Trip-to-Gaussian: A Versatile Framework for Unconditional 3D Generation

Anonymous authors
Paper under double-blind review

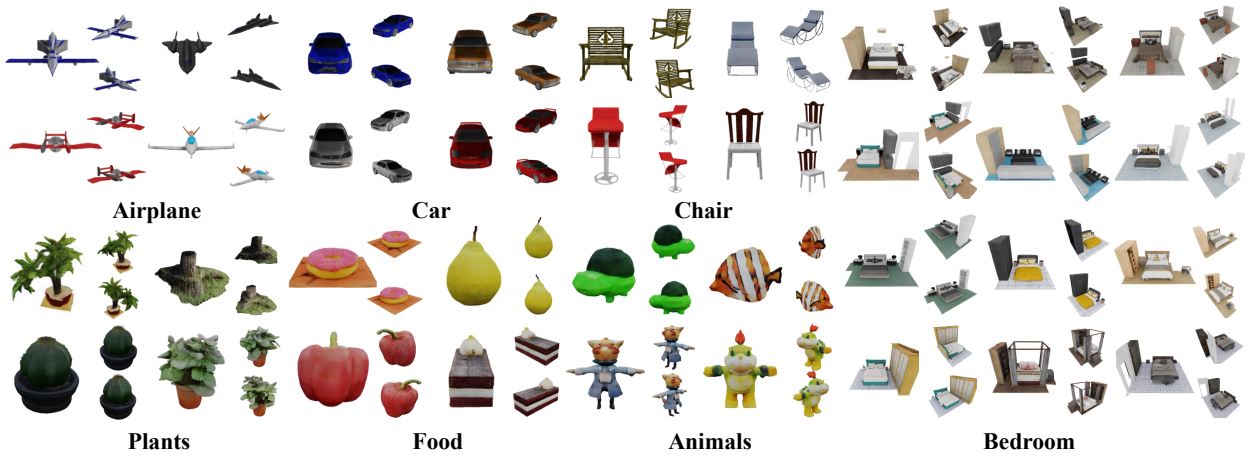


Figure 1: **Gallery of unconditional 3D generation results on ShapeNet (Chang et al., 2015), G-Objaverse (Deitke et al., 2022), and 3D-FRONT (Fu et al., 2021a) datasets.** `Trip2GS` enables versatile 3D generation from objects to scenes, producing geometrically coherent and high-fidelity 3D Gaussians.

Abstract

Unconditional 3D generation is a classical task, which explores effective network architecture to learn the underlying distribution of 3D assets. However, most existing methods limits in versatility, struggling to scale from object- to scene-level generation. Achieving such versatility critically depends on how 3D representations are designed in the latent and output spaces, and how these spaces are connected. In this work, we focus on leveraging the expressiveness of triplane together with the fast and high-fidelity 3D Gaussian Splatting (3DGS). Yet, integrating these two representations remains a challenge due to their fundamentally different natures – the structured triplane and unstructured 3DGS. Our core idea is a coarse-to-fine generation scheme that first extracts reliable geometric priors from triplane and subsequently refines them to capture detailed geometry and textures through 3D Gaussians. To this end, we introduce `Trip-to-Gaussian`, a versatile 3D generation framework that seamlessly integrates two distinct representations. We propose a Gaussian indicator module (GIM) along with surface occupancy fields (SOF), generating coarse anchor points, which serves as a reliable geometric prior for 3D Gaussians. Building upon this, we present a point upsampling module (PUM) that maps discontinuous and coarse anchor points into a continuous space, densifying them to ensure fine-grained representation. Extensive experiments demonstrate that our approach outperforms recent methods in both unconditional object and scene generation, establishing a versatile paradigm for 3D generation.

Table 1: **Comparison of recent unconditional 3D Gaussian generation methods.** Prior works require additional 3DGS fitting stage and lack coarse-to-fine design, limiting scene-level scalability. In contrast, `Trip2GS` employs a coarse-to-fine generation scheme to generate 3DGS from triplane features, achieving versatile 3D generation.

Method	Conference	3DGS Fitting Stage	Latent Representation	Coarse-to-Fine	Scene-level
DiffGS (Zhou et al., 2025)	NeurIPS' 24	Required	Vector	×	×
GaussianCube (Zhang et al., 2024)	NeurIPS' 24	Required	–	×	×
L3DG (Roessle et al., 2024)	SIGGRAPH Asia' 24	Required	Voxel Grid	×	△
AtlasGaussians (Yang et al., 2025)	ICLR' 25	Not Required	Point cloud	×	×
DirectTriGS (Ju & Li, 2025)	CVPR' 25	Required	Triplane	×	×
UVGS (Rai et al., 2025)	CVPR' 25	Required	Image	×	×
Trip2GS (Ours)	–	Not Required	Triplane	✓	✓

1 Introduction

Unconditional 3D generation aims to generate realistic samples which follow the learned 3D data distribution, revealing the fundamental generative capacity of a model and laying the groundwork for subsequent advances in conditional generation. Unlike the rapidly evolving field of conditional generation (Xiang et al., 2025; Zhao et al., 2025; Li et al., 2025; Chen et al., 2025), which benefits from ever-increasing data scales and computational resources, unconditional generation focuses on designing networks that maximize generation quality under constrained data and computational budgets. Recent advances in unconditional 3D generation have shown remarkable progress in producing high-quality textured objects (Gao et al., 2022; Müller et al., 2023; Cao et al., 2024; Lan et al., 2024a; Zhou et al., 2025; Zhang et al., 2024; Yang et al., 2025; Rai et al., 2025), yet most previous studies have been limited in *versatility*: scaling from object- to scene-level generation. Scene-level generation is crucial for real-world applications; however, most existing works struggle with scene-level tasks due to their limited capacity and design of the framework.

To design a versatile and high-quality unconditional generative model, there are a few key challenges. 1) *Output representation:* Joint distribution of 3D geometry and texture is complex. This motivates our use of 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023), which naturally encodes geometry and texture jointly. 2) *Latent representation:* Learning a compact, expressive latent for 3D data is crucial. We address this with the efficient and versatile triplane representation (Chan et al., 2022; Peng et al., 2020; Shue et al., 2023). 3) *Bridging representations:* Mapping from a triplane latent to 3DGS remains a challenging, underexplored problem because triplane has a regular grid structure, whereas 3DGS is an irregular and unstructured representation.

In this study, we propose `Trip-to-Gaussian` (`Trip2GS` in short), a novel 3D generation framework that achieves unconditional generation scalable from objects to scenes, without requiring additional data or computation (see Fig. 1). Our framework is motivated by the observation that existing unconditional 3D Gaussian generation methods either model the complex 3DGS distribution (Zhang et al., 2024; Zhou et al., 2025; Rai et al., 2025) or directly generate 3DGS representations from latent features (Zhou et al., 2025; Yang et al., 2025; Ju & Li, 2025). These approaches often struggle to preserve thin structures and fine geometric details, especially when extended to 3D scenes with complicated geometry. To overcome these limitations, we introduce a coarse-to-fine generation scheme as a key to achieving versatility. We first construct a coarse but reliable geometric prior from triplane, and then progressively refine it to capture fine-grained geometry and texture details. This design enables effective modeling of complex 3D structures and seamlessly bridges the structural gap between triplane and 3DGS representations, yielding a versatile solution for unconditional 3D generation.

In particular, `Trip2GS` comprises two main stages: a Trip2GS VAE and a triplane diffusion model. The Trip2GS VAE encodes complex 3D geometry and appearance into a compact triplane latent and decodes it into 3D Gaussians through a coarse-to-fine process. At its core lies the Trip2GS decoder, which comprises three main components: Gaussian indicator module (GIM), which extracts coarse anchor points from the triplane; point upsampling module (PUM), which generates refined anchor points by densifying and smoothing the coarse anchor points; and the 3DGS decoder, which predicts the final Gaussian parameters. The

triplane diffusion then learns the latent distribution, enabling diverse and high-quality generation. Notably, our framework directly generates 3D Gaussians without any external fitting stage or optimization process. Quantitative and qualitative results across multiple datasets demonstrate that `Trip2GS` outperforms existing unconditional generation methods in generation quality and geometric fidelity.

Our key contributions are as follows:

- We propose `Trip2GS`, a versatile 3D generation framework that extends the capability of unconditional generation by seamlessly bridging structured triplane and unstructured 3DGS representations.
- We introduce a coarse-to-fine generation strategy, where GIM extracts coarse yet reliable geometric priors and PUM progressively refines them into high-quality geometry and textures.
- We present SOF, a representation that enables effective 3D Gaussian generation from triplane features, surpassing conventional SDF-based approaches.
- Extensive experiments demonstrate that `Trip2GS` consistently outperforms prior works in both unconditional object and scene generation.

2 Related Work

Latent Representations for 3D Generation. The design of 3D generative models is closely tied to the choice of intermediate representation. Vector-based methods (Chou et al., 2023; Yariv et al., 2024; Zhou et al., 2025) lack geometric priors and often produce low-quality structures. Voxel-based methods (Shim et al., 2023; Müller et al., 2023; Liu et al., 2023; Zhang et al., 2024; Roessle et al., 2024) intuitively encode 3D information but are limited by prohibitive memory costs at high resolution. Point cloud-based approaches (Zhou et al., 2021; Vahdat et al., 2022; Yang et al., 2025; Lan et al., 2024b) capture geometry well but introduce complexity due to their irregular structure.

In contrast, triplane-based methods (Gao et al., 2022; Gupta et al., 2023; Shue et al., 2023; Lan et al., 2024a; Zhou et al., 2025; Ju & Li, 2025; Shim & Joo, 2024; Cao et al., 2025) can leverage CNN geometric priors through a regular grid while maintaining efficient memory usage. Recent works (Wu et al., 2024b; Lee et al., 2024; Yan et al., 2024; Meng et al., 2025) demonstrate the expressiveness of triplane for generating detailed scene geometry. Building on these strengths, we adopt triplane as our latent representation, enabling `Trip2GS` to efficiently generate high-quality textured 3D objects and scenes.

3DGS Generation with Grid Representations. Recently, 3DGS has emerged as a powerful 3D primitive representation, but generating unstructured 3DGS from a structured grid latent is challenging.

Image-based approaches such as Splatter Image (Szymanowicz et al., 2024) and subsequent works (Shen et al., 2024; Tang et al., 2024; Lin et al., 2025; Rai et al., 2025) treat each image pixel as a Gaussian element, allowing fast and lightweight 3DGS generation. While efficient, these methods lack a 3D structural prior, leading to inferior geometric consistency. Voxel-based approaches provide stronger 3D priors. GaussianCube (Zhang et al., 2024) parameterizes each voxel grid cell as a Gaussian, but its dense voxelization constrains the number of Gaussians and wastes model capacity on learning unnecessary offsets. Another work, L3DG (Roessle et al., 2024) further leverages a sparse voxel grid to encode 3D Gaussians in a structured manner and demonstrates 3D scene generation similar to ours; however, without quantitative evaluation or publicly available code, its versatility remains uncertain. Triplane-based approaches exploit the 2D-structured triplane latent for efficient and robust 3D generation. Unlike conditional generation methods (Zou et al., 2024; Jiang et al., 2024; Barthel et al., 2024a), which obtain geometric information from image or text priors, unconditional generation requires extracting geometry directly from the triplane without any external guidance, making the generation of 3D Gaussians substantially more challenging. Several works (Zhou et al., 2025; Ju & Li, 2025) have attempted to generate 3DGS from triplane for unconditional generation, but still suffer from limited geometric fidelity. In contrast to the stochastic sampling of DiffGS (Zhou et al., 2025) and the deformable SDF-based extraction of DirectTriGS (Ju & Li, 2025), we propose to generate 3D Gaussians in a coarse-to-fine manner, which preserves geometry fidelity and enables extension to scene-level generation.

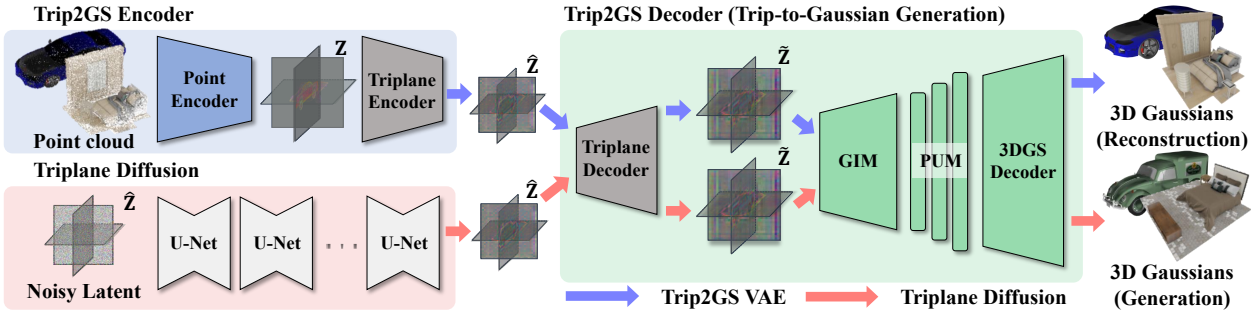


Figure 2: **The overall architecture of the proposed `Trip2GS` framework.** Trip2GS VAE comprises the Trip2GS encoder (blue box) and Trip2GS decoder (green box), which embed an input colored point cloud into a triplane latent representation and then generate 3D Gaussians. Triplane diffusion (red box), trained on triplane latents, generates new triplane latents that are subsequently decoded into 3D Gaussians using the trained Trip2GS decoder.

The most related work is TRELLIS (Xiang et al., 2025), which introduces a structured latent (SLat) by constructing a sparse voxel grid and mapping DINOv2 (Oquab et al., 2024) image features into the voxel space. While TRELLIS have shown both object- and scene-level generation, it trains four independent generative models on separately prepared datasets, resulting in a complex data construction and non–end-to-end pipeline. In contrast, `Trip2GS` parameterizes triplanes into SOF within a single decoding process, removing additional modeling stages. Furthermore, our 2D triplane representation maintains a lightweight architecture with fewer parameters and more efficient 2D operations than the 3D voxel-based SLat.

Tab. 1 compares recent unconditional 3D Gaussian generation methods. `Trip2GS` is the first framework to introduce a coarse-to-fine scheme, surpassing prior approaches.

3 Method

3.1 Overview

The proposed `Trip2GS` mainly consists of three parts: Trip2GS encoder/decoder and triplane diffusion (see Fig. 2). Here, we outline each part of our framework and then discuss the details in the following.

