LiTo: Surface Light Field Tokenization

Anonymous authors
Paper under double-blind review

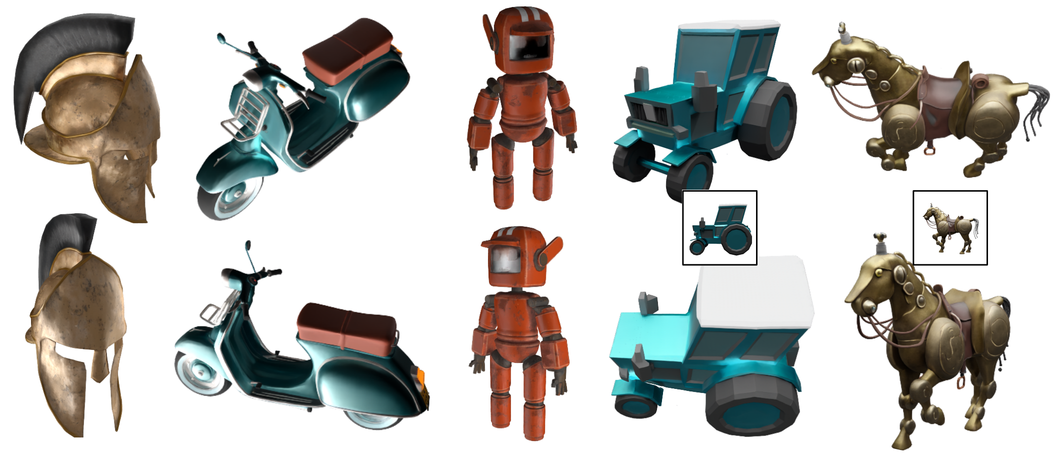


Figure 1: LiTo learns to tokenize surface light fields into a latent representation. It models both 3D geometry and view-dependent appearance like specular reflection. The figure shows reconstructed (first 3 columns) and single-image-to-3D (last two columns) surface light fields. **Please see more result videos on the supplemental website.**

Abstract

We propose a 3D latent representation that jointly models object geometry and view-dependent appearance. Most prior works focus on either reconstructing 3D geometry or predicting view-independent diffuse appearance, and thus struggle to capture realistic view-dependent effects. Our approach leverages that RGB-depth images provide samples of a surface light field. By encoding random subsamples of this surface light field into a compact set of latent vectors, our model learns to represent both geometry and appearance within a unified 3D latent space. This representation reproduces view-dependent effects such as specular highlights and Fresnel reflections under complex lighting. We further train a latent flow matching model on this representation to learn its distribution conditioned on a single input image, enabling the generation of 3D objects with appearances consistent with the lighting and materials in the input. Experiments show that our approach achieves higher visual quality and better input fidelity than existing methods.

1 Introduction

The world is filled with objects that vary widely in shape and material. Some are smooth and reflective, while others are rough, detailed or even translucent. Even familiar objects can appear differently from different viewpoints as light creates reflections and subtle color changes across their surfaces. Capturing this richness is important for building generative models of realistic objects. To do so, we need representations that can model both the underlying 3D geometry of real-world objects as well as their view-dependent appearance.

However, today in machine learning, most existing 3D representations tackle only part of this problem. Many methods are designed to capture geometry alone (He et al., 2025; Li et al., 2025a; Chang et al., 2024), aiming to recover the overall shape of objects. Other approaches (Xiang et al., 2025) include appearance information, but treat it as view-independent diffuse color. As a result, these models struggle to represent view-dependent effects such as reflections, highlights, or subtle changes in shading that are important for realistic appearance.

In this work, we aim to model both the 3D geometry and the view-dependent appearances of objects. We introduce a 3D latent representation that encodes a *surface light field* into a compact set of latent vectors. In summary, rather than encoding geometry and color only, *e.g.* with an input RGB point cloud, we additionally input viewing direction along with surface points and color, to capture how realistic materials change appearance with angle. Because a full surface light field contains highly dense information, we instead provide a random subsample of the surface light field—captured from RGB-depth multiview images—and rely on an encoder to interpolate the missing samples. This approach allows the model to reproduce view-dependent effects such as highlights and Fresnel reflections, that can be visualized via a decoder that outputs Gaussian splats with higher-order spherical harmonics (Kerbl et al., 2023). We evaluate our method by comparing its reconstruction quality against the state-of-the-art 3D latent representations (Xiang et al., 2025; Li et al., 2025a; He et al., 2025; Chen et al., 2025b; Chang et al., 2024), and find that modeling these view-dependent effects improve visual quality without significant degradation in geometric accuracy.

Building on the proposed representation, we train a latent flow matching model that learns the distribution of our 3D latent representations conditioned on a single input image. The generative model learns to infer both geometry and view-dependent appearance from images under different lighting conditions. Given an input image, the model generates a full 3D object whose shape matches the object in the image from the input viewpoint and whose appearance reflects the lighting and view-dependent material properties present in the input. Our approach connects 2D observations to 3D object generation, enabling controllable synthesis of realistic, view-dependent materials from diverse image inputs.

Our work makes the following contributions.

- We introduce a 3D latent representation that captures both geometry and view-dependent appearances by encoding surface light field information into a compact set of latent vectors.
- We design a training framework that jointly supervises geometry and appearance using random subsamples of surface light field data from RGB-depth multiview images, enabling the model to reproduce view-dependent effects such as highlights and fresnel reflections via Gaussian splats with higher-order spherical harmonics.
- We develop a latent flow matching model that learns the distribution of these latent representations conditioned on images, allowing the generation of full 3D objects whose appearances reflect the lighting and materials in the input.

Together, these components enable more accurate reconstruction and better separation of geometry and appearance than existing methods. We plan to release our code and pretrained models.

2 Related Works

A growing number of recent approaches have explored learning latent 3D representations. In Table S1, we summarize and compare their properties, including geometry and appearance modeling mechanisms, data requirements, latent dimensionality, encoder inputs, and training sets. For clarity, we review geometry-only approaches and those that jointly model geometry and appearance separately.

Geometry-only latent. A large body of work focuses on latent representations that model geometry alone. These approaches differ primarily in the underlying 3D signal they encode. PointFlow (Yang et al., 2019), ShapeGF (Cai et al., 2020), and ShapeToken (Chang et al., 2024) learn to model 3D surfaces as 3D distributions. 3DShape2VecSet (Zhang et al., 2023), CLAY (Zhang et al., 2024), TripoSG (Li et al., 2025a), and Hunyuan3D (Zhao et al., 2025), instead model shapes as occupancy or signed distance functions (SDF). Direct3D (Wu et al., 2024), XCube (Ren et al., 2024), LT3SD (Meng et al., 2025), and Make-A-Shape (Hui et al., 2024) embed geometry into dense or sparse voxel grids containing occupancy or SDF values at vertices. While grid-based methods offer structured latents, they face inherent trade-offs between spatial resolution and memory efficiency. A common limitation when relying on occupancy or SDF, however, is the reliance on significant preprocessing of the training data. Many methods require watertight meshes (Zhang et al., 2023; 2022; 2024), expensive

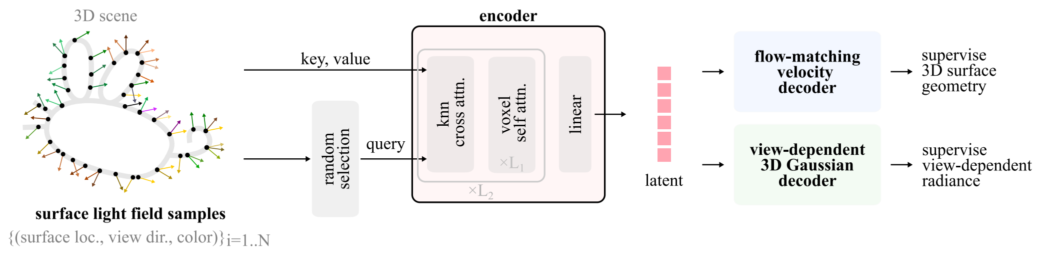

3D scene
surface light field samples
{(surface loc., view dir., color)}$_{i=1..N}$
key, value
query
random selection
encoder
knn cross attn.
voxel self attn.
×L1
×L2
linear
latent
flow-matching velocity decoder
view-dependent 3D Gaussian decoder
supervise 3D surface geometry
supervise view-dependent radiance

Figure 2: **Overview of the 3D latent representation.** Given samples of the surface light field of the scene, we learn a latent representation that reconstruct the *full* surface light field information. The encoder (pink block) condenses input information into the latent representation. We jointly supervise the latent representation to contain full 3D geometry and view-dependent radiance information beyond the input samples. In the architectures, we design localized attention pattern to improve efficiency and support 1 million input tokens.

mesh-to-field conversions, or optimization-based radiance-field fitting in order to define consistent supervision signals. Moreover, these methods capture only geometry, without appearance, texture, or view-dependent effects.

Geometry and appearance latent. More recently, a smaller set of works has begun to extend latent 3D representations beyond pure geometry to also encode appearance. Two of the most relevant are 3DTopia-XL (Chen et al., 2025b) and TRELLIS (Xiang et al., 2025).

3DTopia-XL introduces the PrimX representation, where each primitive encodes not only geometry through signed distance but also material properties such as RGB color, roughness, and metallicity. This design allows the model to generate textured 3D assets that are ready for physically based rendering. However, PrimX requires an optimization step to construct the primitive representation from meshes before training, making data preparation more demanding.

TRELLIS introduces a Structured LATent (SLAT) representation: a sparse voxel grid fused with dense multiview visual features extracted by a foundation vision model (DINOv2) to provide both geometry and appearance cues. Given the coarse geometry of an object, SLAT is constructed by averaging projected DINOv2 features from all input views. The model decodes SLAT into multiple output 3D formats, including 3D Gaussians, meshes, and radiance fields. To handle the sparsity of SLAT efficiently, TRELLIS employs transformers with windowed attention and sparse 3D convolution, and it is trained at scale on roughly 500K assets from Objaverse-XL and related datasets.

