Li, and X. Yang, "MVDream: Multi-view diffusion for 3d generation," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=FUgrjq2pbB

[19] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin, "Magic3d: High-resolution text-to-3d content creation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 300–309.

[20] J. Tang, T. Wang, B. Zhang, T. Zhang, R. Yi, L. Ma, and D. Chen, "Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior," *arXiv preprint arXiv:2303.14184*, 2023.

[21] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, 2023.

[22] J. Tang, J. Ren, H. Zhou, Z. Liu, and G. Zeng, "Dreamgaussian: Generative gaussian splatting for efficient 3d content creation," *arXiv preprint arXiv:2309.16653*, 2023.

[23] T. Yi, J. Fang, J. Wang, G. Wu, L. Xie, X. Zhang, W. Liu, Q. Tian, and X. Wang, "Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models," in *CVPR*, 2024.

[24] J. Gao, T. Shen, Z. Wang, W. Chen, K. Yin, D. Li, O. Litany, Z. Gojcic, and S. Fidler, "Get3d: A generative model of high quality 3d textured shapes learned from images," *Advances In Neural Information Processing Systems*, vol. 35, pp. 31 841–31 854, 2022.

[25] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[26] T. Wu, J. Zhang, X. Fu, Y. Wang, J. Ren, L. Pan, W. Wu, L. Yang, J. Wang, C. Qian *et al.*, "Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 803–814.

[27] T. Groueix, M. Fisher, V. G. Kim, B. C. Russell, and M. Aubry, "A papier-mâché approach to learning 3d surface generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 216–224.

[28] Y. Zheng, L. Wu, X. Liu, Z. Chen, Q. Liu, and Q. Huang, "Neural volumetric mesh generator," 2022.

[29] L. Gao, J. Yang, T. Wu, Y.-J. Yuan, H. Fu, Y.-K. Lai, and H. Zhang, "Sdm-net: Deep generative network for structured deformable mesh," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–15, 2019.

[30] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 40–49. [Online]. Available: https://proceedings.mlr.press/v80/achlioptas18a.html

[31] D. Valsesia, G. Fracastoro, and E. Magli, "Learning localized generative models for 3d point clouds via graph convolution," in *International conference on learning representations*, 2018.

[32] D. W. Shu, S. W. Park, and J. Kwon, "3d point cloud generative adversarial network based on tree structured graph convolutions," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3859–3868.

[33] G. Yang, X. Huang, Z. Hao, M.-Y. Liu, S. Belongie, and B. Hariharan, "Pointflow: 3d point cloud generation with continuous normalizing flows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 4541–4550.

[34] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis, "Lion: Latent point diffusion models for 3d shape generation," *arXiv preprint arXiv:2210.06978*, 2022.

[35] R. Cai, G. Yang, H. Averbuch-Elor, Z. Hao, S. Belongie, N. Snavely, and B. Hariharan, "Learning gradient fields for shape generation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[36] S. Mo, E. Xie, R. Chu, L. Hong, M. Niessner, and Z. Li, "Dit-3d: Exploring plain diffusion transformers for 3d shape generation," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[37] S. Luo and W. Hu, "Diffusion probabilistic models for 3d point cloud generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2837–2845.