Trip-to-Gaussian Encoder. The Trip2GS encoder encodes input to a low-dimensional latent triplane. Given a colored point cloud $\mathbf{X} \in \mathbb{R}^{N\times(3+3)}$, where N is the number of points, we first construct a high-resolution triplane feature by projecting the point features onto three orthogonal planes using PointNet (Qi et al., 2017)-based encoder, similar to ConvONet (Peng et al., 2020). This makes a high-resolution triplane feature $\mathbf{Z} \in \mathbb{R}^{3\times R^2\times C}$, where R is the resolution of each plane, and C is the channel size (we set $R = 256$, $C = 32$). Subsequently, $\mathbf{Z}$ is compressed into a perceptually rich low-dimensional latent triplane $\hat{\mathbf{Z}} \in \mathbb{R}^{3\times \hat{R}^2\times \hat{C}}$ using our triplane encoder, equipped with pooling-based 3D-aware convolutions (Wang et al., 2023; Wu et al., 2024a), where $\hat{R}$ and $\hat{C}$ denote the resolution and channel size of the latent triplane, respectively (we set $\hat{R} = 32$, $\hat{C} = 4$).

It should be noted that to generate 3D Gaussians, we directly encode colored point cloud instead of 3DGS, avoiding the time/memory-consuming 3DGS fitting stage.

Trip-to-Gaussian Decoder. The Trip2GS decoder generates 3DGS from a triplane feature $\tilde{\mathbf{Z}}$, where we exploit a coarse-to-fine strategy to bridge the gap between triplane and 3DGS representations. Concretely, we upsample $\hat{\mathbf{Z}}$ into a high-resolution triplane feature $\tilde{\mathbf{Z}} \in \mathbb{R}^{3\times R^2\times C}$ using a triplane decoder to restore detailed 3D geometry and texture. We project $\tilde{\mathbf{Z}}$ into surface occupancy fields (SOF), our newly introduced concept designed for seamless extraction of 3D Gaussians from the triplane, which implicitly describes near-surface regions. We then regularly sample SOF in 3D space to extract coarse anchor points. Next, we densify and smooth the sparse and discrete coarse anchor points into upsampled ones. Finally, we predict

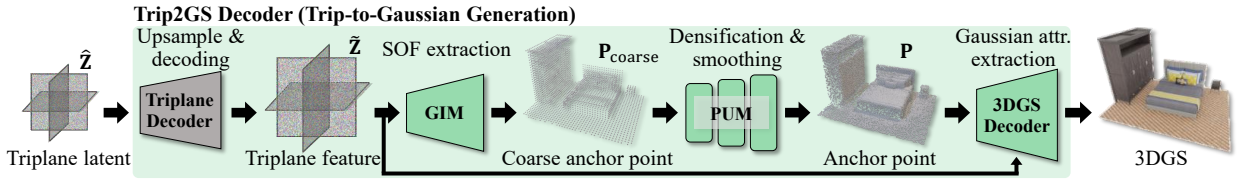


Figure 3: **Illustration of the Trip2GS decoder.** We decode the triplane latent $\hat{\mathbf{Z}}$ into a high-resolution triplane feature $\tilde{\mathbf{Z}}$ using our triplane decoder. Then, GIM generates coarse anchor points $\mathbf{P}_{\text{coarse}}$ from SOF. Next, PUM refines $\mathbf{P}_{\text{coarse}}$ into anchor points $\mathbf{P}$ by increasing their density and smoothness. Finally, the 3DGS decoder extracts Gaussian attributes by sampling $\tilde{\mathbf{Z}}$ at $\mathbf{P}$.

Gaussian parameters by extracting triplane features at each upsampled anchor point to construct the 3DGS representation.

Triplane Diffusion. Triplane diffusion is a variant of the latent diffusion model (Rombach et al., 2022) that generates a new triplane latent $\hat{\mathbf{Z}}$. The network architecture of triplane diffusion is based on a U-Net (Ronneberger et al., 2015) with 3D-aware convolutions (Wang et al., 2023; Wu et al., 2024a) and self-attention blocks (Vaswani et al., 2017), designed to effectively learn the distribution of triplane latents.

3.2 Trip-to-Gaussian Generation

To generate 3DGS, we need to extract 3D Gaussians from the triplane feature $\tilde{\mathbf{Z}}$ decoded by the triplane decoder. A key challenge here is that triplane and 3DGS differ in nature. Prior approaches such as optimization- or SDF-guided sampling (Zhou et al., 2025; Ju & Li, 2025) only loosely couple the two representations, resulting in degraded geometry fidelity and limited scalability to complex scene-level generation. To address this limitation, we explicitly decompose this problem into a coarse-to-fine generation process (see Fig. 3). We first extract a coarse yet reliable geometry prior from the triplane, which is then progressively refined to recover fine-grained geometry and high-quality textures. This decomposition effectively bridges the structural gap between the triplane and 3DGS, while enabling versatile unconditional 3D generation from objects to scenes.

To achieve this, we propose two complementary modules. The Gaussian indicator module (GIM) extracts coarse anchor points for 3D Gaussians by parameterizing triplane features into surface occupancy fields (SOF). The point upsampling module (PUM) then densifies and smooths these sparse and discrete anchor points to upsampled anchor points. Finally, the 3DGS decoder estimates the full Gaussian parameters from these refined anchor points and triplane features.

Gaussian Indicator Module. GIM parameterizes the triplane feature $\tilde{\mathbf{Z}}$ into SOF, a variant of occupancy fields (Mescheder et al., 2019; Peng et al., 2020), to acquire unstructured 3D Gaussians from structured triplane features. SOF estimates the probability of 3D Gaussian existence near the surface by assigning 1 for near-surface regions, unlike traditional occupancy fields that serve inside-outside indicators. Subsequently, we sample surface occupancy at regular grid points from the SOF to construct a Gaussian probability volume $\mathbf{V} \in \mathbb{R}^{R_{\text{voxel}}^3}$, a voxel-shaped probability volume with resolution R_{voxel}. Then, we extract points by filtering voxel cells that

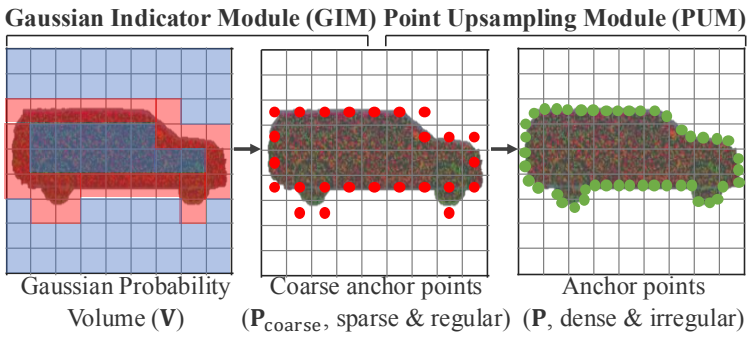


Figure 4: **Illustration of the GIM and PUM processes in 2D.** GIM builds Gaussian probability volume $\mathbf{V}$ by regularly sampling SOF in 3D space. Then, it extracts high-probability regions to generate coarse anchor points $\mathbf{P}_{\text{coarse}}$. Finally, PUM densifies and smooths $\mathbf{P}_{\text{coarse}}$ to produce final anchor points $\mathbf{P}$.

exceed threshold probability to generate coarse anchor points $\mathbf{P}_{\text{coarse}} \in \mathbb{R}^{M_{\text{coarse}} \times 3}$, where M_{coarse} is the number of coarse anchor points. For clarity, we present a 2D illustration of the GIM process in Fig. 4.

The proposed SOF representation reformulates Gaussian position prediction as a binary classification, improving 3DGS decoding quality by reducing the learning burden on the network. By using SOF, we also eliminate computationally expensive and non-differentiable steps like marching cubes (Lorensen & Cline, 1987). Further discussion on SOF in comparison to other representations is provided in the ablation study.

Point Upsampling Module. 3DGS requires hundreds of thousands of Gaussians to represent an object or a scene with detailed geometry and texture. However, the extracted anchor points from GIM are sparse, limiting the ability to capture fine details. We also observed that directly querying triplane features at regular anchor points from GIM causes unexpected artifacts, such as grid pattern and texture abruption in rendering results, which degrade the visual quality (refer to appendix).

To address these issues, we present an MLP-based PUM that maps discrete points to a continuous space and densifies the anchor points, both crucial for high-quality 3D Gaussian generation. Specifically, PUM refines coarse anchor points $\mathbf{P}_{\text{coarse}}$ into anchor points $\mathbf{P} \in \mathbb{R}^{M \times 3}$ by performing upsampling and nonlinear mapping, where M is the number of anchor points after upsampling, satisfying $M \geq M_{\text{coarse}}$. As a result, we can generate densified and continuous anchor points for 3D Gaussians, enhancing the geometric accuracy and texture quality of 3DGS (we visualize a 2D illustration of the PUM process in Fig. 4).

3DGS Decoder. To generate realistic textures, we present a 3DGS decoder to estimate the Gaussian parameters for each upsampled anchor point from PUM. Concretely, the 3DGS decoder first aggregates the triplane feature $\tilde{\mathbf{Z}}$ of the corresponding anchor point by extracting features from each of the three planes. Next, it predicts Gaussian parameters by leveraging per-attribute-specific fully connected layers, inspired by (Barthel et al., 2024b; Lu et al., 2024; Zou et al., 2024). Each 3D Gaussian is parameterized by a 14-dimensional attribute vector composed of spatial offset, color, opacity, and a covariance matrix that defines the shape and orientation of the Gaussian. This structured approach enables a more effective disentanglement of spatial and appearance features, resulting in more precise and visually coherent 3D Gaussian representations.

3.3 Training Objective

We train our VAE with three primary loss terms: rendering loss $\mathcal{L}_{\text{render}}$, geometry loss $\mathcal{L}_{\text{geo}}$, and regularization $\mathcal{L}_{\text{reg}}$:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{render}} + \mathcal{L}_{\text{geo}} + \mathcal{L}_{\text{reg}}. \tag{1}$$

By optimizing $\mathcal{L}_{\text{total}}$, our Trip2GS VAE learns to generate high-quality 3DGS while preserving geometric structure and ensuring a smooth and stable latent space.

Rendering Loss. We follow the original 3DGS (Kerbl et al., 2023) to ensure high-fidelity 3DGS reconstruction from colored point cloud input, applying pixel-wise L2 loss $\mathcal{L}_{\text{L2}}$ and SSIM loss $\mathcal{L}_{\text{SSIM}}$. In addition, we adopt a mask loss $\mathcal{L}_{\text{mask}}$ for silhouette-based supervision, and LPIPS (Zhang et al., 2018) loss $\mathcal{L}_{\text{LPIPS}}$ for perceptual quality. The rendering loss is:

$$\mathcal{L}_{\text{render}} = \mathcal{L}_{\text{L2}} + \mathcal{L}_{\text{SSIM}} + \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{LPIPS}}. \tag{2}$$

Geometry Loss. We employ two additional loss terms for geometry loss. First, we supervise GIM using binary cross-entropy loss $\mathcal{L}_{\text{BCE}}$ to ensure that GIM predicts correct surface occupancy fields:

$$\mathcal{L}_{\text{GIM}} = \mathcal{L}_{\text{BCE}}(\hat{v}_i, v_i), \tag{3}$$

where $\hat{v}_i \in \hat{\mathbf{V}}$ and $v_i \in \mathbf{V}$ are predicted and the ground truth Gaussian existence probability, respectively. Second, we apply Chamfer distance $\mathcal{L}_{\text{CD}}$ to compare ground truth surface geometry (*i.e.*, input point cloud) with both the anchor points from PUM and Gaussian positions from the 3DGS decoder. The geometry loss is formulated as:

$$\mathcal{L}_{\text{geo}} = \lambda_{\text{GIM}} \mathcal{L}_{\text{GIM}} + \lambda_{\text{CD}} \mathcal{L}_{\text{CD}}, \tag{4}$$

where λ_{GIM} and λ_{CD} are set to 1 and 100, respectively.

Regularization. To stabilize the diffusion training and encourage visually smooth rendering, we adopt total variation (TV) regularization (Rudin & Osher, 1994) on both latent space and image space, reducing undesired high-frequency artifacts. Additionally, KL divergence regularization $\mathcal{L}_{\text{KL}}$ is applied to constrain the latent space to follow a Gaussian distribution, which also accelerates the diffusion training process. The regularization term is formulated as:

$$\mathcal{L}_{\text{reg}} = \lambda_{\text{TV}}\mathcal{L}_{\text{TV}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}, \tag{5}$$

where λ_{TV} and λ_{KL} are set to $1e{-}4$ and $1e{-}6$, respectively.

3.4 Triplane Diffusion

Unlike conventional diffusion models (Ho et al., 2020), our triplane diffusion is supervised to estimate denoised latent triplane $\hat{\mathbf{Z}}_0$ instead of the applied noise, following (Zhang et al., 2024):

$$\mathcal{L}_{\text{diff}} = \left\| \mu_\theta(\hat{\mathbf{Z}}_t, t) - \hat{\mathbf{Z}}_0 \right\|_2, \tag{6}$$

where μ_θ is our triplane diffusion network, t is diffusion timestep. We use a linear noise schedule combined with the Heun sampler from (Karras et al., 2022), which allows fast convergence and high-quality sampling.

For the denoising network, we employ a U-Net (Ronneberger et al., 2015) based architecture following the design of ADM (Nichol & Dhariwal, 2021; Dhariwal & Nichol, 2021). We adopt ResNet (He et al., 2016) block consisted of 3D-aware operations (Wang et al., 2023; Wu et al., 2024a) same as triplane encoder of Trip2GS VAE, followed by self-attention block applied to channel-wise concatenated triplanes. By training triplane diffusion, we generate high-fidelity triplane latents, which are subsequently decoded by the Trip2GS decoder to produce diverse and high-quality 3D objects and scenes. Further implementation details can be found in appendix.