TRELLIS has several limitations relative to our approach. First, SLAT requires coarse occupancy information to be known in advance, so generation is performed in two stages, whereas our latent directly encodes complete object information and supports single-stage generation. Second, TRELLIS encodes only view-independent appearance: multiview features are mean-pooled, discarding angular variation and preventing modeling of view-dependent effects. Finally, TRELLIS generates objects in a canonical coordinate system (*i.e.*, their dataset orientation), which necessitates post-processing to align them with input images if needed. This restriction arises from its reliance on preconstructed axis-aligned voxel grids, which makes coordinate transformations like rotation during training difficult. In contrast, our model takes points as input, which allows us to apply coordinate transformations during training, ensuring generated objects are consistently oriented with respect to the input view.

3 METHOD

3.1 PRELIMINARY AND NOTATION

The surface light field jointly models both the 3D surfaces of a scene as well as the outgoing radiance from each point on the surface toward every viewing direction. In theory, if the surface light field is perfectly represented, any image captured by a camera at any arbitrary location and orientation can be directly reconstructed (Wood et al., 2000). We represent the surface light field as a 5D function $\ell(\mathbf{x}, \hat{\mathbf{d}}) : \mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}^3$, where $\mathbf{x} \in \partial\Omega$ is any 3D location on surfaces $\partial\Omega$,

$\hat{\mathbf{d}} \in S^2 = \{\mathbf{v} | \mathbf{v} \in \mathbb{R}^3, \|\mathbf{v}\| = 1\}$ is the viewing direction, and $\mathbf{c} \in \mathbb{R}^3$ is the color of the outgoing radiance from $\mathbf{x}$ toward $\hat{\mathbf{d}}$.

We use bold lowercase symbols (*e.g.*, $\mathbf{v}$) to denote vectors, bold lowercase symbols with hats (*e.g.*, $\hat{\mathbf{v}}$) for unit-norm directions, capital letters (*e.g.*, A) for matrices or transformations, and calligraphic symbols (*e.g.*, $\mathcal{S}$) for sets.

3.2 Overview

Our goal is to learn a 3D latent representation that models the surface light field of an object-centric scene with a compact set of latent vectors $\mathbf{s} \in \mathcal{S}$. Fig. 2 shows an overview of our latent representation. Our encoder E takes N samples of the surface light field defined in the following as input and outputs a small set of k latent tokens of dimension d:

$$\mathcal{X} = \{(\mathbf{x}_i, \hat{\mathbf{d}}_i, \mathbf{c}_i{=}\ell(\mathbf{x}_i, \hat{\mathbf{d}}_i))\}_{i=1}^{N}. \tag{1}$$

To learn a meaningful representation of the surface light field, we must supervise both the decoded 3D geometry as well as view-dependent radiance. A trivial solution would utilize an autoencoder formulation that directly reconstructs the input $\mathcal{X}$. However, in practice we only have sparse, discrete samples of the surface light field (*e.g.*, as rendered from multiview images of a training object), and thus such an approach may not meaningfully represent the entire continuous function ℓ. Thus, rather than directly supervising with the surface light field, we instead opt for indirect supervision with carefully-designed loss functions on decoded geometry and view-dependent appearance:

Geometry supervision. We utilize prior work (Chang et al., 2024), which models 3D surfaces as a 3D probabilistic density function that is aligned with the actual surfaces via flow matching. This formulation enables us to model 3D surfaces beyond the input 3D locations. Specifically, the latent $\mathbf{s}$ is trained to parameterize a 3D distribution $p(\mathbf{x}|\mathbf{s})$ that approximates a dirac delta function lying on 3D surfaces in the scene, *i.e.*, $p(\mathbf{x}|\mathbf{s}) \approx \delta(\mathbf{x} \in \partial\Omega)$. The flow matching formulation also optionally allows us to sample $p(\mathbf{x}|\mathbf{s})$ and get a point cloud lying on surfaces during inference, and zero-shot estimate surface normals.

The loss function follows that used by Chang et al. (2024):

$$\mathcal{L}_{\text{geo}}(\boldsymbol{\theta}) = \mathbb{E}_{t\sim u(0,1)}\mathbb{E}_{\mathbf{x}}\|V(\mathbf{x}_t; t) - (\mathbf{x} - \boldsymbol{\epsilon})\|^2 \, \mathrm{d}t\,, \tag{2}$$

where $\boldsymbol{\theta}$ is all parameters in the encoder and the decoder, t is the flow-matching time, $u(0, 1)$ is the uniform distribution between 0 and 1, $\boldsymbol{\epsilon}$ is noise sampled from standard normal distribution, $V_\theta(\mathbf{x}_t; t)$ is the flow-matching decoder that estimates the velocity at $\mathbf{x}_t = \mathbf{x} + (1 - t) * \boldsymbol{\epsilon}$, and $\mathbf{x}$ is sampled from the surface light field.

View-dependent radiance supervision. The supervision of the view-dependent radiance is through rendering multi-view images. Specifically, we convert the latent $\mathbf{s}$ into a set of 3D Gaussians, which models view-dependent color by spherical harmonics, and we render the 3D Gaussians from random viewpoints and compare with ground-truth images. The loss is

$$\mathcal{L}_{\text{radiance}}(\theta) = \mathbb{E}_{H,E}\|I_{\text{est}} - I_{\text{gt}}\|^2 + \lambda \text{ lpips}\left(I_{\text{est}}, I_{\text{gt}}\right), \tag{3}$$

where $I_{\text{est}} = \texttt{Render}(D(\mathbf{s}, \mathcal{O}), H, E)$ is the rendered image from 3D Gaussians at camera pose H and intrinsic E, $I_{\text{gt}} = \texttt{Render}(\text{object}, H, E)$ is the ground-truth image, D is the Gaussian decoder that will be detailed below, $D(\mathbf{s}, \mathcal{O})$ are the estimated 3D Gaussians given the latent $\mathbf{s}$ and a low-resolution sparse occupancy grid $\mathcal{O}$ constructed from the sampled point cloud or an occupancy estimator, and $\boldsymbol{\theta}$ is all parameters in the encoder and the decoder. In all experiments, we use $\lambda = 0.2$.

In the rest of this section, we discuss the architectures for our surface light-field encoder, geometry decoder and Gaussian decoder in more detail.

3.3 Encoder

We first describe how we sample surface light field to obtain the input to the encoder and the samples for the empirical mean in Eq. (2). Then we detail our encoder architecture.

Input. To sample from the surface light field $\ell(\mathbf{x}, \hat{\mathbf{d}})$, we need to sample random surface locations and view directions. We achieve this by densely rendering multi-view RGBD images. Since we focus on object-centric scenes, the cameras are placed uniformly on a sphere surrounding the object. The surface location $\mathbf{x}$ can be obtained by back-projecting the depth map, view direction $\hat{\mathbf{d}}_i$ is derived from the pinhole camera model, and $\mathbf{c}_i$ from the pixel color[1]. This operation densely samples both the surfaces and viewing directions and returns $\mathcal{X} = \{(\mathbf{x}_i, \hat{\mathbf{d}}_i, \mathbf{c}_i)\}_{i=1}^N$.

In our experiments, we box-normalize the scene to $[-1, 1]$, and we render 150 images of resolution 1036×1036 with 40 degree field of view, uniformly on a sphere of radius 3.5. This provides 160 million samples of light field ℓ introduced in Sec. 3.1, of which we randomly sample $N{=}2^{20}$ as our input to the encoder and the rest to serve as the ground-truth to supervise Eq. (2).

Architecture. We use Perceiver IO (Jaegle et al., 2022) as our encoder, which is widely used in prior latent 3D representations (Chang et al., 2024; Zhang et al., 2023; Li et al., 2025a). The encoder contains cross and self attention blocks, and the number of the initial queries of the first cross attention block determines the number of output latent tokens. The output of the Perceiver IO is passed to a linear layer to reduce the latent dimension to d. Our latent is thus a set of k tokens of d dimension discussed in Sec. 3.2.

To capture enough information from light field ℓ introduced in Sec. 3.1, we use $N = 2^{20}$ ($\sim$1 million) samples as input. However, the large number of input tokens makes the typical cross attention in Perceiver IO computationally expensive. To solve the problem, we are inspired by the non-overlapping patchification in Vision Transformers (Dosovitskiy et al., 2021), which converts dense pixels into coarse tokens. Instead of using a convolution layer to aggregate information from individual 16×16 patches into tokens, we use cross attention. However, our inputs are scattered points on 3D surfaces instead of pixels on a regular grid, and it is non-trivial to patchify 3D surfaces. We design an approximation of 2D patchification on 3D surfaces with k-nearest neighbor. Specifically, given the input samples $\mathcal{X}$ in Eq. (1), we first randomly select k samples as the query $\mathcal{Q}$ to the first cross attention layer, similar to Zhang et al. (2023). The number of samples is equal to the number of latent tokens, which is 8192 in our setup. To patchify 3D surface, for each sample $\mathbf{x} \in \mathcal{X}$ we find its closest point in $\mathcal{Q}$ in terms of ℓ_2 distance of $\mathbf{x}$ and assign the index of the closest point to the sample. Finally, during the cross attention, a query only attends to input samples that have its index. This operation can be implemented by standard libraries like `xformers` (Lefaudeux et al., 2022) or `FlashAttention` (Dao, 2024).

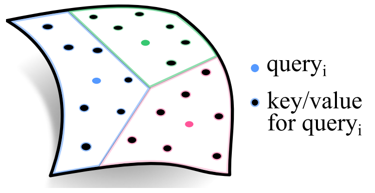


Figure 3: 3D patchification

An illustration is shown in Fig. 3. Note that this is an approximation because we use ℓ_2 distance of $\mathbf{x}$ instead of geodesic distance. When there are more than one surface lie in the neighborhood, the query will attend across surfaces. As ℓ_2 distance is much faster to compute than the geodesic distance, we think it is a good trade-off.