[38] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, and S. Savarese, "Topnet: Structural point cloud decoder," in *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.
[39] R. Zhang, J. Chen, W. Gao, G. Li, and T. H. Li, "Pointot: Interpretable geometry-inspired point cloud generative model via optimal transport," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 10, pp. 6792–6806, 2022.
[40] T. Kimura, T. Matsubara, and K. Uehara, "Topology-aware flow-based point cloud generation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 11, pp. 7967–7982, 2022.
[41] Y. Zhong, Y. Qi, Y. Gryaditskaya, H. Zhang, and Y.-Z. Song, "Towards practical sketch-based 3d shape generation: The role of professional sketches," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 9, pp. 3518–3528, 2021.
[42] R. Or-El, X. Luo, M. Shan, E. Shechtman, J. J. Park, and I. Kemelmacher-Shlizerman, "Stylesdf: High-resolution 3d-consistent image and geometry generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 503–13 513.
[43] X.-Y. Zheng, Y. Liu, P.-S. Wang, and X. Tong, "Sdf-stylegan: Implicit sdf-based stylegan for 3d shape generation," in *Comput. Graph. Forum (SGP)*, 2022.
[44] X.-Y. Zheng, H. Pan, P.-S. Wang, X. Tong, Y. Liu, and H.-Y. Shum, "Locally attentional sdf diffusion for controllable 3d shape generation," *arXiv preprint arXiv:2305.04461*, 2023.
[45] L. Yariv, O. Puny, N. Neverova, O. Gafni, and Y. Lipman, "Mosaic-sdf for 3d generative models," *arXiv preprint arXiv:2312.09222*, 2023.
[46] Z. Cao, F. Hong, T. Wu, L. Pan, and Z. Liu, "Large-vocabulary 3d diffusion model with transformer," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=q57JLSE2j5
[47] Z. Chen, X. Xu, Y. Yan, Y. Pan, W. Zhu, W. Wu, B. Dai, and X. Yang, "Hyperstyle3d: Text-guided 3d portrait stylization via hypernetworks," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2024.
[48] X. Zhu, J. Zhou, L. You, X. Yang, J. Chang, J. J. Zhang, and D. Zeng, "Dfie3d: 3d-aware disentangled face inversion and editing via facial-contrastive learning," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2024.
[49] B. Zhang, Y. Cheng, J. Yang, C. Wang, F. Zhao, Y. Tang, D. Chen, and B. Guo, "Gaussiancube: Structuring gaussian splatting using optimal transport for 3d generative modeling," *arXiv preprint arXiv:2403.19655*, 2024.
[50] J.-M. Sun, T. Wu, and L. Gao, "Recent advances in implicit representation-based 3d shape generation," *Visual Intelligence*, vol. 2, 03 2024.
[51] W. Chen, J. Gao, H. Ling, E. Smith, J. Lehtinen, A. Jacobson, and S. Fidler, "Learning to predict 3d objects with an interpolation-based differentiable renderer," in *Advances In Neural Information Processing Systems*, 2019.
[52] W. Chen, J. Litalien, J. Gao, Z. Wang, C. Fuji Tsang, S. Khamis, O. Litany, and S. Fidler, "Dib-r++: learning to predict lighting and material with a hybrid differentiable renderer," *Advances in Neural Information Processing Systems*, vol. 34, pp. 22 834–22 848, 2021.
[53] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang, "4d gaussian splatting for real-time dynamic scene rendering," *arXiv preprint arXiv:2310.08528*, 2023.
[54] Z. Xu, S. Peng, H. Lin, G. He, J. Sun, Y. Shen, H. Bao, and X. Zhou, "4k4d: Real-time 4d view synthesis at 4k resolution," *arXiv preprint arXiv:2310.11448*, 2023.
[55] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8110–8119.
[56] S. Luo and W. Hu, "Diffusion probabilistic models for 3d point cloud generation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 2837–2845.
[57] L. Wu, D. Wang, C. Gong, X. Liu, Y. Xiong, R. Ranjan, R. Krishnamoorthi, V. Chandra, and Q. Liu, "Fast point cloud generation with straight flows," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 9445–9454.
[58] Z. Liu, Y. Feng, M. J. Black, D. Nowrouzezahrai, L. Paull, and W. Liu, "Meshdiffusion: Score-based generative 3d mesh modeling," *arXiv preprint arXiv:2303.08133*, 2023.

BIOGRAPHY SECTION

Haochen Yu received the B.E. degree of the Department of Computer Science and Technology from the University of Science and Technology Beijing in 2022. Currently, he is pursuing the Ph.D. degree from the University of Science and Technology Beijing. His research focuses on 3D controllable generation and simulation.

Weixi Gong received the B.E. degree of the Department of Computer Science and Technology from the University of Science and Technology Beijing in 2022. Currently, he is pursuing the master's degree from the University of Science and Technology Beijing. His research focuses on computer graphics and autonomous driving simulation.

Jiansheng Chen (Senior Member. IEEE) received the M.S. degree from the Department of Computer Science and Technology, Tsinghua University, in 2002, and the Ph.D. degree in computer science and technology from The Chinese University of Hong Kong in 2006. He is currently a Professor with the School of Computer and Communication Engineering, University of Science and Technology Beijing. His research and teaching interests include computer vision and machine learning.

Huimin Ma (Senior Member. IEEE) is a professor at the School of Computer and Communication Engineering, University of Science and Technology Beijing. She received the Ph.D. degree from Beijing Institute of Technology in 2001. She was an associate professor at Tsinghua University from 2006 to 2019. She is now the vice dean of the School of Computer and Communication Engineering at USTB. She is also the secretary-general of the China Society of Image and Graphics. Her research interests include 3D image cognition and simulation. She has published extensively in refereed scientific journals (TPAMI, TIP, etc.) and international conferences (CVPR, NIPS, etc.).