4 Experiments

We conduct various experiments to demonstrate the effectiveness and versatility of `Trip2GS`. Sec. 4.1 describes the datasets, evaluation metrics, and baseline models used for comparison. In Sec. 4.2, we evaluate our method for unconditional 3D generation across multiple datasets and demonstrate that it outperforms existing baselines both qualitatively and quantitatively. Sec. 4.3 provides an ablation study on the key components of Trip2GS VAE, analyzing their contributions to high-quality 3DGS generation. *It is strongly recommended to refer to appendix for additional results and evaluations, including conditional generation and comparison with TRELLIS (Xiang et al., 2025).*

4.1 Dataset, Metrics, and Baselines

Dataset. We adopt two datasets for object-level generation. ShapeNetv1 (Chang et al., 2015) serves as a widely used benchmark dataset in unconditional 3D generation, providing clean and category-consistent 3D objects. G-Objaverse (Zuo et al., 2024; Qiu et al., 2024; Deitke et al., 2022; 2023) is a large-scale collection of diverse and high-quality 3D assets, allowing us to evaluate the versatility on detailed geometries and textures. For scene-level generation, we employ 3D-FRONT (Fu et al., 2021a;b), consisting of synthetic indoor scenes with various room layouts, furniture, and styles, to assess the generation capability on complex scenes.

Metrics. We use two widely adopted metrics in unconditional 3D generation: FID@50K (Heusel et al., 2017) and KID@50K (Bińkowski et al., 2018). These metrics measure the distributional similarity between 50K rendered images from generated samples and ground-truth images. We also use CLIP-FID@50K (Parmar et al., 2022) (abbreviated as C-FID) as a more semantically-aware measurement.

Baselines. As baselines, we select state-of-the-art methods that are capable of generating both geometry and texture while supporting training on high-resolution images. GET3D (Gao et al., 2022) is a GAN-based approach that uses DMTet (Shen et al., 2021) for high-resolution 3D generation, often outperforming

Table 2: **Quantitative comparison with other methods on ShapeNet and 3D-FRONT using FID, KID, and C-FID metrics.** We depict cases with excessively large values (> 200) using a triangle (△).

Method	Chair			Airplane			Car			Bedroom		
	FID	**KID**	**C-FID**	**FID**	**KID**	**C-FID**	**FID**	**KID**	**C-FID**	**FID**	**KID**	**C-FID**
GET3D	75.89	48.71	15.23	72.61	57.66	7.18	109.35	83.83	14.83	△	△	60.54
DiffGS	75.21	55.45	17.42	65.18	52.76	10.57	128.01	114.64	14.94	△	△	58.87
GaussianCube	23.45	14.40	6.24	17.84	11.92	3.28	20.92	13.26	4.43	96.78	97.33	22.19
Atlas Gaussians	**13.46**	4.61	2.72	12.87	6.34	2.76	19.79	12.23	3.36	74.86	62.69	15.18
`Trip2GS` (ours)	14.13	**3.97**	**1.96**	**12.24**	**5.76**	**1.28**	**18.63**	**9.56**	**2.54**	**16.30**	**9.10**	**2.87**

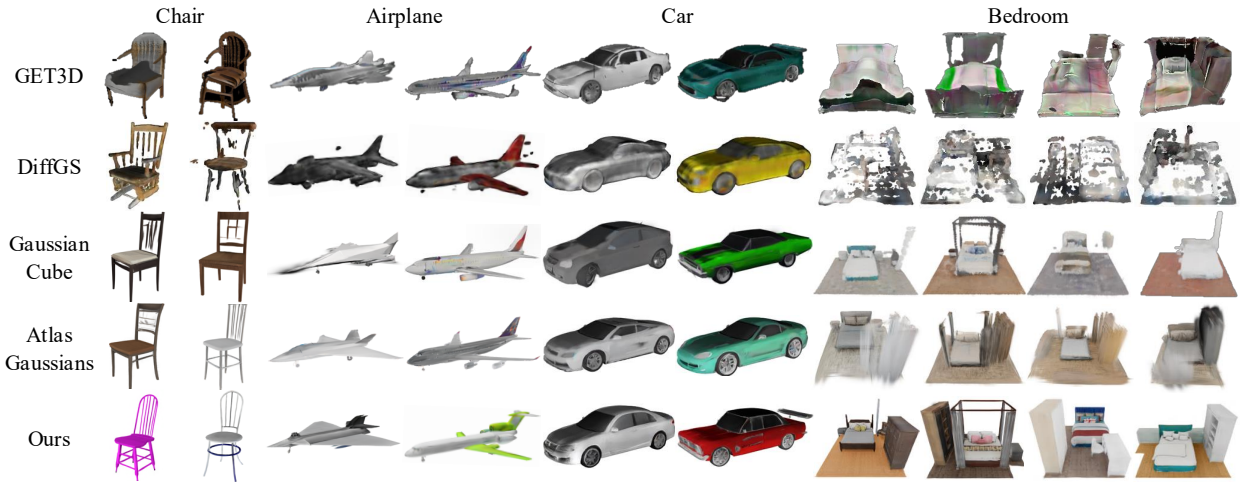


Figure 5: **Qualitative comparisons of unconditional generation on ShapeNet and 3D-FRONT.**

recent methods in various benchmarks. DiffGS (Zhou et al., 2025), GaussianCube (Zhang et al., 2024), and AtlasGaussians (Yang et al., 2025) utilize 3DGS as a 3D primitive. DiffGS encodes precomputed 3D Gaussians to triplane and subsequently compresses the triplane to a latent vector, while GaussianCube directly allocates 3D Gaussians to a fixed-size voxel grid (*e.g.*, 32). AtlasGaussians adopts a point cloud-based latent representation to capture the object geometry better.

4.2 Generation Results

3D Object Generation. Tab. 2 and Fig. 5 present results on three ShapeNet categories: chair, airplane, and car. Across all metrics, `Trip2GS` outperforms existing baselines, producing complete geometry and sharper textures. Notably, our model captures thin structures such as chair backs and legs, demonstrating that the coarse-to-fine design effectively preserves high-frequency geometric details.

We further evaluate `Trip2GS` on G-Objaverse, a large-scale dataset with diverse and high-quality 3D assets across plants, food, and animals categories (see Tab. 3 and Fig. 6). Our method consistently achieves superior visual quality even on complex 3D assets, accurately modeling intricate structures such as leaves and petals.

Overall results confirm that `Trip2GS` generalizes beyond simple shapes, capturing the distribution of diverse, high-quality 3D assets.

3D Scene Generation. To assess versatility to more complex geometries, we evaluate our method on the bedroom category of 3D-FRONT, which offers diverse layouts and styles with sufficient samples (3,886 after preprocessing following (Paschalidou et al., 2021)) for training an unconditional model (see Tab. 2 and Fig. 5). `Trip2GS` surpasses all SOTA baselines by a clear margin, both quantitatively and qualitatively.

Table 3: **Quantitative comparison with other methods on G-Objaverse dataset using FID, KID, and C-FID metrics.**

Method	Plants			Food			Animals		
	FID	KID	C-FID	FID	KID	C-FID	FID	KID	C-FID
GET3D	122.69	110.03	15.97	108.82	79.54	18.79	104.69	83.22	14.59
DiffGS	160.78	111.00	22.18	171.39	73.48	32.70	199.53	100.35	33.98
GaussianCube	127.39	98.15	17.07	96.03	67.98	15.46	131.05	99.38	16.32
Atlas Gaussians	89.70	56.80	13.20	73.85	43.41	13.96	87.98	65.22	12.83
`Trip2GS` (ours)	**67.09**	**46.56**	**7.75**	**55.21**	**31.91**	**10.45**	**67.25**	**49.74**	**8.31**

Figure 6: **Qualitative comparisons of unconditional generation on G-Objaverse.**

Methods such as GET3D and DiffGS, which rely on simple vector-shape latents, fail to capture the spatial complexity of scenes and produce unrealistic results. Although GaussianCube and AtlasGaussians show moderate results, they still struggle with fine object details and secondary elements such as wardrobes and lamps. In contrast, `Trip2GS` accurately reconstructs complex scene geometry, preserves spatial consistency, and generates realistic textures for both major and minor objects.

These results confirm that our coarse-to-fine design effectively models the complex 3D structure, providing a unified and versatile framework for unconditional 3D generation from individual objects to multi-object scenes.

4.3 Ablation Study

Intermediate Geometry Representation. The core of our coarse-to-fine generation scheme lies in extracting a reliable geometric prior through GIM. In this process, SOF plays a critical role in indicating the spatial regions where 3D Gaussians should exist, serving as an intermediate geometry representation between the structured triplane and unstructured 3DGS. To assess the effectiveness of our SOF, we compare with existing geometry representations commonly used in prior works: signed distance fields (SDF) and deep marching tetrahedra (DMTet) (Shen et al., 2021). For fair comparison, we evaluate a variant without PUM, where anchor points from GIM are directly used as query points for the 3DGS decoder. Since SDF requires computationally expensive marching cubes algorithm (Lorensen & Cline, 1987) to obtain the object surface, we train SDF and 3DGS decoder separately.

(a) SDF (b) DMTet (c) SOF

Figure 7: **Comparison with other geometry representations.** Our SOF demonstrates robustness in the reconstruction of sharp and thin shapes.

Table 4: **Comparison of geometry representations for chair category on ShapeNet test set.** Trip2GS VAE is trained for 300k iterations using each geometry representation.

Representation	PSNR	SSIM	LPIPS
SDF	20.26	0.8411	0.2111
DMTet	23.49	0.8960	0.1188
SOF (ours)	**24.52**	**0.9125**	**0.1005**

Unlike SDF, DMTet enables fast mesh extraction via the differentiable marching tetrahedra algorithm, allowing the predicted surface points to be used as query points for the 3DGS decoder during training. This approach is similar to DirectTriGS (Ju & Li, 2025), which employs FlexiCubes (Shen et al., 2023) to generate anchor points. However, DMTet still requires sampling points from the reconstructed mesh, making the overall pipeline partially non-differentiable and hindering end-to-end optimization. Moreover, extracting mesh solely for point sampling introduces unnecessary computational overhead, reducing the overall efficiency of the process. To address these limitations, we propose SOF, a fully differentiable and reliable intermediate geometry representation for 3DGS generation.

As shown in Tab. 4 and Fig. 7, our SOF achieves the highest reconstruction quality. Since the primary goal of GIM is to provide reliable surface regions where 3D Gaussians should exist, the key factor is the *fidelity* of the extracted geometry prior. Continuous representations such as SDF and DMTet are effective for smooth surface reconstruction, but they tend to oversmooth thin or high-frequency structures, leading to missing or disconnected regions (Fig. 7-(a), (b)). In contrast, the proposed SOF representation, trained with a classification objective, produces more stable surface region predictions even for thin geometries (Fig. 7-(c)). Although SOF generates fewer and discrete anchor points, their reliability enables superior 3D Gaussian generation, validating SOF as an effective intermediate representation for bridging triplane and 3DGS.

Table 5: **Ablation study on PUM for the ShapeNet airplane category after 500k iterations.** k denotes upsampling ratio.

Point mapping	k	FID	KID	CLIP-FID
X	1	35.42	27.86	2.53
O	1	23.11	13.65	2.02
O	4	15.86	8.74	1.69
O	16	**12.24**	**5.76**	**1.28**

Table 6: **Ablation study on TV regularization for the ShapeNet chair category after 1M training iterations.**

$\mathcal{L}_{\text{TV-image}}$	$\mathcal{L}_{\text{TV-latent}}$	FID	KID	CLIP-FID
		17.01	4.44	2.15
✓		15.63	4.28	2.06
	✓	14.78	4.15	2.02
✓	✓	**14.13**	**3.97**	**1.96**

Point Mapping and Upsampling. The placement and number of 3D Gaussians are crucial for capturing geometric and textural details. Since the anchor points extracted from GIM are discontinuous and sparse, we introduce PUM, which maps them into a continuous space and upsamples them into a denser point set aligned with the object surface. Tab. 5 shows that PUM consistently improves generation quality. Due to its lightweight MLP-based design, PUM can efficiently densify Gaussians with minimal computational overhead (refer to appendix).

Total Variation Regularization. Tab. 6 presents an ablation study analyzing the effect of TV regularization loss when applied to both image and latent spaces. Applying it to both domains consistently yields the best performance. We attribute this improvement to the sparsity of 3D space, where reducing noisy features in empty regions enhances stability and overall generation quality.

5 Conclusion

We have proposed `Trip2GS`, a versatile 3D generative framework that bridges structured triplane and unstructured 3D Gaussian Splatting (3DGS), enabling high-quality generation from individual objects to complex scenes. The core of our approach lies in a coarse-to-fine generation scheme that first extracts reliable geometric priors from triplane and progressively transforms them into 3DGS for detailed geometry and textures. To realize this, we proposed Gaussian indicator module (GIM) and surface occupancy fields (SOF), which produce faithful surface regions outperforming conventional SDF-based methods. We further presented point upsampling module (PUM), which refines coarse anchor points to capture the details, consistently enhancing generation quality. Extensive experiments validated that `Trip2GS` scales across diverse 3D domains, establishing a strong foundation for future advancements in this area.

References

Florian Barthel, Arian Beckmann, Wieland Morgenstern, Anna Hilsmann, and Peter Eisert. Gaussian splatting decoder for 3d-aware generative adversarial networks. In *CVPRW*, 2024a.

Florian Barthel, Arian Beckmann, Wieland Morgenstern, Anna Hilsmann, and Peter Eisert. Gaussian splatting decoder for 3d-aware generative adversarial networks. In *CVPR*, 2024b.

Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *ICLR*, 2018.

Ziang Cao, Fangzhou Hong, Tong Wu, Liang Pan, and Ziwei Liu. Large-vocabulary 3d diffusion model with transformer. In *ICLR*, 2024.

Ziang Cao, Fangzhou Hong, Tong Wu, Liang Pan, and Ziwei Liu. Difftf++: 3d-aware diffusion transformer for large-vocabulary 3d generation. *IEEE TPAMI*, 2025.

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, 2022.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Kevin Chen, Christopher B Choy, Manolis Savva, Angel X Chang, Thomas Funkhouser, and Silvio Savarese. Text2shape: Generating shapes from natural language by learning joint embeddings. In *ACCV*, 2019.

Zhaoxi Chen, Jiaxiang Tang, Yuhao Dong, Ziang Cao, Fangzhou Hong, Yushi Lan, Tengfei Wang, Haozhe Xie, Tong Wu, Shunsuke Saito, et al. 3dtopia-xl: Scaling high-quality 3d asset generation via primitive diffusion. In *CVPR*, 2025.

Gene Chou, Yuval Bahat, and Felix Heide. Diffusion-sdf: Conditional generative modeling of signed distance functions. In *ICCV*, 2023.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv preprint arXiv:2212.08051*, 2022.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. *arXiv preprint arXiv:2307.05663*, 2023.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021.

Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *ICCV*, 2021a.

Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *IJCV*, 2021b.

Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. In *NeurIPS*, 2022.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Lutao Jiang, Xu Zheng, Yuanhuiyi Lyu, Jiazhou Zhou, and Lin Wang. Brightdreamer: Generic 3d gaussian generative framework for fast text-to-3d synthesis. *arXiv preprint arXiv:2403.11273*, 2024.

Xiaoliang Ju and Hongsheng Li. Directtrigs: Triplane-based gaussian splatting field representation for 3d generation. In *CVPR*, 2025.

Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *NeurIPS*, 2022.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4):1–14, 2023.

Yushi Lan, Fangzhou Hong, Shuai Yang, Shangchen Zhou, Xuyi Meng, Bo Dai, Xingang Pan, and Chen Change Loy. Ln3diff: Scalable latent neural fields diffusion for speedy 3d generation. In *ECCV*, 2024a.

Yushi Lan, Shangchen Zhou, Zhaoyang Lyu, Fangzhou Hong, Shuai Yang, Bo Dai, Xingang Pan, and Chen Change Loy. Gaussiananything: Interactive point cloud latent diffusion for 3d generation. In *ICLR*, 2024b.

Jumin Lee, Sebin Lee, Changho Jo, Woobin Im, Juhyeong Seon, and Sung-Eui Yoon. Semcity: Semantic scene generation with triplane diffusion. In *CVPR*, 2024.

Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, et al. Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models. *CoRR*, 2025.

Chenguo Lin, Panwang Pan, Bangbang Yang, Zeming Li, and Yadong Mu. Diffsplat: Repurposing image diffusion models for scalable gaussian splat generation. *arXiv preprint arXiv:2501.16764*, 2025.

Zhen Liu, Yao Feng, Michael J Black, Derek Nowrouzezahrai, Liam Paull, and Weiyang Liu. Meshdiffusion: Score-based generative 3d mesh modeling. In *ICLR*, 2023.

William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH*, 1987.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *CVPR*, 2024.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *CVPR*, 2022.

Quan Meng, Lei Li, Matthias Nießner, and Angela Dai. Lt3sd: Latent trees for 3d scene diffusion. In *CVPR*, 2025.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019.

Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulo, Peter Kontschieder, and Matthias Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In *CVPR*, 2023.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *ICML*, 2021.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *TMLR*, 2024.

Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022.

Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. In *NeurIPS*, 2021.

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020.

Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. In *NeurIPS*, 2021.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

Lingteng Qiu, Guanying Chen, Xiaodong Gu, Qi Zuo, Mutian Xu, Yushuang Wu, Weihao Yuan, Zilong Dong, Liefeng Bo, and Xiaoguang Han. Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d. In *CVPR*, 2024.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Aashish Rai, Dilin Wang, Mihir Jain, Nikolaos Sarafianos, Arthur Chen, Srinath Sridhar, and Aayush Prakash. Uvgs: Reimagining unstructured 3d gaussian splatting using uv mapping. In *CVPR*, 2025.

Barbara Roessle, Norman Müller, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, Angela Dai, and Matthias Nießner. L3dg: Latent 3d gaussian diffusion. In *SIGGRAPH Asia*, 2024.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

L.I. Rudin and S. Osher. Total variation based image restoration with free local constraints. In *ICIP*, 1994.

Jianghao Shen, Nan Xue, and Tianfu Wu. A pixel is worth more than one 3d gaussians in single-view 3d reconstruction. *CoRR*, 2024.

Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *NeurIPS*, 2021.

Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. *ACM TOG*, 2023.

Jaehyeok Shim and Kyungdon Joo. Ditto: Dual and integrated latent topologies for implicit 3d reconstruction. In *CVPR*, 2024.

Jaehyeok Shim, Changwoo Kang, and Kyungdon Joo. Diffusion-based signed distance fields for 3d shape generation. In *CVPR*, 2023.

J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *CVPR*, 2023.

Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *CVPR*, 2024.

Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *ECCV*, 2024.

Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. In *NeurIPS*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, et al. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *CVPR*, 2023.

Rundi Wu, Ruoshi Liu, Carl Vondrick, and Changxi Zheng. Sin3DM: Learning a diffusion model from a single 3d textured shape. In *ICLR*, 2024a.

Zhennan Wu, Yang Li, Han Yan, Taizhang Shang, Weixuan Sun, Senbo Wang, Ruikai Cui, Weizhe Liu, Hiroyuki Sato, Hongdong Li, et al. Blockfusion: Expandable 3d scene generation using latent tri-plane extrapolation. *ACM TOG*, 2024b.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *CVPR*, 2025.

Han Yan, Yang Li, Zhennan Wu, Shenzhou Chen, Weixuan Sun, Taizhang Shang, Weizhe Liu, Tian Chen, Xiaqiang Dai, Chao Ma, et al. Frankenstein: Generating semantic-compositional 3d scenes in one tri-plane. In *SIGGRAPH Asia*, 2024.

Haitao Yang, Yuan Dong, Hanwen Jiang, Dejia Xu, Georgios Pavlakos, and Qixing Huang. Atlas gaussians diffusion for 3d generation. In *ICLR*, 2025.

Lior Yariv, Omri Puny, Oran Gafni, and Yaron Lipman. Mosaic-sdf for 3d generative models. In *CVPR*, 2024.

Bowen Zhang, Yiji Cheng, Jiaolong Yang, Chunyu Wang, Feng Zhao, Yansong Tang, Dong Chen, and Baining Guo. Gaussiancube: A structured and explicit radiance representation for 3d generative modeling. In *NeurIPS*, 2024.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *CoRR*, 2025.

Junsheng Zhou, Weiqi Zhang, and Yu-Shen Liu. Diffgs: Functional gaussian splatting diffusion. In *NeurIPS*, 2025.

Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *ICCV*, 2021.

Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. In *CVPR*, 2024.

Qi Zuo, Xiaodong Gu, Yuan Dong, Zhengyi Zhao, Weihao Yuan, Lingteng Qiu, Liefeng Bo, and Zilong Dong. High-fidelity 3d textured shapes generation by sparse encoding and adversarial decoding. In *ECCV*, 2024.

Appendix

Overview

In appendix, we provide further details on the following topics:

- Section A offers an in-depth discussion of the experimental setting, dataset preparation process, model architecture, and training strategies.
- Section B provides an extended comparison with TRELLIS, highlighting architectural differences, reconstruction results, and generation challenges under the unconditional setting.
- Section C presents additional details and results for the various conditional generation tasks, including implementation details for conditional generation cases.
- Section D reports a comprehensive efficiency analysis covering training time, memory usage, and inference latency across all baselines.
- Section E includes further ablation studies focusing on the scene completion application and upsampling effects of the proposed PUM.
- Section F provides additional quantitative and qualitative results on the ShapeNet, 3D-FRONT, and the recent G-buffer Objaverse dataset.
- Section G discusses the key component of our approach and outlines potential future research directions.

A Implementation Details

We trained all networks on 8 NVIDIA A100 GPUs using the PyTorch framework. Experiments were conducted on a system equipped with an AMD EPYC 7742 64-Core Processor, 1.0 TiB of RAM, and running Ubuntu 20.04 LTS. To ensure reproducibility and fairness, we fixed the random seeds to 0 for both NumPy and PyTorch to eliminate performance variations caused by stochastic factors.

A.1 Dataset Preparation

As we mentioned in the main paper, we utilize colored point cloud as an input for our Trip2GS VAE. To construct an input colored point cloud, we follow the data construction step of Shap-E (Jun & Nichol, 2023). To construct the input colored point cloud, we first render 20 RGB and depth images of a given mesh from randomly sampled viewpoints on a sphere. Using these rendered images, we unproject each valid pixel into 3D space based on its depth value. Specifically, for each pixel (u, v) in the image, we compute its corresponding 3D ray direction and scale the depth value accordingly. The 3D coordinate $\mathbf{p} \in \mathbb{R}^3$ is obtained as $\mathbf{p} = \mathbf{o} + s \cdot \mathbf{d}$, where $\mathbf{o} \in \mathbb{R}^3$ denotes the camera origin, $\mathbf{d} \in \mathbb{R}^3$ is the unit ray direction, and $s \in \mathbb{R}$ represents the depth scale factor. To ensure that only meaningful surface points are retained, we filter out pixels with infinite depth (*e.g.*, background pixels) and those with low transparency ($\alpha < 1$). Finally, after unprojecting all valid pixels into 3D space, we randomly sample 100k points from the extracted colored point cloud to construct the set of input points.

A.2 Details of Trip-to-Gaussian VAE

Hyperparameters. We train the Trip2GS decoder for 300k iterations with batch size of 8 using the AdamW optimizer with a cosine annealing learning rate scheduler, setting the initial learning rate to $1e{-}4$. Training with full iterations takes approximately 2.5 days on 8 A100 GPUs. The resolution of GIM is fixed

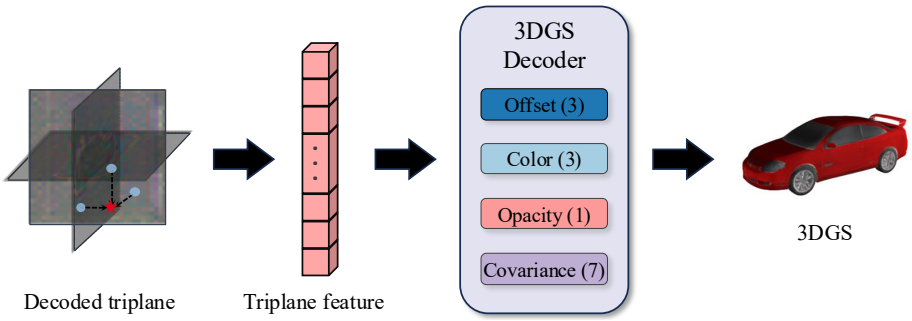


Figure 8: **Network architecture of 3DGS decoder.** The 3DGS decoder consists of four linear layers, each responsible for decoding a specific Gaussian attribute.

at 64, ensuring computational efficiency and reliability of geometric prior for 3DGS generation. Additionally, to effectively generate a sufficient number of 3D Gaussians for detailed representation, we apply different point upsampling ratios based on the complexity of the geometry and texture of an object/scene. The upsampling ratio is set differently according to the scale and texture complexity of the object: 4 for chairs, 8 for bedrooms, and 16 for airplane and car categories. These configurations allow our model to adaptively increase the density of 3D Gaussians, ensuring high-quality reconstruction across various object and scene categories.

Gaussian Indicator Module (GIM). After the triplane decoder, the triplane channel is increased from 4 to 256 using a U-Net style network with triplane convolutions, making the shape to (32, 32, 256). GIM first upsamples triplane features to (64, 64, 128) to match the resolution of R_{voxel}. Uniform voxel points query triplane features, which are then passed to a 5-layer MLP with sigmoid activation to predict SOF values; values above 0.5 yield M_{coarse} coarse anchor points.

Point Upsampling Module (PUM). PUM is a 3-layer MLP that maps (M_{coarse}, 3) to (M_{coarse}, $3k$) and reshape to (M, 3) refined dense anchor points, where $M = M_{coarse} \times k$.

3DGS Decoder. Our 3DGS decoder consists of four sub-decoders, each responsible for predicting different attributes of 3D Gaussians: offset, color, covariance, and opacity (see Fig. 8).

- **Offset o**. We apply a sigmoid function to predict the offset, scale it to the range $[-1/32, 1/32]$, and add it to the anchor point position to determine the final 3D Gaussian center: $\mathbf{p}_{\text{final}} = \mathbf{p} + (\sigma(\mathbf{o}) - 0.5)/64$, where $\mathbf{p} \in \mathbf{P}$ is the anchor point position, $\mathbf{o}$ is the predicted offset, and $\sigma(\cdot)$ denotes sigmoid function.
- **Color (RGB) c**. The feature values are directly used as RGB values without additional transformation: $\mathbf{c} = \mathbf{f}_{\text{RGB}}$, where $\mathbf{f}_{\text{RGB}}$ is the extracted triplane feature.
- **Opacity** α. A sigmoid function is applied to ensure the opacity values are within the range $[0, 1]$: $\alpha = \sigma(o_\alpha)$, where o_α is the predicted opacity.
- **Scale** s. A sigmoid function is applied to predict the scale, followed by a multiplication by 0.1 to stabilize 3D Gaussian optimization: $s = 0.1 \cdot \sigma(o_s)$, where o_s is the predicted scale.
- **Rotation q**. The predicted rotation values are normalized to form a quaternion representation, ensuring a valid rotation: $\mathbf{q} = \frac{\mathbf{o}_q}{\|\mathbf{o}_q\|}$, where o_q is the predicted rotation vector, and $\|\cdot\|$ represents the Euclidean norm.

Loss Computation. Mask loss is an L2 loss between the predicted and GT binary mask, 0 for the background and 1 for the object pixels. Chamfer loss is computed between the input point cloud and the center of 3D Gaussians.

Training Strategy. For effective training of the Trip2GS VAE, we introduce two learning strategies. First, to ensure GIM provides a robust geometry prior, we employ a gradient control strategy. Specifically, we detach the coarse anchor points $\mathbf{P}_{\text{coarse}}$ obtained from GIM from gradient computation from other losses except for $\mathcal{L}_{\text{GIM}}$. Without this detachment, gradients from the other losses propagate back to GIM, where we empirically found to interfere with the learning of geometric priors. Since GIM is intended to provide robust geometric priors for object representation, we enforce that it is trained solely with the occupancy loss, without interference from other gradients.

Second, to stabilize early-stage training, we train the 3DGS decoder and GIM separately in the initial iterations. Early in training, GIM does not yet provide accurate geometric information, leading to unstable supervision signals for 3DGS generation. These incorrect surface priors fail to provide meaningful guidance to the 3DGS decoder, which negatively impacts the learning process. To mitigate this, in the early training phase, we sample query points for the 3DGS decoder from the ground-truth point cloud rather than using anchor points extracted from GIM.

These two strategies ensure the stable training process of Trip2GS VAE, allowing each module to focus on its designated task, geometry learning for GIM and texture modeling for 3DGS decoder, which ultimately leads to higher-quality outputs.

A.3 Details of Triplane Diffusion

For the denoising network, we employ a U-Net (Ronneberger et al., 2015) based architecture following the design of ADM (Nichol & Dhariwal, 2021; Dhariwal & Nichol, 2021). It is composed of four hierarchical levels, with each level containing two residual blocks, each followed by a channel-wise self-attention block. For self-attention and cross-attention in conditional generation, the number of attention heads per channel is set to 64. At each level, the spatial resolution is halved while the number of channels is doubled. We use the batch size of 64 and AdamW optimizer with a cosine annealing learning rate scheduler, setting the learning rate to $2e{-}5$. The training iterations for triplane diffusion are set as follows: 500k iterations for the airplane category, 600k for car, and 1M for both chair and bedroom. The training takes about 1.5 days for the airplane and car categories and 3 days for the chair and bedroom categories.