For self attention, we use a voxel-based attention mechanism. Specifically, during self attention, tokens that lie within a predefined coarse voxel grid attend to each other, and the coarse voxel grid shifts by a half cell width every layer. Note that unlike TRELLIS (Xiang et al., 2025), whose tokens lie on a voxel grid, our tokens have continuous coordinates and do not lie on a voxel grid. Overall, the encoder has 59.2 million parameters. Together with decoders below, the model is trained with 256 batch size for 90k iterations on 64 GPUs for 9 days.

3.4 Decoder

Flow-matching velocity decoder. We utilize the same flow-matching velocity decoder used by Chang et al. (2024). Specifically, it takes the latent $\mathbf{s}$, a 3D location, and flow-matching time as input, and it predicts the flow-matching velocity at the 3D location. To ensure we model a 3D distribution, *i.e.*, $p(\mathbf{x}|\mathbf{s})$, the decoder processes each 3D point independently (only cross attention and point-wise operations are used). The decoder has 8.8 million parameters.

[1]We assume the depth map measures the distance to the first intersection point of the scene, regardless of transparency. For example, in blender, this can be achieved by setting the alpha threshold to be 0.

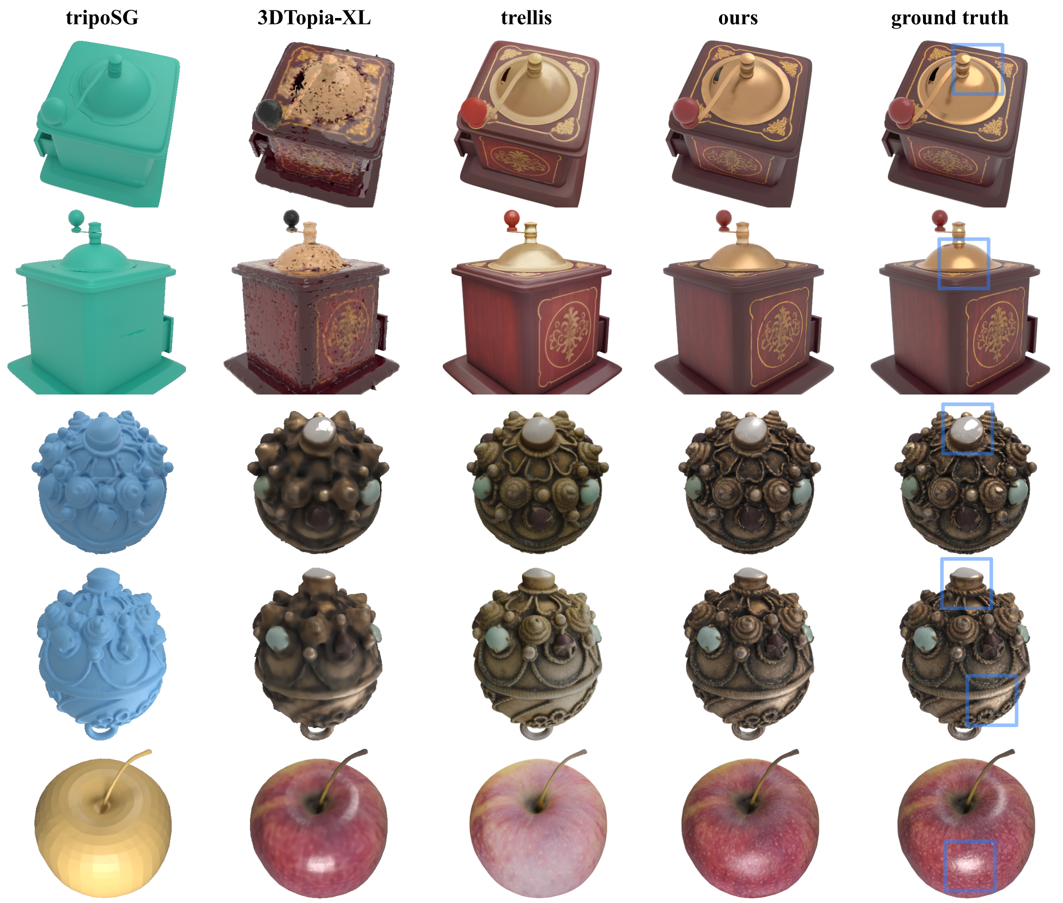


Figure 4: **Reconstruction results on various lighting conditions.** Boxes on ground-truth highlight specular and Fresnel reflection. Please refer to Tab. 1 for quantitative results. Mesh credit: DigitalSouls (2019); 3Dji (2025b); of Małopolska (2020).

View-dependent Gaussian decoder. Similar to our encoder, we use a Perceiver IO architecture (Jaegle et al., 2022) to implement our Gaussian decoder. We use a low-resolution sparse occupancy grid for our initial queries, and cross attend to the predicted latent $\mathbf{s}$. At the output, we use a small MLP to output 64 3D Gaussians for each occupied voxels. Unlike past work that only uses Gaussians with view-independent color (Xiang et al., 2025), our decoder predicts Gaussians of spherical harmonics degree 3 to model view-dependent radiance. The decoder has 77.3 million parameters.

At training time, we use ground-truth occupancy for the decoder queries, like recent work leveraging structured latent representations (Xiang et al., 2025; He et al., 2025; Wu et al., 2025). After learning the representation, we can either use points sampled from the aforementioned flow-matching geometry decoder or alternatively train a downstream occupancy decoder, to directly predict sparse occupied voxels from the encoded latent. Thus, at generation time, our approach does not require a second generative model to predict occupancy unlike structured latent-based approaches (Xiang et al., 2025; He et al., 2025; Wu et al., 2025), simplifying the overall pipeline.

3.5 Generative Model

To demonstrate our latent representation, we train a flow-matching model that generates 3D latents conditioned on an image of an object. We rely on a standard Diffusion Transformer (DiT) architecture (Peebles & Xie, 2022), with a zero-initialized learnable positional encoding for each latent token. The input image is encoded by DINOv2-large image embeddings (Oquab et al., 2024) and a learnable patchification layer. While we originally considered using more explicit camera geometry encoding, *e.g.*, Plucker ray embeddings, we found in practice that such an approach reduced overall performance — please see the supplement for an ablation. In total, the model has 623 million parameters.

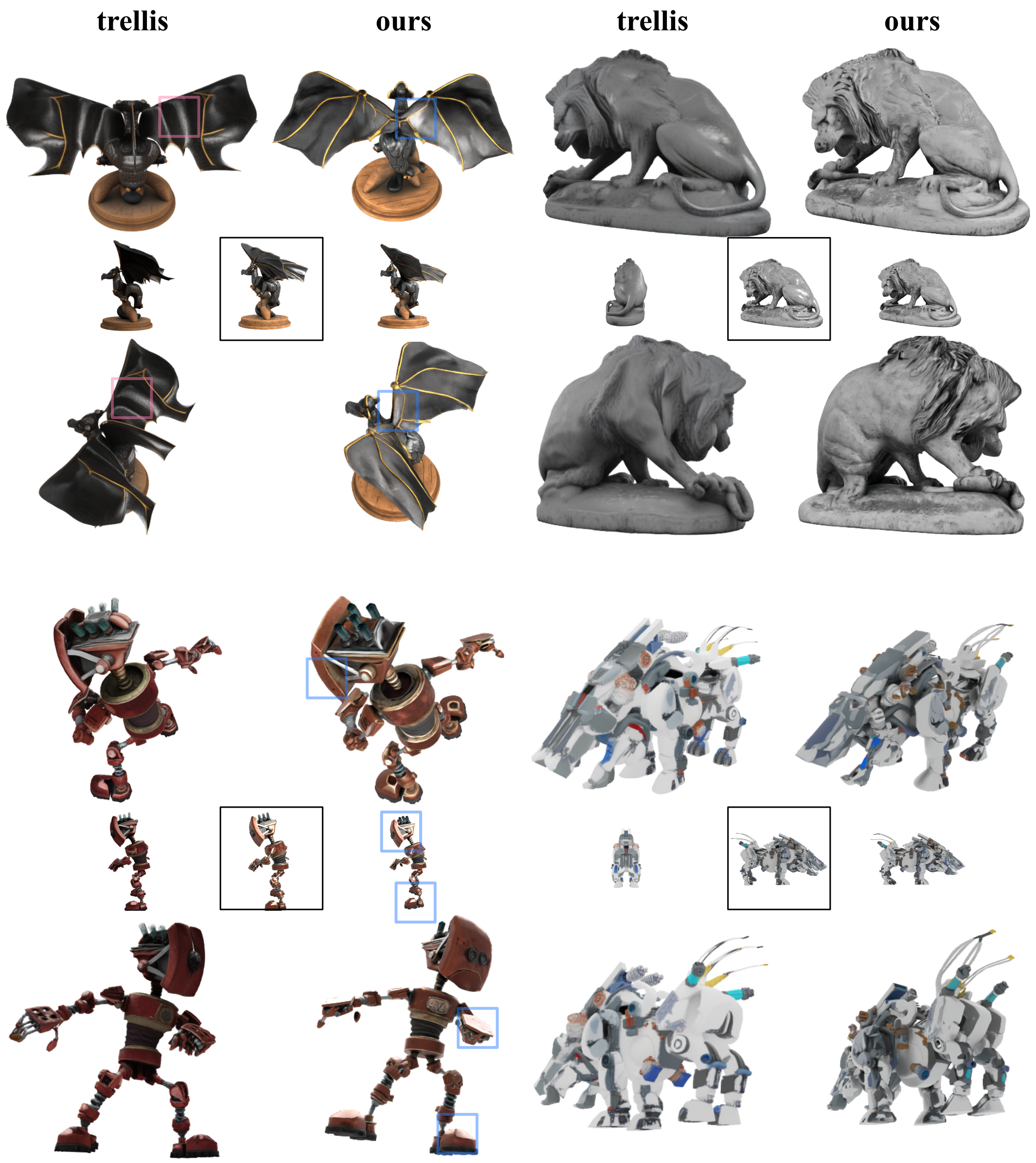


Figure 5: **Single image to 3D results.** The input image is shown at the center of each set with black border. The rendering of the generated image at the input view (+x, 0, 0) is shown with the input image. Please refer to Tab. 3 for quantitative results. Mesh credit: Vetech82 (2021); Rigsters (2017); 3d coat (2015); 3Dji (2025a).

4 Experiments

Following the typical latent generation pipeline, we first train the latent representation, and once learned, we then train a latent flow-matching model conditioned on an input image. We discuss the training and evaluation of our latent representation in Section 4.1, and our generative image-to-3d model in Section 4.2.

4.1 Reconstruction

Datasets. We train the encoder-decoder on the 500k high-quality object subset of Objaverse-XL (Deitke et al., 2023) as selected by TRELLIS (Xiang et al., 2025). Unlike TRELLIS, instead of using all 500k objects for training, we divide the data into training, validation, and test sets in

Table 1: **Reconstruction on Toys4k.** We provide input needed by individual methods. TRELLIS (Xiang et al., 2025) takes the ground-truth mesh and 150 sphere-distributed renderings. Ours uses RGBD images from 150 evenly distributed views. For appearance evaluation, we render each model's output from 100 random cameras, varying difficulty by adjusting camera radius. Please refer to Fig. 4 for qualitative results and Sec. C for comprehensive quantitative results. The better one is highlighted.

Method	Simple, Camera Radius [3, 4]			Hard, Camera Radius [1, 3]		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
TRELLIS	31.12±3.39	0.974±0.022	0.034±0.022	27.57±3.38	0.941±0.050	0.090±0.055
Ours	34.16±3.39	0.985±0.016	0.023±0.018	32.36±3.77	0.967±0.040	0.055±0.046

Table 2: **Geometric reconstruction evaluation**. We report **squared** Chamfer distances multiplied by 10^4 for readability, computed using 100k sampled points each from ground-truth and reconstruction. As 3DTopia-XL (Chen et al., 2025b) and TripoSG (Li et al., 2025a) can be sensitive to input geometry, we also list variants with their worst-performing 10% of objects removed. We separate our tested approaches based on those that require ground-truth coarse geometry for decoding the latent representation, and those that do not utilize this information. Our method outputs the best geometry among the approaches in the latter category, and it is competitive with the techniques in the former. Red highlights the best method in each category.

	Method	Appearance	Latent size	PBR-Objaverse	Toys4k	GSO
0	GT	-	-	0.482±0.245	0.422±0.268	0.533±0.273
Requires coarse geometry oracle:						
1	TripoSF (He et al., 2025)	✗	≈ 244k × 11	0.632±0.583	0.557±0.593	0.714±0.738
2	TRELLIS (Xiang et al., 2025)	✓	≈ 20k × 11	0.616±0.277	0.608±0.437	0.737±0.331
3-1	3DTopia-XL (Chen et al., 2025b)	✓	2048 × 64	124.8±720.2	17.52±115.8	0.693±0.434
3-2	(worst 10% removed)	✓	2048 × 64	4.702±13.47	0.895±1.330	0.612±0.331
4	Ours (oracle, mesh decoder)	✓	8192 × 32	0.558±0.316	0.506±0.439	0.654±0.338
Does not utilize coarse geometry oracle:						
5-1	TripoSG (Li et al., 2025a)	✗	2048 × 64	33.43±74.25	36.49±63.83	46.41±88.86
5-2	(worst 10% removed)	✗	2048 × 64	14.88±21.35	20.36±27.33	21.89±39.35
6	Shape Tokens (Chang et al., 2024)	✗	1024 × 16	1.116±0.447	1.062±0.591	1.240±0.482
7-1	Ours (no mesh decoder)	✓	8192 × 32	0.720±0.321	0.668±0.385	0.816±0.396
7-2	Ours (mesh decoder)	✓	8192 × 32	0.569±0.332	0.524±0.484	0.665±0.355

an 8:1:1 ratio. For each object, we pair it with 3 lighting conditions: 1) fixed smooth area lighting (matching TRELLIS)[2], 2) an all-white environment map, and 3) randomly placed lights. For each configuration, we render using Blender from 150 viewpoints uniformly distributed on a sphere, to sample the surface light field as input for our encoder. We render from 100 random viewpoints to supervise our view-dependent Gaussian decoder.

We evaluate the models on Toys4k (Stojanov et al., 2021), GSO (Downs et al., 2022), and Objaverse-XL (Deitke et al., 2023). For Objaverse-XL, we select a subset of 200 objects with PBR materials, which we dub PBR-Objaverse.

Qualitative results: Fig. 4 shows a few objects with view-dependent appearance, including specular reflections from metallic surfaces and Fresnel reflections when viewed at grazing angles. **Please see our supplemental website for more informative visualizations.**

Quantitative results (appearance): To evaluate appearance quality, we render the 3DGS from 100 random views on a sphere and measure PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018). Tab. 1 shows reconstruction metrics under different zoom-in levels on the Toys4k dataset rendered with TRELLIS's training lighting condition. Our surface light-field representation outperforms competitor appearance representations across all the tested metrics. More evaluations on other datasets are described in Sec. C.

Quantitative results (geometry): To evaluate the quality of reconstructed 3D geometry, we estimate ground truth point clouds by unprojecting the rendered depth of a target object from 100 uniformly distributed views on the sphere and randomly selecting 100k reference points. We then compute Chamfer distance between these ground truth point clouds and reconstructed point clouds from latent

[2]https://github.com/microsoft/TRELLIS/blob/6b0d64751ad54d9c3/dataset_toolkits/blender_script/render.py#L178-L209

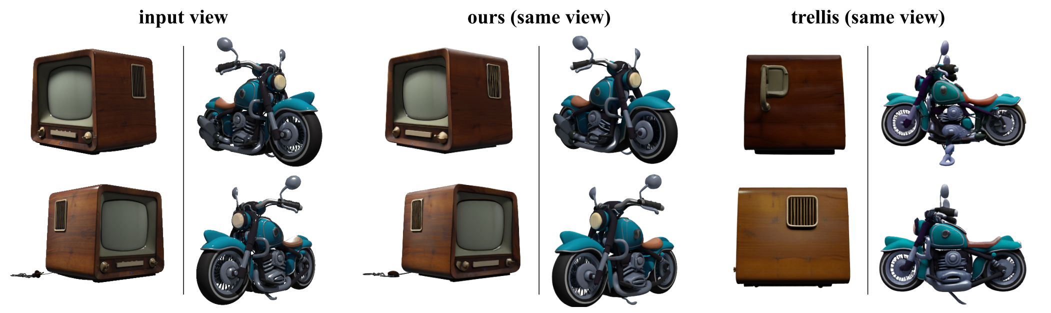


Figure 6: **Fidelity to input view.** Our image-to-3d generative model respects the coordinate system of the input view. In contrast, existing state-of-the-art techniques, *e.g.*, TRELLIS (Xiang et al., 2025), do not. Mesh credit: of Małopolska (2016); animanyarty (2022).

representations. For our method and Chang et al. (2024), we sample 100k points from the flow-matching velocity decoder to produce the output point cloud. For baselines that output meshes (Xiang et al., 2025; Li et al., 2025a; He et al., 2025), similar to the ground truth points, we unproject rendered depth from another set of 100 views on the sphere and select 100k points for the Chamfer calculation.

Table S9 shows geometry evaluation when input is lit with TRELLIS's training lighting condition. Our method is competitive with recent geometry-only latent representations, despite additionally representing appearance information. Furthermore, our geometry estimates do not utilize additional ground truth coarse geometry information that other state-of-the-art approaches require (Xiang et al., 2025; He et al., 2025).

Table 3: **Single-image-conditioned generation on Toys4k.** KID is reported by $\times 100$. CFG scale for both models are 3.0. The best is highlighted. See Fig. 5, 6 and supp. website for qualitative results.

Method	CLIP↑	Conditioning View				Novel View			
		FID↓	KID↓	FID_{dino}↓	KID_{dino}↓	FID↓	KID↓	FID_{dino}↓	KID_{dino}↓
TRELLIS	0.899±0.045	12.84	0.088	84.692	2.311	7.600	0.100	67.458	3.166
Ours	0.905±0.041	6.219	0.009	41.621	1.333	6.216	0.058	66.530	3.522

4.2 Generation

We train our image-to-3D DiT on the same tokenizer-training set for 280k iterations (effective batch size of 256 on 128 H100 GPUs for 9 days). We evaluate generation results with the same fixed area lighting as TRELLIS to allow a fair comparison. We qualitatively evaluate our model's performance in Fig. 5. Our model generates complex geometry and view-dependent appearance, despite being trained on other lighting types. **Please see our supplemental website for more informative visualizations.** We also visualize our model's input view fidelity compared to TRELLIS in Fig. 6.

To quantitatively evaluate the quality of our image-to-3d generative model, we calculate two distribution-wise metrics. First, to evaluate the fidelity of the generative model to the input content, we render the generated 3D asset at the same pose as the conditioning view. As shown in Table 3, our approach produces significantly improved FID and KID scores in this setting compared to TRELLIS. Second, to measure the overall quality of the generated asset, we render from four novel views distributed around the object at a pitch of 30°, following the evaluation setup of TRELLIS (Xiang et al., 2025). As shown in Table 3, despite our model's increased faithfulness to the input view, the overall generation performance does not significantly degrade.

5 Conclusion

We propose an autoencoder that learns a compact latent space for view-dependent 3D assets. In particular, we build an encoding of the surface light field, that can be easily produced via multi-view RGBD rendering. With a flow-matching geometry decoder and a view-dependent Gaussian decoder, our representation can be easily applied with an off-the-shelf DiT for generating view-dependent 3D assets. We validate the performance of our view-dependent 3d representation in both reconstruction and generation.