B Comparison to TRELLIS

In the main paper, we show that `Trip2GS` outperforms prior methods in unconditional 3D generation, achieving superior quality in both geometry and texture. A key contributor to this improvement is our coarse-to-fine generation strategy built on the Gaussian indicator module (GIM) and point upsampling module (PUM), which effectively bridge structured triplane and unstructured 3DGS representations.

The most relevant approach to ours is TRELLIS (Xiang et al., 2025), a large-scale conditional generation framework. TRELLIS introduces the structured latent (SLat) representation, which encodes sparse voxel grids with DINOv2 (Oquab et al., 2024) image features of a 3D asset. Conceptually, TRELLIS can be regarded as a heavyweight multi-stage 3D counterpart to ours – it explicitly constructs a voxel grid representing surface occupied regions, whereas `Trip2GS` implicitly learns such regions within an end-to-end generative pipeline. To generate 3D Gaussians from its sparse structure, TRELLIS extracts DINOv2 features from input images and embeds them into the voxel space, followed by training an additional VAE model.

Since TRELLIS is built for large-scale conditional generation, we adapt it for unconditional generation to enable fair comparison. Because the XL (2B-parameter) model used in the paper is computationally prohibitive, we reproduce experiments using the officially released B (base) model. All models are trained under comparable computational budgets and training schedules to ensure a fair efficiency comparison (see Tab. 9).

Table 7: **Comparison of `Trip2GS` with TRELLIS on VAE reconstruction quality measured by PSNR.** TRELLIS* denotes our reproduced version of the original TRELLIS (Xiang et al., 2025), used to ensure a fair comparison in the unconditional generation setting (see Sec. D).

Method	ShapeNet			Objaverse			3D-FRONT
	Airplane	Car	Chair	Animals	Food	Plants	Bedroom
TRELLIS*	**35.13**	**31.68**	29.57	**30.50**	**29.23**	28.42	**26.88**
`Trip2GS` (ours)	32.84	30.13	**30.24**	29.88	27.80	**29.20**	26.00

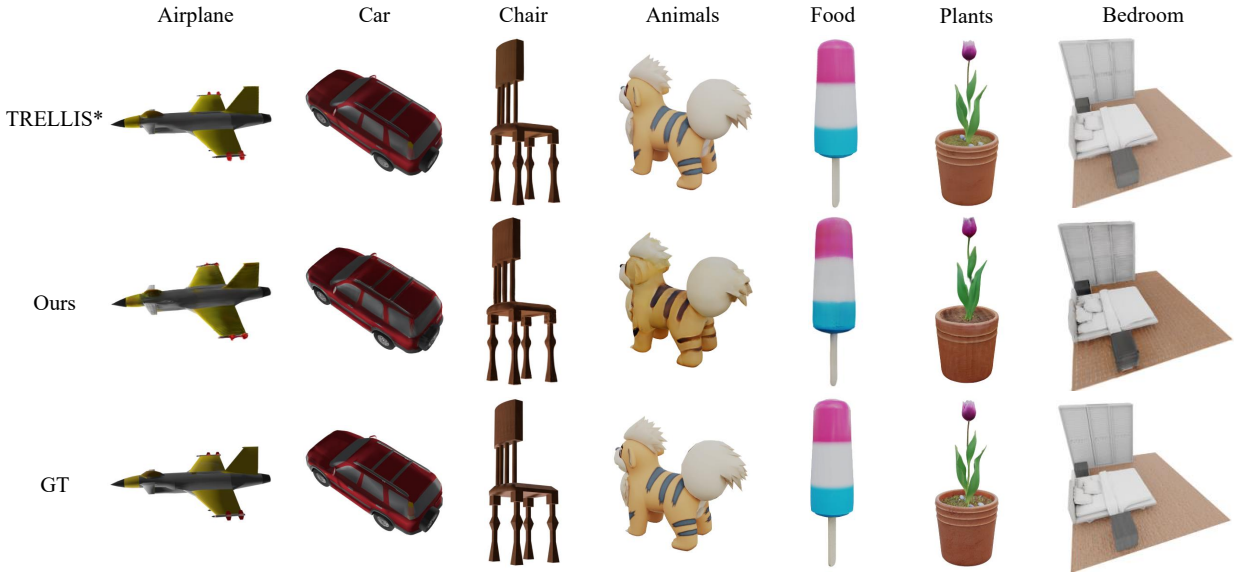


Figure 9: **Qualitative comparison of `Trip2GS` with TRELLIS on VAE reconstruction.** TRELLIS* denotes our reproduced version of the original TRELLIS (Xiang et al., 2025), used to ensure a fair comparison in the unconditional generation setting (see Sec. D).

B.1 Reconstruction

We first compare `Trip2GS` with TRELLIS on the task of VAE reconstruction, where the goal is to generate 3D Gaussians from a constructed input representation of a 3D asset. Both models are trained under comparable settings: TRELLIS with 304.16M parameters and our model with 227.47M parameters, using batch size 8 for 300k iterations for both. Results are provided in Fig. 9 and Tab. 7.

TRELLIS adopts two-stage design that explicitly disentangles coarse geometry and fine structures by constructing two types of data – sparse structure and SLat – and trains two separate VAEs for each. In addition, TRELLIS relies on a larger latent representation ($16 \times 16 \times 16 \times 8$), which is approximately $2.67\times$ larger than ours ($32 \times 32 \times 4 \times 3$), providing additional capacity that naturally benefits reconstruction quality. Combined with extra preprocessing and supervision, these factors give TRELLIS an inherent advantage in reconstruction, making it challenging for a fully end-to-end model like ours to surpass it. Despite this, `Trip2GS` achieves comparable reconstruction quality, even outperforming TRELLIS in the chair and plants categories. As shown in the Fig. 9, the visual differences between the two methods are minimal, further confirming that our end-to-end approach preserves reconstruction fidelity.

Furthermore, TRELLIS generates significantly more Gaussians (e.g., TRELLIS 250k vs. ours 30k for the chair category), yet our model achieves similar or better fidelity, demonstrating that `Trip2GS` produces highly efficient and structurally meaningful Gaussians rather than relying on the quantity. Although we

apply total variation (TV) regularization, which slightly reduces reconstruction PSNR while substantially improving generation quality as discussed in the main paper, our model still remains competitive.

Despite using substantially fewer Gaussians and no additional computation, `Trip2GS` achieves comparable reconstruction quality while providing a simpler and fully differentiable architecture, underscoring the efficiency and representational strength of our end-to-end coarse-to-fine design.

B.2 Generation

For generation, we followed the official TRELLIS configuration as closely as possible, training the model with 285.36M parameters (508.50M for ours), batch size 16 (64 for ours), and 500k iterations (also 500k for ours), which yields a comparable training budget to ours while keeping the remaining hyperparameters unchanged. However, under these conditions, the flow matching model fails to converge: it consistently produces unnatural samples with noticeable artifacts (see Fig. 10). These results indicate that TRELLIS struggles to converge under limited data and computation resources and requires additional training or hyperparameter tuning to achieve stable generation quality. In contrast, `Trip2GS` remains stable and effective under the same training conditions, highlighting the robustness of our network design for unconditional generation.

Figure 10: **Generation results produced by TRELLIS.** The model consistently fails to generate realistic and structurally coherent samples under unconditional generation setting.

Since the goal of unconditional 3D generation is to achieve strong generative performance under constrained data amount and computational resources, the results demonstrate that the end-to-end design of `Trip2GS` offers a more practical and lightweight solution than the two-stage TRELLIS framework.

C Conditional Generation

Building on Trip2GS, we showcase a diverse set of conditional generation scenarios. We consider four types of conditions: geometry (point cloud and voxel grid), layout, text, and image. Our overall architecture for conditional generation is shown in Fig. 11, where a conditional encoder and condition injection layers are integrated into the triplane diffusion. For conditional generation, we train the diffusion model using the following formulation:

$$\mathcal{L}_{\text{diff}} = \left\| \mu_\theta(\hat{\mathbf{Z}}_t, t, \mathcal{C}) - \hat{\mathbf{Z}}_0 \right\|_2, \tag{7}$$

where $\mathcal{C}$ is an input condition.

C.1 Condition Preparation

For the point cloud condition, we sample the XYZ coordinates from the input colored point cloud without the RGB values. We sample 10k points for objects and 20k points for scenes, considering each scale.

For the voxel condition, we convert a given point cloud into a binary voxel grid. First, we initialize an empty voxel grid with all values set to zero. Next, each point is mapped to a voxel index within the grid. This is achieved by normalizing the point coordinates to the voxel grid resolution. The points are scaled into the range of the voxel grid and then rounded down to the nearest integer indices. To ensure that only valid points are considered, we filter out indices that fall outside the boundary of the voxel grid. Specifically, we retain only those indices that are within the valid range $[0, R-1]$ along all three dimensions, which we use $R = 64$ for all experiments. After filtering, the valid 3D voxel indices are converted into a single-dimensional index representation. Finally, the corresponding locations in the voxel grid are marked as occupied by setting their values to 1. This step finalizes the discretization process, ensuring that all input points are represented within the voxel grid structure.

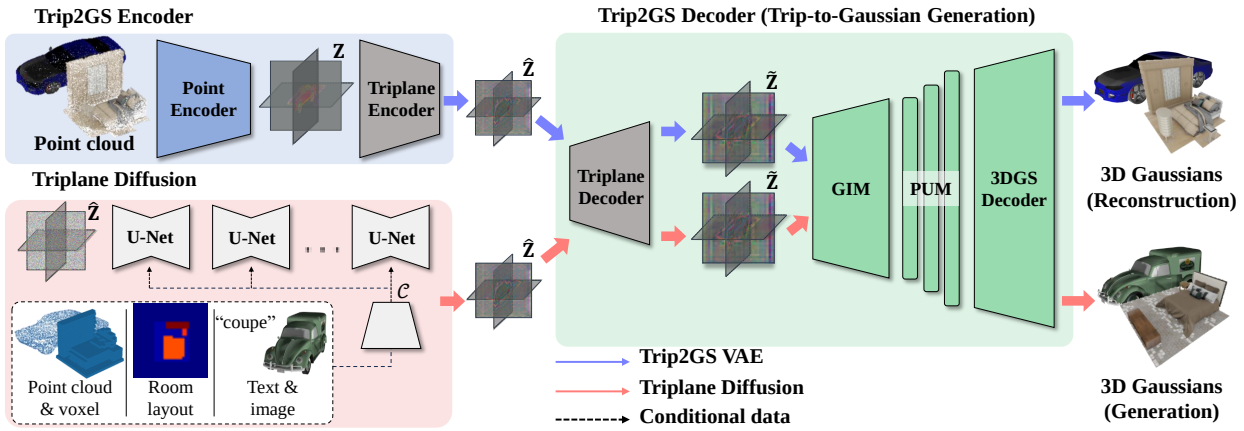


Figure 11: **The overall architecture of `Trip2GS` for conditional generation.**

The layout condition is obtained by rendering the scene from a top-down orthographic view, where each object category is mapped to a unique label. The resulting layout maps are stored as $32 \times 32 \times 1$ tensors, where the channel dimension encodes the category ID of each pixel.

Text conditions are obtained from the Text2Shape dataset (Chen et al., 2019) for the chair category. For feature embedding, we use a pretrained CLIP text encoder from `openai/clip-vit-large-patch14-336`, which outputs (77, 768) sized feature vectors.

For image conditions, we use images of each object captured from the same viewpoint. Likewise, in the text condition, we use a pretrained CLIP image encoder from `openai/clip-vit-large-patch14-336`, which outputs (577, 1024) sized feature vectors.

C.2 Network Architecture

For geometry-conditioned generation which aims to generate textured objects or scenes that adhere to a given coarse geometric structure, such as point clouds or voxel grids, we jointly train the condition encoder with triplane diffusion. We first embed point cloud or voxel grid condition into a triplane representation using condition-specific encoder so that the features from the conditions have the same shape as the latent triplane. The point cloud condition encoder adopts the PointNet (Qi et al., 2017)-based encoder like ConvONet (Peng et al., 2020) similar to the point encoder used in Trip2GS VAE, consisting of three 3D-aware ResNet blocks. The voxel condition encoder processes an occupancy voxel grid of shape $V_{\text{cond}} \in \mathbb{R}^{32\times32\times32}$ using two 3D convolution blocks, followed by axis-wise mean pooling along each of the three axes to construct the triplane representation. Once the condition is embedded in the triplane space, we inject it into the diffusion model using adaptive group normalization (AdaGN) (Dhariwal & Nichol, 2021; Zhang et al., 2024). The condition encoder is jointly trained with the triplane diffusion, allowing for end-to-end learning of geometry-conditional 3D generation.

For layout-conditioned generations, we use a semantic segmentation map of top-down orthogonal view. Unlike geometry-conditioned generation, the layout condition is directly used without an additional condition encoder. Since the layout information is aligned with the xy-plane of the triplane latent, we apply AdaGN to inject the condition into the xy-plane of the triplane feature and vanilla group normalization to other planes.

For text and image-conditioned generations, we leverage a pretrained CLIP (Radford et al., 2021) encoder to extract semantic features from textual and visual inputs. During diffusion training, we integrate these features into the triplanes using cross-attention blocks, allowing the model to learn high-level semantic concepts to spatially structured triplane features. To ensure effective conditioning, all conditioning mechanisms are applied at every block of the encoder of the denoiser U-Net, allowing the model to fully capture and propagate conditional information throughout the diffusion process.

C.3 Generation Results.

To demonstrate the applicability of Trip2GS, we present example of generated samples with diverse conditions. Further conditional generation results can be found in Sec. F

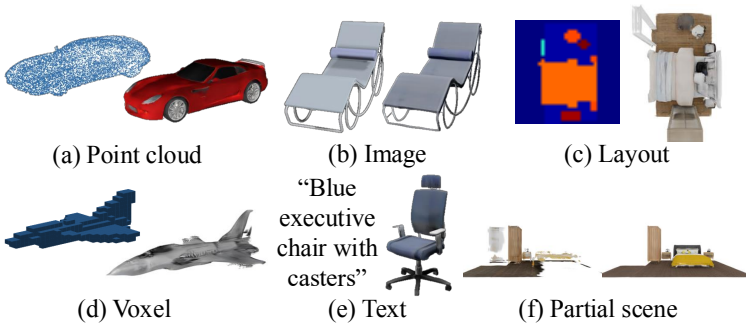


Figure 12: **Examples of various conditional generations.**

Geometry Conditioning. We consider two types of conditional inputs: point cloud and voxel grid. As shown in Fig. 12-(a) and (d), our method generates plausible 3D objects aligned well with given geometry conditions.