REFERENCES

1812panorama. Drummer of the revel infantry regiment, 2019. URL https://skfb.ly/6Xq6W. Licensed under CC0 Public Domain. 16

3d coat. Robot steampunk 3d-coat 4.5 pbr, 2015. URL https://skfb.ly/EEIE. Licensed under Creative Commons Attribution. 7

3Dji. Mechanical beast, 2025a. URL https://skfb.ly/pAFoE. Licensed under Creative Commons Attribution. 7

3Dji. Coffee grinder, 2025b. URL https://skfb.ly/pzpn7. Licensed under Creative Commons Attribution. 6

a108082046. Telephone, 2022. URL https://skfb.ly/ovDBJ. Licensed under CC Attribution. 16

AdamJonesCGD. Conrad carriage, 2020. URL https://skfb.ly/oyy9z. Licensed under CC Attribution. 16

Conseil des musées Alienor.org. La grand' goule, 2016. URL https://skfb.ly/UvvD. Licensed under CC Attribution-NonCommercial-NoDerivs. 16

alzarac. Hypostomus / coroncoro, 2019. URL https://skfb.ly/6W8Dz. Licensed under CC Attribution. 16

animanyarty. Motorcycle, 2022. URL https://sketchfab.com/3d-models/motorcycle-38404e2077ca4b209cd2f1db30541b94. Licensed under Creative Commons Attribution. 9

Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *European Conference on Computer Vision (ECCV)*, pp. 364–381. Springer, 2020. 2, 15

Jen-Hao Rick Chang, Yuyang Wang, Miguel Angel Bautista Martin, Jiatao Gu, Xiaoming Zhao, Josh Susskind, and Oncel Tuzel. 3D Shape Tokenization via Latent Flow Matching. *arXiv preprint arXiv:2412.15618*, 2024. 1, 2, 4, 5, 8, 9, 14, 15, 21

Rui Chen, Jianfeng Zhang, Yixun Liang, Guan Luo, Weiyu Li, Jiarui Liu, Xiu Li, Xiaoxiao Long, Jiashi Feng, and Ping Tan. Dora: Sampling and Benchmarking for 3D Shape Variational Auto-Encoders. In *CVPR*, 2025a. 15

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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26576–26586, 2025b. 2, 3, 8, 14, 15, 21

Gene Chou, Yuval Bahat, and Felix Heide. Diffusion-sdf: Conditional generative modeling of signed distance functions. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 2262–2272, 2023. 15

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. 5

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. 7, 8

DigitalSouls. Delicious red apple, 2019. URL https://skfb.ly/6RxAt. Licensed under Creative Commons Attribution-NonCommercial. 6, 26

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021. 5

Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Michael Hickman, Krista Reymann, Thomas Barlow McHugh, and Vincent Vanhoucke. Google Scanned Objects: A High-Quality Dataset of 3D Scanned Household Items. 2022. 8

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025. 19

GJ2012. Toaster - kitchenaid artsan, 2013. URL https://www.blendswap.com/blend/8552. Licensed under CC0. 16

Xianglong He, Zi-Xin Zou, Chia-Hao Chen, Yuan-Chen Guo, Ding Liang, Chun Yuan, Wanli Ouyang, Yan-Pei Cao, and Yangguang Li. SparseFlex: High-Resolution and Arbitrary-Topology 3D Shape Modeling. *arxiv*, 2025. 1, 2, 6, 8, 9, 14, 15, 21

Ka-Hei Hui, Aditya Sanghi, Arianna Rampini, Kamal Rahimi Malekshan, Zhengzhe Liu, Hooman Shayani, and Chi-Wing Fu. Make-a-shape: a ten-million-scale 3d shape model. In *International Conference on Machine Learning (ICML)*, 2024. 2, 15

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver IO: a General Architecture for Structured Inputs & Outputs. In *ICLR*, 2022. 5, 6

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *TOG*, 2023. 2

Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, Daniel Haziza, Luca Wehrstedt, Jeremy Reizenstein, and Grigory Sizov. xformers: A modular and hackable transformer modelling library. https://github.com/facebookresearch/xformers, 2022. 5

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. *arXiv*, 2025a. 1, 2, 5, 8, 9, 15, 21

Zhihao Li, Yufei Wang, Heliang Zheng, Yihao Luo, and Bihan Wen. Sparc3D: Sparse representation and construction for high-resolution 3d shapes modeling. *arXiv preprint arXiv:2505.14521*, 2025b. 15

S. Luo and W. Hu. Diffusion probabilistic models for 3d point cloud generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 15

Quan Meng, Lei Li, Matthias Nießner, and Angela Dai. LT3SD: Latent trees for 3d scene diffusion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 650–660, 2025. 2, 15

nastasyas. Gart_220_centaur, 2019. URL https://skfb.ly/6WR7W. Licensed under CC Attribution. 16

Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 15

Virtual Museums of Małopolska. Black and white "belweder", 2016. URL https://skfb.ly/NnEr. Licensed under Creative Commons 0 Public Domain. 9

Virtual Museums of Małopolska. Coffee grinder, 2020. URL https://skfb.ly/6VyBJ. Licensed under Creative Commons 0 Public Domain. 6

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Q. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russ Howes, Po-Yao (Bernie) Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Huijiao Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. *TMLR*, 2024. 6

William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022. 6

Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. XCube: Large-Scale 3D Generative Modeling using Sparse Voxel Hierarchies. In *CVPR*, 2024. 2, 15

Rigsters. Lion crushing a serpent, 2017. URL https://skfb.ly/68s9T. Licensed under Creative Commons Attribution. 7

Stefan Stojanov, Anh Thai, and James M. Rehg. Using Shape to Categorize: Low-Shot Learning with an Explicit Shape Bias. In *CVPR*, 2021. 8

Zhicong Tang, Shuyang Gu, Chunyu Wang, Ting Zhang, Jianmin Bao, Dong Chen, and Baining Guo. Volumediffusion: Flexible text-to-3d generation with efficient volumetric encoder. *arXiv preprint arXiv:2312.11459*, 2023. 15

Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:10021–10039, 2022. 15

Vetech82. Metal dragon, 2021. URL https://skfb.ly/o8wB9. Licensed under Creative Commons Attribution. 7, 26

Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, pp. 195–206, Goslar, DEU, 2007. Eurographics Association. ISBN 9783905673524. 22

Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. Moge-2: Accurate monocular geometry with metric scale and sharp details, 2025. URL https://arxiv.org/abs/2507.02546. 17

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 8

Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pp. 287–296, 2000. 3

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Jingxi Xu, Philip Torr, Xun Cao, and Yao Yao. Direct3D: Scalable Image-to-3D Generation via 3D Latent Diffusion Transformer. In *NeurIPS*, 2024. 2, 15

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Yikang Yang, Yajie Bao, Jiachen Qian, Siyu Zhu, Xun Cao, Philip Torr, et al. Direct3d-s2: Gigascale 3d generation made easy with spatial sparse attention. *arXiv preprint arXiv:2505.17412*, 2025. 6, 14, 15

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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21469–21480, 2025. 1, 2, 3, 5, 6, 7, 8, 9, 14, 15, 17, 18, 19, 21

Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D point cloud generation with continuous normalizing flows. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. 2, 15

Jiayu Yang, Taizhang Shang, Weixuan Sun, Xibin Song, Ziang Chen, Senbo Wang, Shenzhou Chen, Weizhe Liu, Hongdong Li, and Pan Ji. Pandora3D: A Comprehensive Framework for High-Quality 3D Shape and Texture Generation. *arXiv*, 2025. 15

Lior Yariv, Omri Puny, Oran Gafni, and Yaron Lipman. Mosaic-sdf for 3d generative models. In *CVPR*, 2024. 15

Biao Zhang, Matthias Nießner, and Peter Wonka. 3DILG: Irregular Latent Grids for 3D Generative Modeling. In *NeurIPS*, 2022. 2, 15

Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models. *TOG*, 2023. 2, 5, 15

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets. *TOG*, 2024. 2, 15

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. *CVPR*, 2018. 8

Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, BIN FU, Tao Chen, Gang YU, and Shenghua Gao. Michelangelo: Conditional 3D Shape Generation based on Shape-Image-Text Aligned Latent Representation. In *NeurIPS*, 2023. 15

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. *arXiv*, 2025. 2, 15

Appendix – LiTo: Surface Light Field Tokenization

This supplement is organized as follows:

1. Sec. A discusses more on related works;
2. Sec. B discusses limitations;
3. Sec. C provides more comprehensive reconstruction quantitative results;
4. Sec. D provides more comprehensive generation quantitative results;
5. Sec. E introduces more implementation details;
6. Sec. F showcases more studies.

A More Related Works

Tab. S1 provides an overview of related works with respect to 1) how they model the geometry; 2) how they model the appearance; 3) the requirements on the data preparation to enable the model training; 4) the compactness of the latent size; 5) the input to the encoder and the training dataset.

B Limitations

We utilize 3D Gaussians with spherical harmonics to model surface light field. While we show that the improved reconstruction quality as we increase the degree of the spherical harmonics, we are constraint by the 3DGS implementation that supports up to degree 3, which limits our capability to faithfully reconstruct transparent or high-frequency specularities.

LiTO slightly underperforms some recent work in 3D representation learning in terms of geometric reconstruction accuracy. We hypothesize a few reasons. First, as mentioned in the main paper, many recent approaches (He et al., 2025; Xiang et al., 2025; Wu et al., 2025; Chen et al., 2025b) rely on structured representations and take coarse ground-truth geometry as input. This allows the latent representation to focus on local finegrind geometry as coarse information is provided at decoding that our approach does not utilize. Second, our approach relies on the flow-matching velocity decoder proposed by (Chang et al., 2024) to estimate geometry. While this decoder has many theoretical strengths, *e.g.*, zero-shot normal estimation, it can also produce noise in the geometry estimation thanks to the inherent probabilistic decoding. The small capacity of the velocity decoder limits the frequency of the learned score function and in terms contributes to the noise in the sampled point cloud, resulting in increased Chamfer distance.