Layout Conditioning. For layout condition, we generate scenes aligned to a given scene layout. As shown in Fig. 12-(c), `Trip2GS` effectively generates a scene that closely follows the given layout.

Text/Image Conditioning. For text/image-to-3D generation, we show that `Trip2GS` generates high-fidelity objects that follow the given text or image (see Fig. 12-(b) and (e)). This demonstrates that our method can learn semantics as well as spatial cues.

Scene Completion. We present scene completion as an additional application to validate the versatility of `Trip2GS`. For scene completion, we adopt RePaint (Lugmayr et al., 2022), a diffusion-based inpainting method that leverages a stochastic denoising process to refine missing regions iteratively. Given a partial scene, we utilize our triplane diffusion with RePaint to generate the missing half, ensuring structural and contextual coherence. As shown in Fig. 12-(f), `Trip2GS` effectively completes the missing scene while maintaining consistency with the given partial input. Further details on hyperparameters and additional results for scene completion are provided in Sec. E.3.

C.4 Comparison on Text-to-3D

In this section, we compare our approach with recent text-to-3D generation methods (Peng et al., 2021; Lan et al., 2024a). We present the quantitative comparison results in Tab. 8, where our method outperforms existing approaches, even in the text-conditioned 3D generation task. As shown in Fig. 14 and Fig. 15, `Trip2GS` captures more detailed geometry while maintaining better alignment with the given text prompts. These results demonstrate the effectiveness of our conditional mechanism in following given semantic cues, in addition to the advantage of our framework in capturing fine-grained details.

Table 8: **Quantitative comparison of text-to-3D generation on ShapeNet chair and car categories using CLIP-Score.**

Category	Shap-E	LN3Diff	Trip2GS (Ours)
Chair	30.31	28.87	**30.84**
Car	26.18	26.70	**27.05**

C.5 Comparison on Image-to-3D

We also show qualitative results on image-to-3D generation compared with existing works (Peng et al., 2021; Szymanowicz et al., 2024) in Fig. 16. The results confirm that our method generates 3D outputs that are well-aligned with the given images. It should be noted that compared to recent works curated for image-to-3D generation, `Trip2GS` shows comparable or even better results.

Through further comparisons on conditional generation tasks, we validate that our method is a scalable approach for both text and image conditioned 3D generation as well as unconditional 3D generation.

Table 9: **Efficiency comparison between our model and baselines.** All experiments are conducted using 8 A6000 48G GPUs. TRELLIS* denotes our reproduced version of the original TRELLIS (Xiang et al., 2025), used to ensure a fair comparison in the unconditional generation setting (see Sec. B).

Method	Train (d)	Train Mem (G)	Inference (s)
GET3D (Gao et al., 2022)	4.3	5.91	0.35
DiffGS (Zhou et al., 2025)	6.7	22.34	33.24
GaussianCube (Zhang et al., 2024)	10.3	29.15	12.13
AtlasGaussians (Yang et al., 2025)	1.6	23.38	9.22
TRELLIS* (Xiang et al., 2025)	7.9	18.19	13.72
Ours	7.4	17.98	9.48

D Efficiency Comparison

In this section, we provide a comprehensive comparison of our method against baseline approaches in terms of training time, training memory, and inference time. The quantitative results are summarized in Tab. 9. Regarding training time, AtlasGaussians converges the fastest with 1.6 days of training, while GaussianCube requires the longest time (10.3 days) due to its heavy preprocessing and voxel-based 3D architecture. Our model takes 7.4 days, placing it in the mid-range among baselines while achieving superior generative quality.

For training memory, GET3D shows the lowest memory usage because it employs GAN-based generative framework, which requires less memory than diffusion-based methods. Among the models that represent objects using 3D Gaussian primitives, our model demonstrates the most efficient memory consumption. This indicates that our design, which leverages a triplane latent representation together with the GIM and PUM modules, enables efficient generation of 3D Gaussians without incurring substantial memory overhead.

In terms of inference time, GAN-based GET3D achieves the fastest generation latency. However, they tend to generate less realistic samples compared to diffusion-based methods. DiffGS exhibits the longest generation time, due to their time-consuming octree-based optimization for Gaussian extraction. Among the diffusion-based generative models, our model is one of the fastest, highlighting the benefit of our compact latent representation and efficient Gaussian generation pipeline.

E Additional Ablation Studies

Table 10: **Reconstruction quality on the test set of ShapeNet airplane category with different upsampling ratios using neural Gaussians.** Here, $\boldsymbol{k}$ means upsampling ratio.

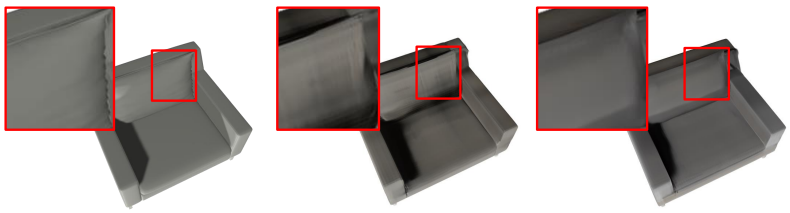
$\boldsymbol{k}$	PSNR	SSIM	LPIPS
1	27.13	**0.9586**	0.0471
4	**27.66**	**0.9586**	**0.0451**
16	27.24	0.9573	0.0471

E.1 Comparison with Neural Gaussians

One possible approach to increase the number of Gaussians from a limited set of anchor points is generating neural Gaussians (Lu et al., 2024; Shen et al., 2024). These methods decode point features $\mathbf{f} \in \mathbb{R}^{N \times C}$ into 3D Gaussians $\mathbf{g} \in \mathbb{R}^{kN \times 14}$, where each point generates k Gaussians, efficiently augmenting the Gaussians

Table 11: **Reconstruction quantitative results on test set of ShapeNet airplane category with different upsampling ratios.** Here, ***k*** denotes the upsampling ratio, **M** the training memory, and **T** the inference speed.

k	**PSNR**	**SSIM**	**LPIPS**	**M (GB)**	**T (s)**
1	26.72	0.9558	0.0513	15.35	0.099
4	27.56	0.9641	0.0435	18.17	0.101
16	**27.94**	**0.9702**	**0.0396**	24.01	0.107



(a) Ground truth (b) w/o mapping (c) w/ mapping

Figure 13: **Visualization of the generated results with respect to the mapping function of our PUM.** Mapping function alleviates tile artifacts caused by the discrete nature of coarse anchor points from GIM.

with minimal computational overhead. However, we observed that it introduces undesired grid pattern in the output rendering (refer to Fig. 13-(b)). The issue arises because the anchor points extracted from GIM are inherently discrete and discontinuous. During querying the triplane feature, the anchor points are fixed at predetermined locations, thus causing discrepancies between adjacent features. Furthermore, generating multiple Gaussians from a limited feature set imposes constraints to decode high-quality Gaussians, limiting the effectiveness of upsampling. As shown in Tab. 10, increasing the upsampling ratio k of neural Gaussians results in either similar or worse reconstruction quality.

In contrast, our PUM consistently improves all metrics, validating the effectiveness of our approach for upsampling 3D Gaussians (see Tab. 11). With our lightweight MLP-based PUM network, we can efficiently generate sufficient number of Gaussians without excessive computational burden.

E.2 Effect of Point Mapping

In this work, we proposed PUM, which upsamples the coarse anchor point from GIM while mapping discrete points into continuous space. Here, we investigate the effect of the mapping function of PUM. Specifically, we compare the visual quality of our `Trip2GS` with and without PUM, while setting the upsampling ratio 1, which means PUM does not perform upsampling. By using coarse anchor points from GIM directly as query points for 3DGS decoder, we observed unusual grid pattern or texture abruption in the rendered results, as shown in Fig. 13-(b). In contrast, by leveraging the mapping function of PUM, grid pattern is mitigated, producing visually smooth and coherent outputs (refer to Fig. 13-(c)). These findings indicate that the mapping function enhances the visual quality of the generated 3D Gaussians. With the mapping function of our PUM, reconstruction quality is consistently improved as the upsampling ratio increases (see Tab. 11).

E.3 Hyperparameters for Scene Completion

Scene completion application is achieved by using our trained triplane diffusion model, where half-masked triplane is provided as input and used as a condition. Our completion is based on RePaint (Lugmayr

Table 12: **Comparison of FID, KID, and C-FID scores for different hyperparameters for repaint of scene completion.**

	$j = 10$			$j = 20$			$j = 30$			$j = 40$			$j = 50$		
	FID	**KID**	**C-FID**	**FID**	**KID**	**C-FID**	**FID**	**KID**	**C-FID**	**FID**	**KID**	**C-FID**	**FID**	**KID**	**C-FID**
$r = 5$	17.79	5.31	1.78	17.24	5.04	1.73	17.13	5.02	1.73	17.08	4.96	1.73	17.18	5.12	1.77
$r = 10$	17.63	5.13	1.78	17.14	4.92	1.72	17.16	4.99	**1.71**	**16.98**	**4.91**	1.73	17.01	4.98	1.75
$r = 15$	17.59	5.07	1.74	17.21	4.98	**1.71**	17.05	5.00	1.72	17.05	5.01	1.73	16.99	4.97	1.72

et al., 2022). RePaint proposes an iterative resampling strategy that repeats forward and reverse processes and addresses the disharmony problem that arises due to a mismatch between sampled masked region and denoised result. Selecting an appropriate *jump* (j), which denotes the length of diffusion step for back-and-forth iterations, and *resampling* (r), which indicates the number of iterative resample steps, is crucial for scene completion quality. Experiment results are shown in Tab. 12.

F Additional Results

F.1 Results on ShapeNet/3D-FRONT Datasets

In this section, we present additional generation results, including unconditional generation in Fig. 18, as well as conditional generation with different inputs: geometry (point cloud and voxel) in Fig. 19, room-scale layout in Fig. 20, text in Fig. 21, and image in Fig. 22 on the ShapeNet and 3D-FRONT datasets. In Fig. 23, we present scene completion results using partial scenes.

F.2 Results on Objaverse Dataset

To further validate the generalizability of our method to the high-quality 3D assets, we showcase additional results on the recent G-buffer Objaverse (Zuo et al., 2024; Qiu et al., 2024; Deitke et al., 2022) dataset. G-buffer Objaverse dataset contains a subset of Objaverse (Deitke et al., 2022) dataset, including 10 general categories (*e.g.*, Animals, Furnitures, Food, etc.). Among them, we show unconditional generation results on three categories: Plants, Food and Animals, which contain samples with diverse geometries and textures.

Fig. 17 presents a qualitative comparison of unconditional generation results on the G-buffer Objaverse dataset. Overall, our method produces the most realistic and coherent samples across both geometry and texture compared to baseline models. From a geometry perspective, GET3D generates incomplete structures, and DiffGS sometimes produces empty holes or places Gaussians outside the object boundaries. GaussianCube produces torn surfaces which fails to reconstruct complete surfaces, while AtlasGaussians captures geometry relatively well but shows minor misalignment at fine details. In contrast, our method explicitly captures object surfaces and predicts Gaussian positions accordingly, resulting in accurate, well-aligned geometry. Regarding texture and appearance, baseline models often generate oversmoothed textures that lack high-frequency details. However, our approach synthesizes realistic, diverse, and detailed textures, capturing both global structure and fine details. Further qualitative results are presented in Fig. 24, which demonstrate strong performance on a dataset with diverse geometries and textures.

G Discussion and Future Works

Discussion on GIM. In this work, GIM serves as a key component for the direct generation of 3D Gaussians from triplane features. Specifically, the primary function of GIM is to generate coarse anchor points for 3D Gaussians by extracting surface information, which acts as a reliable geometric prior. We use a resolution of 64 for GIM, which makes our Trip2GS VAE affordable on widely accessible GPUs with 24GB of VRAM, such as the RTX 3090 or 4090. By utilizing GPUs with higher VRAM capacities, we can capture finer details by increasing the resolution of GIM. Additionally, while GIM currently generates coarse anchor points based solely on the geometry of an object, a promising direction for future research involves

enhancing GIM to also incorporate texture information, which could provide more informative anchor points for high-quality 3D generation.

Large-scale Scene Generation. We have demonstrated that our method successfully scales from object-level to scene-level generation. However, generating scenes containing numerous objects or producing large-scale environments remains a challenging problem, and addressing such scenarios typically requires compositional generation strategies or auto-regressive generation. Prior works (Wu et al., 2024b; Meng et al., 2025; Lee et al., 2024) have highlighted the potential of image inpainting techniques as a promising solution for scalable scene construction. Given that our model exhibits strong modeling capacity across both object and scene-level generation, and achieves high-quality scene completion (see Fig. 23) using the Repaint (Lugmayr et al., 2022) algorithm, we believe it provides a solid foundation for tackling these challenges. These capabilities indicate that our framework can be naturally extended to large-scale environments. Advancing in this direction would enable more flexible and scalable scene synthesis pipelines, broaden potential real-world applications, and further advance research in 3D scene generation.

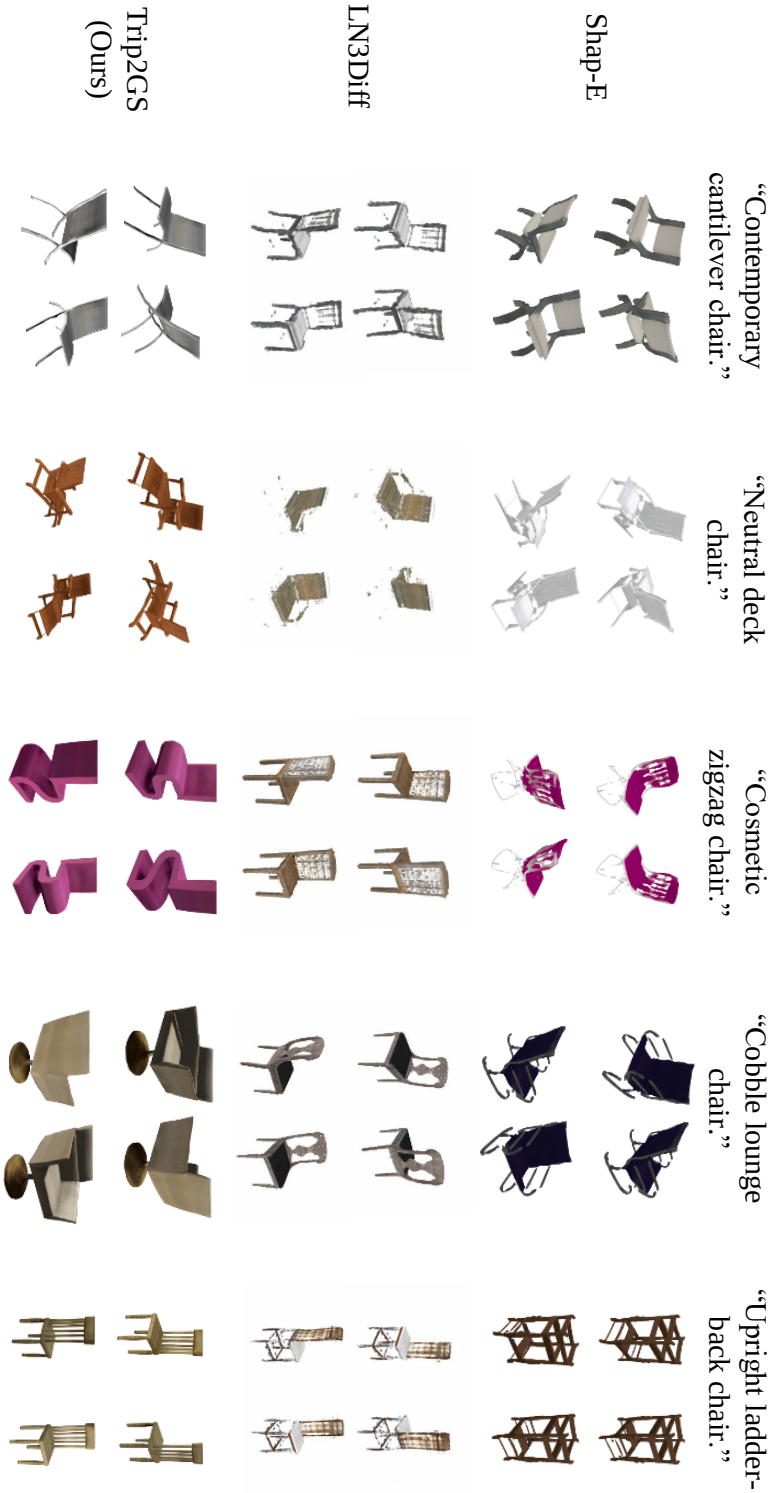


Figure 14: **Qualitative comparisons on text-to-3D generation on the ShapeNet chair category.**

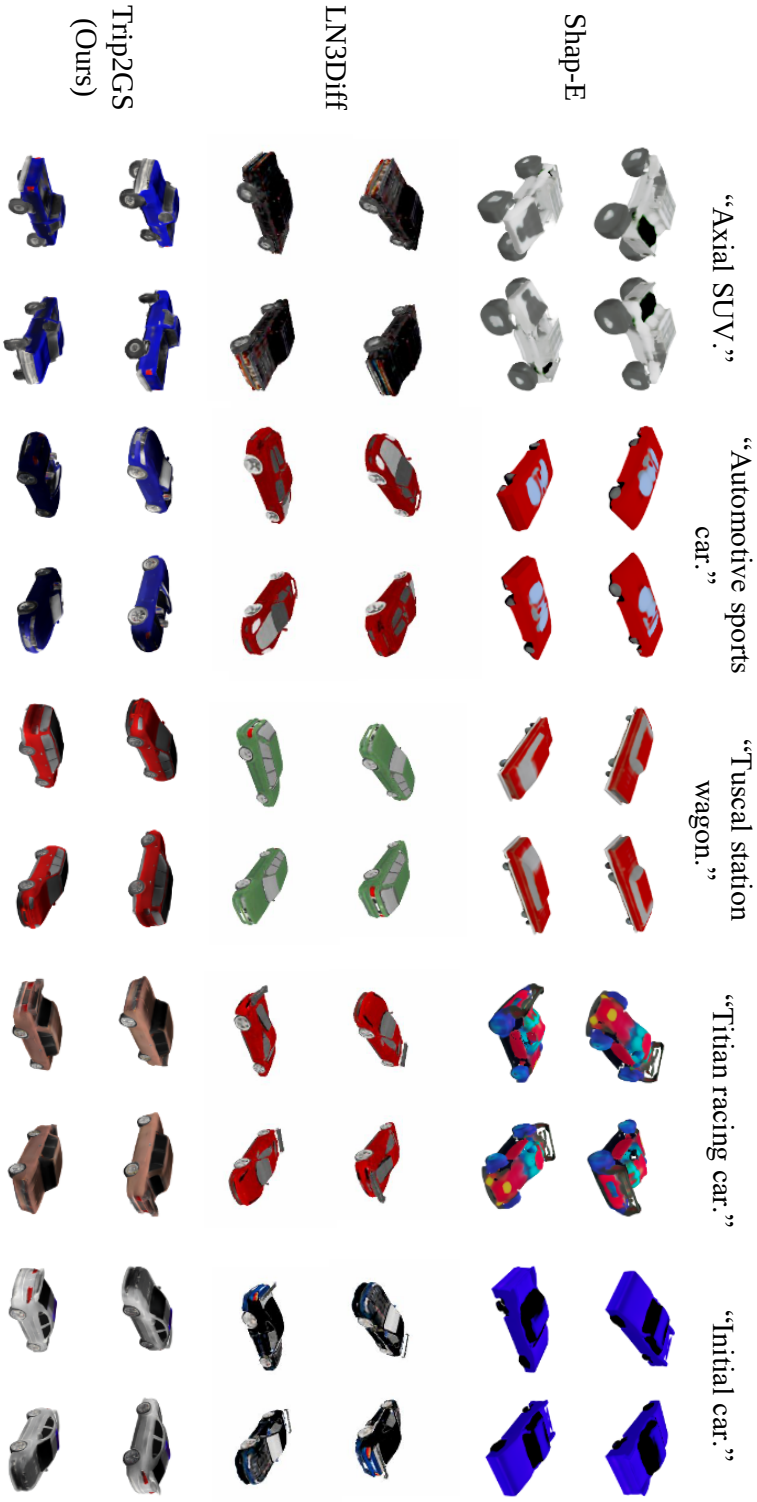


Figure 15: **Qualitative comparisons on text-to-3D generation on the ShapeNet car category.**

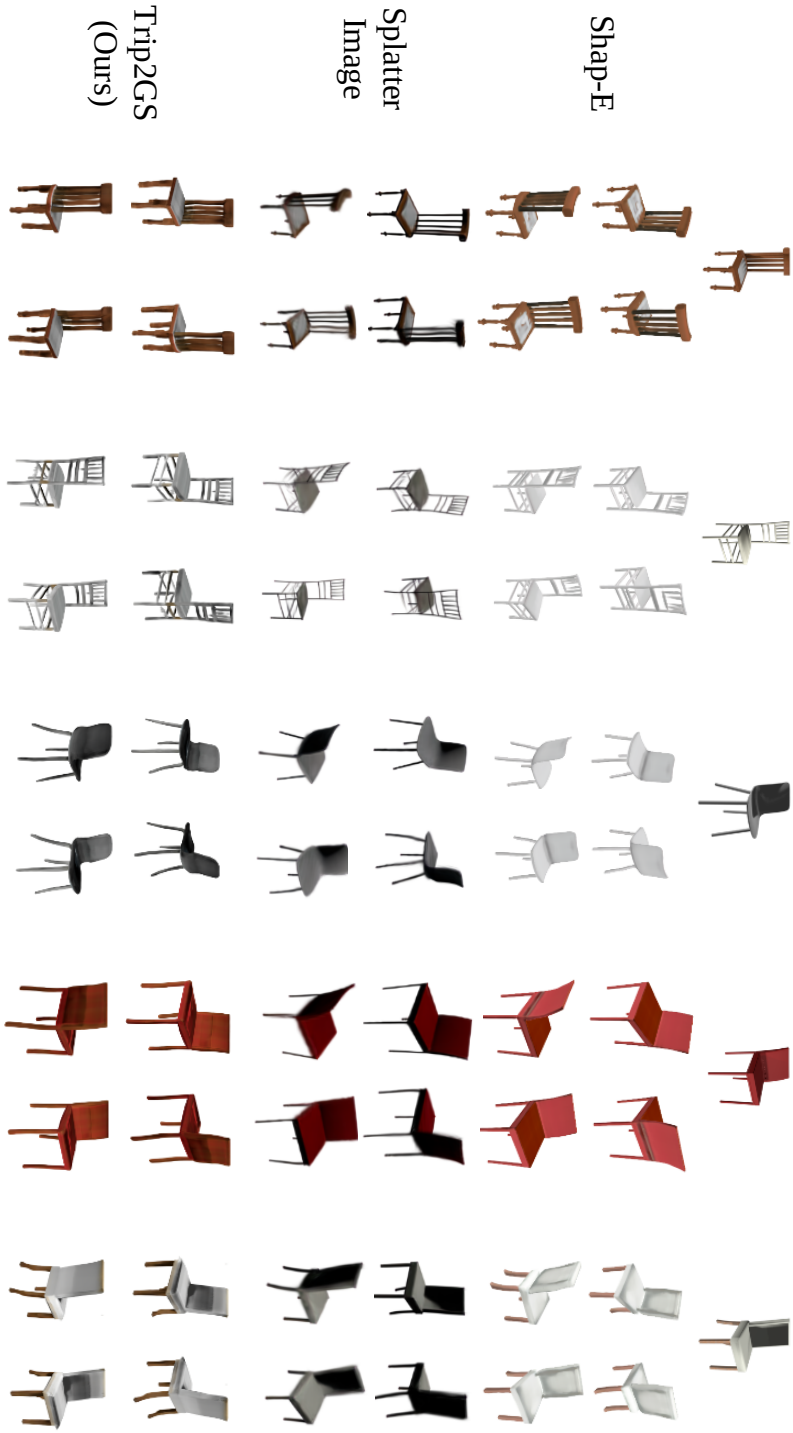


Figure 16: **Qualitative comparisons on image-to-3D generation on the ShapeNet chair category.**

Figure 17: **Detailed comparison of unconditional generation on G-buffer Objaverse plants, food, and animals categories.**

Figure 18: **Various unconditional generation results.**

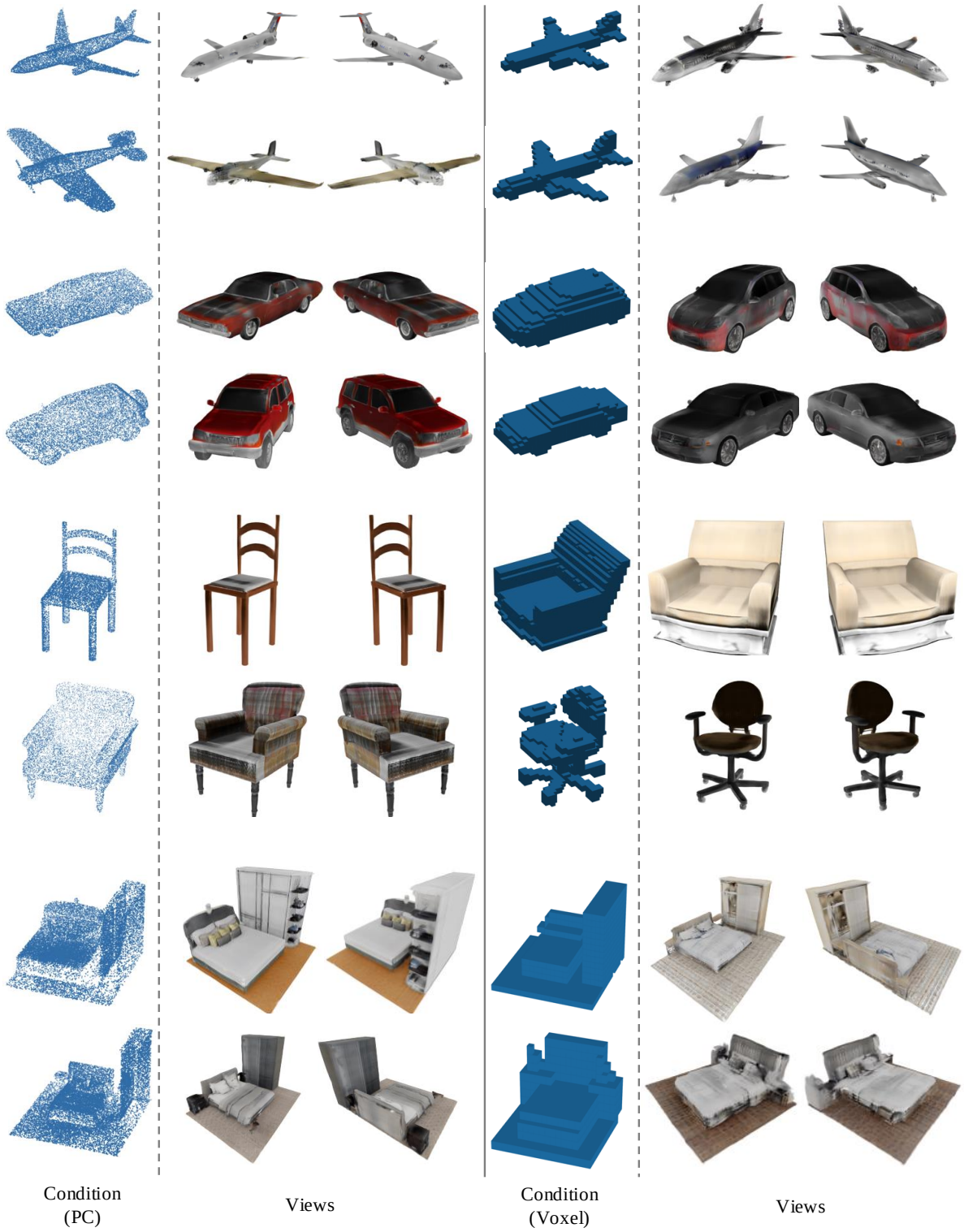


Figure 19: **Various point cloud-conditioned or voxel-conditioned generation results.**