C Comprehensive Reconstruction Results

We provide comprehensive quantitative results for reconstruction in Tab. S2, S3, and S4. As discussed in Sec. 4.1, we pair each dataset with three distinct lighting conditions to thoroughly evaluate the appearance modeling capabilities of our method. Unlike previous approaches, which primarily assess performance on zoomed-out views, we additionally evaluate appearance modeling under close-up settings. Close-up views demand greater fidelity in capturing high-frequency details, where all methods face challenges; nevertheless, LiTo consistently demonstrates the most robust performance.

Further, we provide qualitative results for reconstructed mesh in Fig. S1.

C.1 Ablations on Model Designs

As far as we know (see Tab. S1), we are the first to utilize 1) viewing directions in the encoder; and 2) higher order spherical harmonics in the decoder during 3D asset tokenization training. Thus, we are mainly interested in understanding the effects of these design choices.

When examining LPIPS across Tab. S2, S3, and S4, we observe: 1) increasing the degree of spherical harmonics from 0 to 3 improves the capacity consistently, *e.g.*, from row 1-3 to 1-6 (or row 2-3 to 2-6, 3-3 to 3-6) in all three tables; and 2) simply adding ray information does not directly enhance

Table S1: **Recent latent 3D representations.** The table provides a summary of recent 3D representations and their properties. We compare the properties that are relevant to machine learning applications. *Minimal preprocessing* indicates how easy is it to utilize a 3D dataset (*e.g.*, do we need to convert data to watertight meshes, do we need optimization radiance fields to acquire the actual training dataset). *Continuous latent* indicates whether the 3D representation is fully differentiable (*e.g.*, no graph topology or sparsity patterns). Total latent dimension indicates the total size to represent one scene. Note that there may be multiple variants of the same method with different latent dimensions. We choose the representative one in each paper. * indicates a second generative model is used in the paper to add texture to a texture-less meshes.

name	**geometry**	**appearance**	**data requirements**	**total latent dimension**	**input to encoder**	**training dataset**
DDPM-PointCloud (2021)	p(xyz)	-	point cloud	256	point cloud ($\mathbf{x}$)	ShapeNet
PointFlow (2019)	p(xyz)	-	point cloud	512	point cloud ($\mathbf{x}$)	ShapeNet
ShapeGF (2020)	p(xyz)	-	point cloud	256	point cloud ($\mathbf{x}$)	ShapeNet
Shape Token (2024)	p(xyz)	-	point cloud	1024×16	point cloud ($\mathbf{x}$)	Objaverse
Ours	p(xyz)	view-dep. 3DGS	multiview RGBD	8192×32	surface light field ($\mathbf{x}, \mathbf{c}, \hat{\mathbf{d}}$)	Objaverse, ObjaverseXL
Point-E (2022)	fixed size point set	diffuse RGB	point cloud ($\mathbf{x}$)	-	-	proprietary dataset
LION (2022)	fixed size point set	-	point cloud	128 + 8192	point cloud ($\mathbf{x}$)	ShapeNet
3DShape2VecSet (2023)	occupancy field	-	watertight mesh	512×32	point cloud ($\mathbf{x}$)	ShapeNet-watertight
3DILG (2022)	occupancy field	-	watertight mesh	512×2	point cloud ($\mathbf{x}$)	ShapeNet-watertight
Michelangelo (2023)	occupancy field	-	watertight mesh	$512 \times 64 + 768$	point cloud ($\mathbf{x}, \hat{\mathbf{n}}$)	ShapeNet, 3D cartoon monster
CLAY (2024)	occupancy field	-*	watertight mesh	2048×64	point cloud ($\mathbf{x}$)	Objaverse
Dora (2025a)	occupancy field	-	watertight mesh	1280×64	point cloud ($\mathbf{x}$)	Objaverse
Pandora3D (2025)	occupancy field	-*	watertight mesh	2048×64	point cloud ($\mathbf{x}, \hat{\mathbf{n}}$)	Objaverse, ObjaverseXL, ABO, BuildingNet, HSSD, Toy4k, polygone dataset, proprietary
Direct3D (2024)	occupancy grid	-	watertight mesh	$3 \times 32 \times 32 \times 16$	point cloud ($\mathbf{x}, \hat{\mathbf{n}}$)	proprietary dataset
Direct3D-s2 (2025)	SDF grid	-	watertight mesh	$(128^3 \times 16)$	point cloud ($\mathbf{x}, \hat{\mathbf{n}}$)	Objaverse, ObjaverseXL
XCube (2024)	occupancy grid	-	watertight mesh	$16^3 \times 16$ + more	occupancy grid	ShapeNet, Objaverse
LT3SD (2025)	UDF grid	-	watertight mesh	$(2\times1\times2)\times (5+4^3\times4 + 16^3\times4)$	UDF grid	3D Front
Diffusion-SDF (2023)	SDF field	-	watertight mesh	768	point cloud ($\mathbf{x}$)	ShapeNet-watertight, YCB
MOSAIC-SDF (2024)	SDF field	-	watertight mesh and optimization	$1024 \times (3+1+7^3)$	-	ShapeNet-watertight, scalable 3D captioning dataset
TripoSG (2025a)	SDF field	-	watertight mesh	2048×64	point cloud ($\mathbf{x}, \hat{\mathbf{n}}$)	Objaverse, ObjaverseXL
Hunyuan3D 2.0 (2025)	SDF field	-*	watertight mesh	3072×64	point cloud ($\mathbf{x}$)	Objaverse, ObjaverseXL, more
Make-A-Shape (2024)	SDF grid	-	watertight mesh	9M	-	18 datasets
3DTopia-XL (2025b)	PrimX (SDF field)	RGB, PBR	PrimX optimization	$2048\times(3+1+4^3) = 139,264$	PrimX	Objaverse
Sparc3D (2025b)	SDF grid	-	watertight mesh, grid optimization	unknown	SDF grid	
Volume Diffusion (2023)	radiance field	diffuse RGB	run inference network	$32^3 \times 4$	multiview images	Objaverse
TRELLIS (2025)	occupancy grid	diffuse 3DGS	multiview DINOv2	$\sim$20,000 $\times$ 11 (64^3 grid)	sparse feature grid	Objaverse, ObjaverseXL, ABO, 3D-future, HSSD
TripoSF (2025)	SDF grid	-	multiview depth and normal	$\sim$183,000 $\times$ 11 (256^3 grid)	point cloud ($\mathbf{x}, \hat{\mathbf{n}}$)	Objaverse, ObjaverseXL

appearance modeling performance, *e.g.*, row 1-2 *vs*. 1-3 (or row 2-2 *vs*. 2-3, 3-2 *vs*. 3-3). We hypothesize that this is because zero-degree spherical harmonics cannot capture view-dependent effects, which then becomes a bottleneck, preventing the model from fully leveraging the information contained in the view directions. To verify, we ablate by removing the ray information from our encoder when using 3-degree spherical harmonics. The improvement in row 1-6, which incorporates ray information, from 1-7 (or row 2-6 *vs*. 2-7, 3-6 *vs*. 3-7) corroborates our hypothesis.

C.2 Ablations on Number of Input Views in Inference

We are interested in understanding to what extent our approach is robust to the discrepancies between the number of input views during training and inference. We provide quantitative evaluations in Tab. S5.

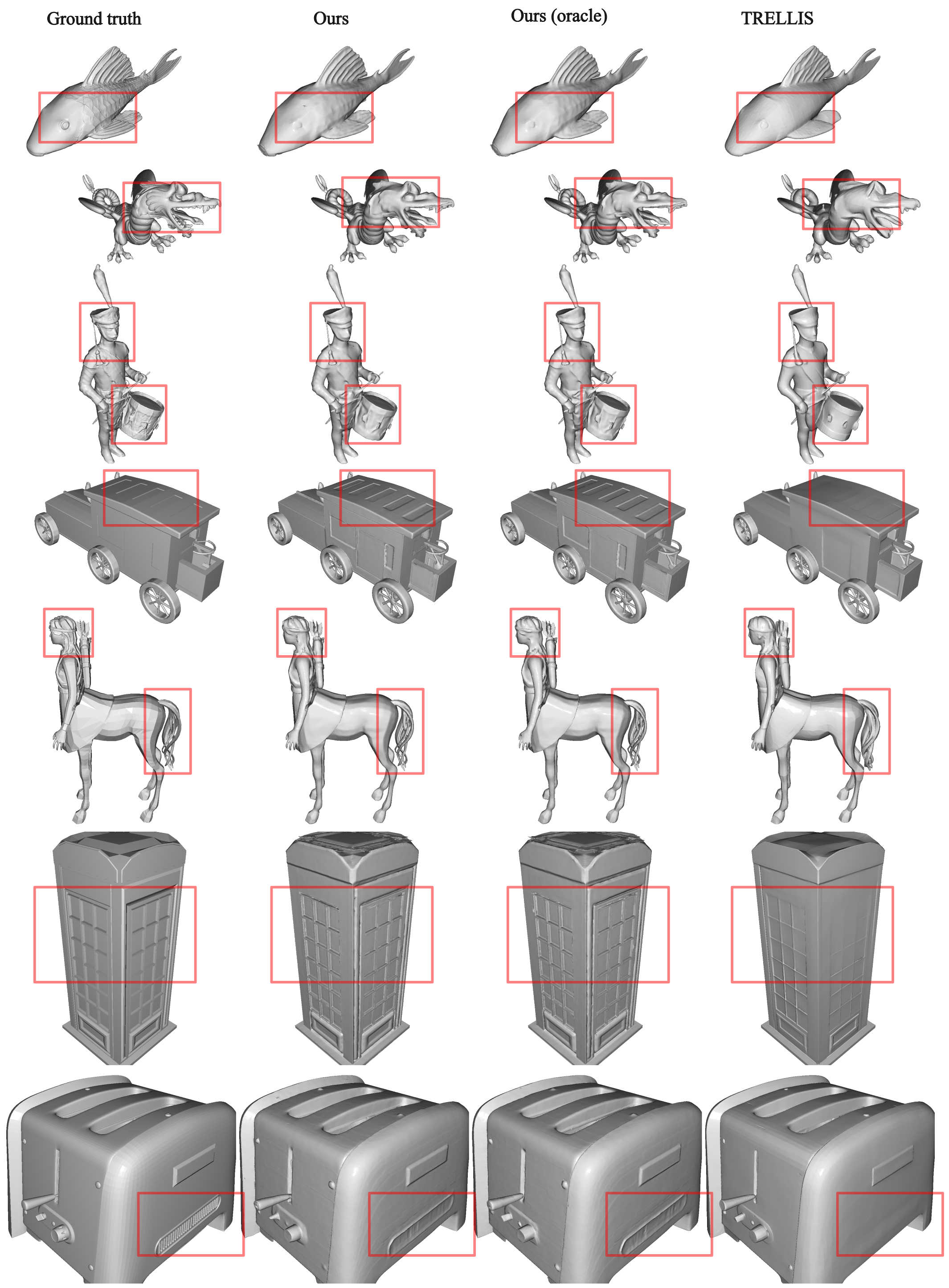


Figure S1: **Mesh comparisons.** We demonstrate the qualities of our mesh decoder results to TRELLIS. As highlighted, our produced mesh maintains more details. Mesh credit: 1812panorama (2019); alzarac (2019); AdamJonesCGD (2020); a108082046 (2022); nastasyas (2019); Alienor.org (2016); GJ2012 (2013).

D Comprehensive Generation Results

As demonstrated in Fig. 6, we are interested in aligning the generation with the input view faithfully. To achieve this, for each sample used during training, we carefully rotate the world coordinate system such that the input view's corresponding camera poses are at the identity orientation. This relieves the model from the burden of inferring the orientation of 3D space during training. Further, we consider utilizing the view direction during the generative model training as well to enable the model be aware of 3D orientation. Since we make the orientation identity, ray information essentially means the availability of camera intrinsics. Then, during inference, we use an off-the-shelf intrinsic estimator (Wang et al., 2025) to obtain the intrinsics. However, as shown in row 3 *vs*. 2 in Tab. S6, it seems like the intrinsic information is unnecessary. Thus we use the generative model trained without any ray information to report our qualitative and quantitative results in the paper.

Table S2: **Reconstruction on Toys4k.** For 3D assets, we adapt inputs per model. TRELLIS (Xiang et al., 2025) takes the ground-truth mesh and 150 sphere-distributed renderings. Ours uses RGB-D images from 150 evenly distributed views. For appearance evaluation, we render each model's output from 100 random cameras, varying difficulty by adjusting camera radius. Each model is further evaluated under three distinct lighting conditions. Importantly, no separate models are trained; all evaluations are conducted on the same model. As a result, we conduct evaluations at the scale of over 3000 (objects) × 100 (views) × 2 (difficulties) × 3 (lightings) ≈ **1.8 million images**. We report in the format of `mean±std`, where the standard deviation is computed across objects. Note, row 1–9 have the same appearance metrics as row 1–8. The same applies to rows 2–9 and 3–9.

	Method	SH Deg	Enc Ray	Pred Occ	Mesh	Simple, Camera Radius [3, 4]			Hard, Camera Radius [1, 3]			CD (100k)↓
						PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	
						Uniform Lighting						
1-1	TRELLIS	0	✗	–	✓	28.17±4.09	0.970±0.024	0.039±0.024	24.63±4.01	0.934±0.054	0.098±0.059	0.646±0.457
1-2	Ours	0	✗	✗	✗	34.40±3.62	0.984±0.017	0.025±0.019	32.19±3.95	0.965±0.042	0.059±0.047	0.677±0.378
1-3	Ours	0	✓	✗	✗	34.44±3.47	0.984±0.017	0.026±0.020	32.18±3.79	0.964±0.042	0.060±0.048	0.678±0.382
1-4	Ours	1	✓	✗	✗	35.12±3.39	0.986±0.015	0.023±0.017	33.17±3.76	0.968±0.040	0.054±0.044	0.647±0.393
1-5	Ours	2	✓	✗	✗	35.32±3.45	0.986±0.016	0.023±0.017	33.29±3.80	0.969±0.040	0.055±0.044	0.658±0.389
1-6	Ours	3	✓	✗	✗	35.32±3.38	0.986±0.015	0.022±0.017	33.39±3.73	0.969±0.039	0.053±0.044	0.663±0.383
1-7	Ours	3	✗	✗	✗	35.54±3.63	0.986±0.015	0.023±0.017	33.37±3.97	0.969±0.040	0.055±0.044	0.672±0.370
1-8	Ours	3	✓	✓	✗	35.27±3.36	0.986±0.015	0.022±0.017	33.38±3.71	0.969±0.040	0.052±0.044	0.662±0.384
1-9	Ours	3	✓	✓	✓	35.27±3.36	0.986±0.015	0.022±0.017	33.38±3.71	0.969±0.040	0.052±0.044	0.532±0.683
1-10	Oracle	3	–	–	✓	35.26±3.34	0.986±0.015	0.022±0.017	33.42±3.69	0.970±0.039	0.051±0.043	0.513±0.639
						TRELLIS Lighting						
2-1	TRELLIS	0	✗	–	✓	31.12±3.39	0.974±0.022	0.034±0.022	27.57±3.38	0.941±0.050	0.090±0.055	0.608±0.437
2-2	Ours	0	✗	✗	✗	32.47±3.83	0.980±0.020	0.029±0.022	30.21±4.19	0.958±0.046	0.067±0.053	0.675±0.379
2-3	Ours	0	✓	✗	✗	32.47±3.69	0.980±0.020	0.029±0.022	30.21±4.06	0.957±0.046	0.068±0.052	0.677±0.382
2-4	Ours	1	✓	✗	✗	34.00±3.38	0.984±0.016	0.025±0.019	32.03±3.74	0.965±0.040	0.059±0.047	0.648±0.391
2-5	Ours	2	✓	✗	✗	34.06±3.40	0.984±0.016	0.024±0.019	32.12±3.79	0.966±0.041	0.058±0.047	0.655±0.389
2-6	Ours	3	✓	✗	✗	34.19±3.39	0.985±0.016	0.024±0.019	32.36±3.77	0.967±0.040	0.056±0.046	0.668±0.385
2-7	Ours	3	✗	✗	✗	34.16±3.68	0.985±0.017	0.025±0.019	32.11±4.04	0.966±0.041	0.058±0.047	0.669±0.371
2-8	Ours	3	✓	✓	✗	34.16±3.39	0.985±0.016	0.023±0.018	32.36±3.77	0.967±0.040	0.055±0.046	0.668±0.384
2-9	Ours	3	✓	✓	✓	34.16±3.39	0.985±0.016	0.023±0.018	32.36±3.77	0.967±0.040	0.055±0.046	0.524±0.484
2-10	Oracle	3	–	–	✓	34.14±3.37	0.985±0.016	0.023±0.018	32.38±3.74	0.967±0.040	0.054±0.045	0.506±0.439
						Random Lighting						
3-1	TRELLIS	0	✗	–	✓	27.94±3.77	0.966±0.025	0.038±0.024	24.37±3.66	0.927±0.054	0.098±0.058	0.631±0.449
3-2	Ours	0	✗	✗	✗	32.12±3.23	0.981±0.018	0.026±0.021	30.08±3.67	0.961±0.043	0.062±0.051	0.675±0.381
3-3	Ours	0	✓	✗	✗	32.18±3.12	0.981±0.019	0.026±0.021	30.11±3.57	0.960±0.044	0.063±0.052	0.680±0.383
3-4	Ours	1	✓	✗	✗	33.02±2.92	0.984±0.017	0.023±0.019	31.20±3.39	0.965±0.041	0.057±0.047	0.652±0.395
3-5	Ours	2	✓	✗	✗	33.13±2.99	0.984±0.017	0.023±0.019	31.34±3.49	0.966±0.041	0.058±0.048	0.664±0.393
3-6	Ours	3	✓	✗	✗	33.22±2.95	0.984±0.017	0.023±0.019	31.50±3.41	0.966±0.041	0.056±0.048	0.669±0.387
3-7	Ours	3	✗	✗	✗	33.23±3.32	0.984±0.017	0.024±0.019	31.30±3.83	0.965±0.041	0.058±0.049	0.675±0.372
3-8	Ours	3	✓	✓	✗	33.18±2.93	0.984±0.017	0.022±0.019	31.49±3.39	0.966±0.041	0.055±0.048	0.669±0.387
3-9	Ours	3	✓	✓	✓	33.18±2.93	0.984±0.017	0.022±0.019	31.49±3.39	0.966±0.041	0.055±0.048	0.541±0.819
3-10	Oracle	3	–	–	✓	33.15±2.90	0.984±0.016	0.022±0.019	31.50±3.36	0.967±0.040	0.054±0.047	0.517±0.694

Table S3: **Reconstruction on GSO.** For 3D assets, we adapt inputs per model. TRELLIS (Xiang et al., 2025) takes the ground-truth mesh and 150 sphere-distributed renderings. Ours uses RGB-D images from 150 evenly distributed views. For appearance evaluation, we render each model's output from 100 random cameras, varying difficulty by adjusting camera radius. Each model is further evaluated under three distinct lighting conditions. Importantly, no separate models are trained; all evaluations are conducted on the same model. As a result, we conduct evaluations at the scale of over 1000 (objects) $\times$ 100 (views) $\times$ 2 (difficulties) $\times$ 3 (lightings) $\approx$ **600 thousand images**. We report in the format of `mean`$\pm$`std`, where the standard deviation is computed across objects. Note, row 1–9 have the same appearance metrics as row 1–8. The same applies to rows 2–9 and 3–9.