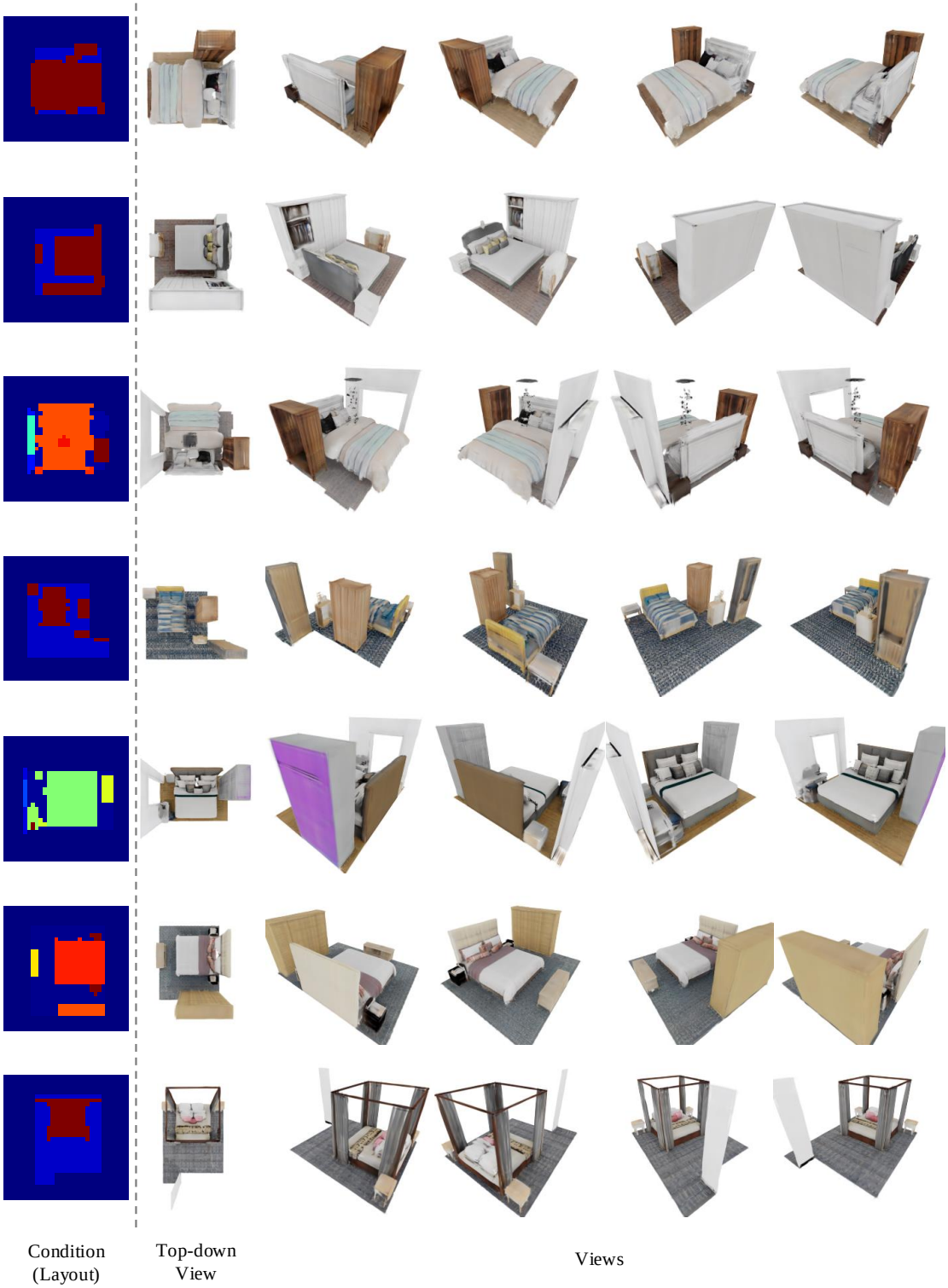


Figure 20: **Various layout-conditioned generation results on the 3D-FRONT bedroom category.**

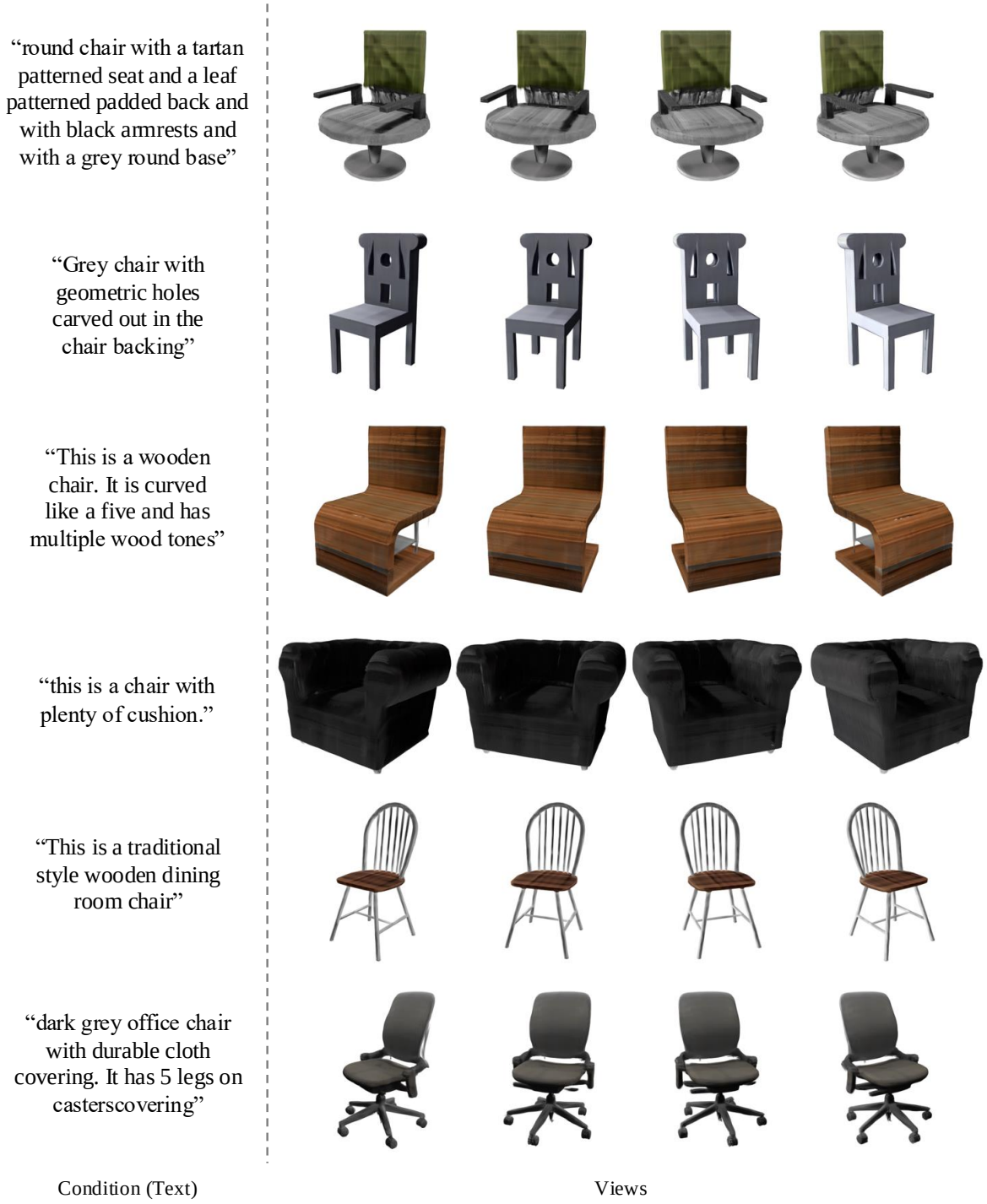


Figure 21: **Various text-conditioned generation results on the ShapeNet chair category.**

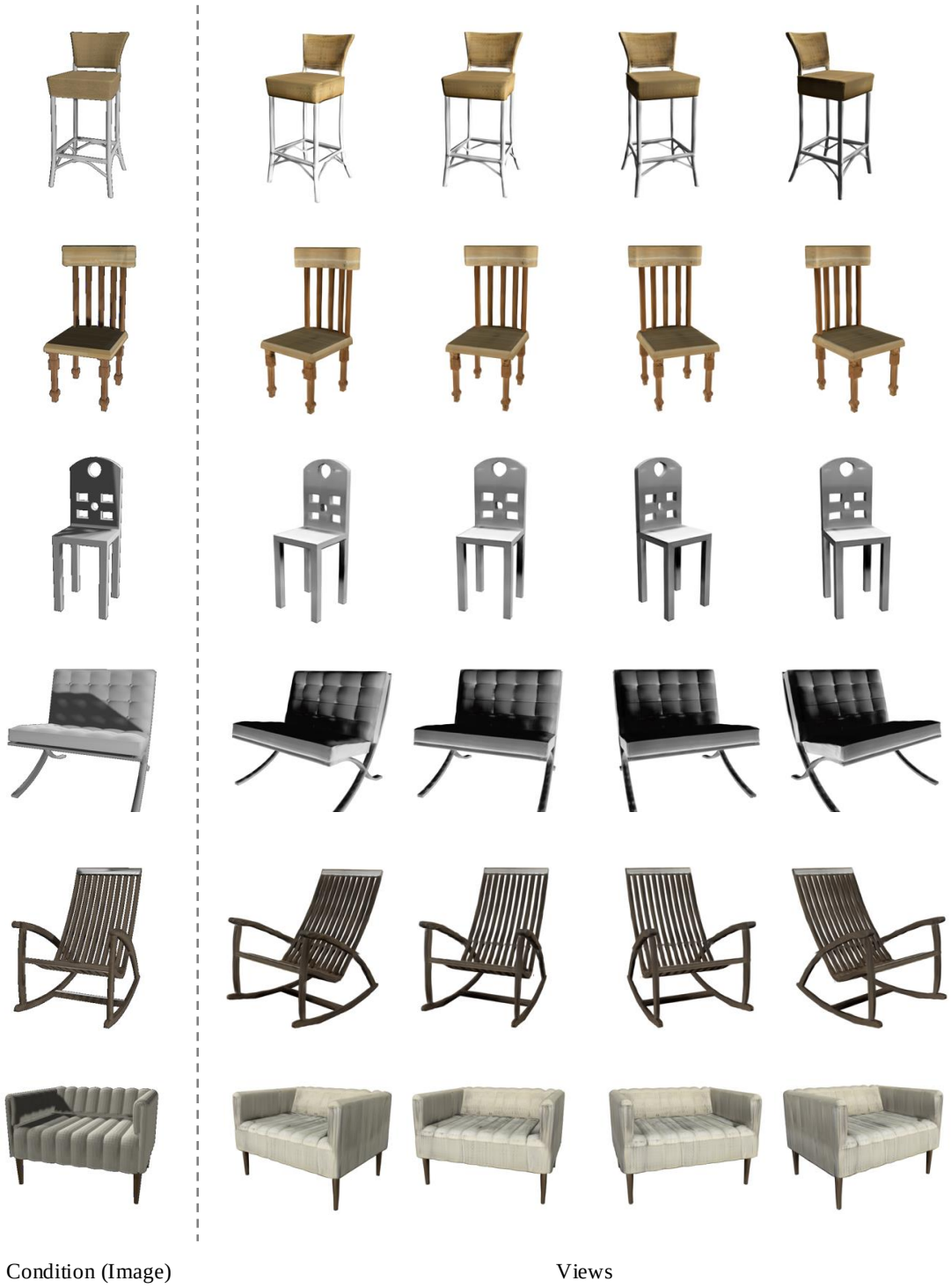


Figure 22: **Various image-conditioned generation results on the ShapeNet chair category.**

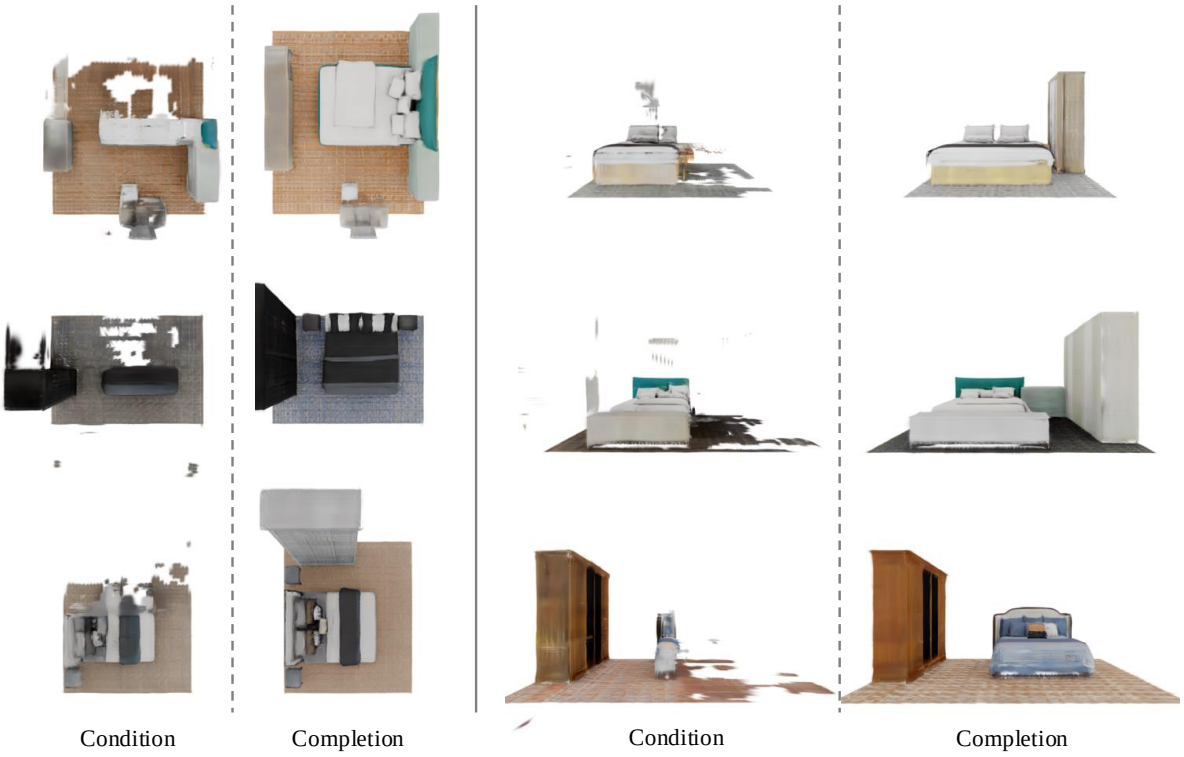


Figure 23: **Various scene completion results on the 3D-FRONT bedroom category.**

Figure 24: **Various unconditional generation results on the G-buffer Objaverse plants, food, and animals categories.**