	Method	SH Deg	Enc Ray	Pred Occ	Mesh	Simple, Camera Radius [3, 4]			Hard, Camera Radius [1, 3]			CD (100k)↓
						PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	
						Uniform Lighting						
1-1	TRELLIS	0	✗	–	✓	27.34 ± 3.82	0.947 ± 0.036	0.053 ± 0.029	23.72 ± 3.66	0.883 ± 0.068	0.139 ± 0.065	0.774 ± 0.337
1-2	Ours	0	✗	✗	✗	34.27 ± 3.25	0.975 ± 0.022	0.034 ± 0.025	31.39 ± 3.61	0.937 ± 0.046	0.093 ± 0.055	0.837 ± 0.376
1-3	Ours	0	✓	✗	✗	34.04 ± 3.23	0.974 ± 0.022	0.034 ± 0.025	31.15 ± 3.59	0.935 ± 0.048	0.093 ± 0.055	0.842 ± 0.386
1-4	Ours	1	✓	✗	✗	34.55 ± 3.18	0.976 ± 0.021	0.031 ± 0.023	31.75 ± 3.60	0.939 ± 0.045	0.087 ± 0.053	0.809 ± 0.441
1-5	Ours	2	✓	✗	✗	34.62 ± 3.24	0.976 ± 0.021	0.031 ± 0.024	31.77 ± 3.65	0.939 ± 0.046	0.087 ± 0.053	0.803 ± 0.397
1-6	Ours	3	✓	✗	✗	34.69 ± 3.22	0.976 ± 0.021	0.031 ± 0.024	31.88 ± 3.65	0.940 ± 0.046	0.086 ± 0.053	0.819 ± 0.398
1-7	Ours	3	✗	✗	✗	34.93 ± 3.24	0.977 ± 0.020	0.031 ± 0.023	32.00 ± 3.63	0.942 ± 0.044	0.087 ± 0.053	0.841 ± 0.389
1-8	Ours	3	✓	✓	✗	34.67 ± 3.21	0.976 ± 0.021	0.031 ± 0.024	31.88 ± 3.65	0.940 ± 0.046	0.086 ± 0.053	0.819 ± 0.396
1-9	Ours	3	✓	✓	✓	34.67 ± 3.21	0.976 ± 0.021	0.031 ± 0.024	31.88 ± 3.65	0.940 ± 0.046	0.086 ± 0.053	0.631 ± 0.355
1-10	Oracle	3	–	–	✓	34.66 ± 3.20	0.976 ± 0.021	0.030 ± 0.023	31.92 ± 3.65	0.941 ± 0.045	0.085 ± 0.053	0.621 ± 0.342
						TRELLIS Lighting						
2-1	TRELLIS	0	✗	–	✓	30.81 ± 2.67	0.958 ± 0.028	0.047 ± 0.026	27.21 ± 2.56	0.907 ± 0.055	0.126 ± 0.058	0.737 ± 0.331
2-2	Ours	0	✗	✗	✗	33.99 ± 2.54	0.978 ± 0.017	0.033 ± 0.023	31.65 ± 2.71	0.948 ± 0.036	0.089 ± 0.052	0.832 ± 0.370
2-3	Ours	0	✓	✗	✗	33.71 ± 2.43	0.978 ± 0.018	0.033 ± 0.024	31.40 ± 2.62	0.947 ± 0.037	0.088 ± 0.051	0.838 ± 0.381
2-4	Ours	1	✓	✗	✗	34.75 ± 2.60	0.980 ± 0.016	0.030 ± 0.022	32.50 ± 2.87	0.952 ± 0.035	0.080 ± 0.048	0.803 ± 0.437
2-5	Ours	2	✓	✗	✗	34.87 ± 2.68	0.980 ± 0.017	0.030 ± 0.022	32.58 ± 2.95	0.952 ± 0.036	0.081 ± 0.049	0.795 ± 0.390
2-6	Ours	3	✓	✗	✗	34.91 ± 2.65	0.980 ± 0.016	0.029 ± 0.022	32.67 ± 2.95	0.952 ± 0.036	0.080 ± 0.049	0.816 ± 0.396
2-7	Ours	3	✗	✗	✗	35.19 ± 2.72	0.981 ± 0.016	0.030 ± 0.022	32.79 ± 2.97	0.953 ± 0.034	0.081 ± 0.049	0.834 ± 0.384
2-8	Ours	3	✓	✓	✗	34.89 ± 2.64	0.980 ± 0.016	0.029 ± 0.022	32.68 ± 2.94	0.952 ± 0.036	0.079 ± 0.049	0.815 ± 0.392
2-9	Ours	3	✓	✓	✓	34.89 ± 2.64	0.980 ± 0.016	0.029 ± 0.022	32.68 ± 2.94	0.952 ± 0.036	0.079 ± 0.049	0.665 ± 0.355
2-10	Oracle	3	–	–	✓	34.87 ± 2.63	0.981 ± 0.016	0.029 ± 0.021	32.70 ± 2.94	0.953 ± 0.036	0.078 ± 0.048	0.654 ± 0.338
						Random Lighting						
3-1	TRELLIS	0	✗	–	✓	27.66 ± 3.26	0.948 ± 0.033	0.050 ± 0.028	24.11 ± 3.08	0.886 ± 0.064	0.133 ± 0.062	0.767 ± 0.349
3-2	Ours	0	✗	✗	✗	33.09 ± 2.47	0.977 ± 0.018	0.031 ± 0.023	30.97 ± 2.81	0.945 ± 0.039	0.086 ± 0.052	0.831 ± 0.378
3-3	Ours	0	✓	✗	✗	32.97 ± 2.40	0.976 ± 0.018	0.031 ± 0.023	30.82 ± 2.77	0.943 ± 0.040	0.087 ± 0.052	0.835 ± 0.381
3-4	Ours	1	✓	✗	✗	33.46 ± 2.41	0.978 ± 0.017	0.028 ± 0.021	31.41 ± 2.81	0.947 ± 0.038	0.080 ± 0.049	0.810 ± 0.444
3-5	Ours	2	✓	✗	✗	33.61 ± 2.47	0.978 ± 0.017	0.029 ± 0.022	31.55 ± 2.88	0.947 ± 0.038	0.081 ± 0.049	0.802 ± 0.399
3-6	Ours	3	✓	✗	✗	33.67 ± 2.46	0.979 ± 0.017	0.028 ± 0.022	31.65 ± 2.89	0.948 ± 0.038	0.080 ± 0.050	0.818 ± 0.397
3-7	Ours	3	✗	✗	✗	33.98 ± 2.53	0.980 ± 0.016	0.028 ± 0.021	31.84 ± 2.93	0.949 ± 0.036	0.081 ± 0.049	0.840 ± 0.390
3-8	Ours	3	✓	✓	✗	33.64 ± 2.43	0.979 ± 0.017	0.028 ± 0.022	31.64 ± 2.87	0.948 ± 0.038	0.080 ± 0.050	0.818 ± 0.397
3-9	Ours	3	✓	✓	✓	33.64 ± 2.43	0.979 ± 0.017	0.028 ± 0.022	31.64 ± 2.87	0.948 ± 0.038	0.080 ± 0.050	0.631 ± 0.356
3-10	Oracle	3	–	–	✓	33.61 ± 2.42	0.979 ± 0.017	0.028 ± 0.021	31.65 ± 2.86	0.949 ± 0.038	0.079 ± 0.049	0.621 ± 0.342

D.1 Ablations on ODE Numerical Integration

We study the effect of ODE numerical integration used when sampling from our generative model. Specifically, we ablate the algorithms (Euler and Heun), the step size (or equivalently the number of steps) used during the numerical integration, and the numerical precision of the model (`float32` and `bfloat16`) during sampling. We provide quantitative results in Sec. S7. The results suggest our generative model is robust to numerical integration — we observe small change in performance when switching from the second-order method Heun with 100 steps using `float32` (conditioning view FID = 6.6), to a relatively cheaper first-order Euler with 25 steps using `bfloat16` (conditioning view FID = 6.7).

Table S4: **Reconstruction on PBR-Objaverse.** For 3D assets, we adapt inputs per model. TRELLIS (Xiang et al., 2025) takes the ground-truth mesh and 150 sphere-distributed renderings. Ours uses RGB-D images from 150 evenly distributed views. For appearance evaluation, we render each model's output from 100 random cameras, varying difficulty by adjusting camera radius. Each model is further evaluated under three distinct lighting conditions. Importantly, no separate models are trained; all evaluations are conducted on the same model. As a result, we conduct evaluations at the scale of 200 (objects) $\times$ 100 (views) $\times$ 2 (difficulties) $\times$ 3 (lightings) $\approx$ **120 thousand images**. We report in the format of `mean`$\pm_{\texttt{std}}$, where the standard deviation is computed across objects. Note, row 1–9 have the same appearance metrics as row 1–8, so we can ignore them. The same applies to rows 2–9 and 3–9.

	Method	SH Deg	Enc Ray	Pred Occ	Mesh	Simple, Camera Radius [3, 4]			Hard, Camera Radius [1, 3]			CD (100k)↓
						PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	
						Uniform Lighting						
1-1	TRELLIS	0	✗	–	✓	28.63±3.09	0.955±0.028	0.046±0.025	25.06±2.93	0.902±0.057	0.121±0.062	0.657±0.305
1-2	Ours	0	✗	✗	✗	32.95±2.87	0.974±0.018	0.033±0.020	30.07±3.02	0.939±0.042	0.087±0.051	0.727±0.323
1-3	Ours	0	✓	✗	✗	33.14±2.68	0.974±0.018	0.034±0.021	30.21±2.85	0.937±0.042	0.089±0.053	0.724±0.325
1-4	Ours	1	✓	✗	✗	34.35±2.37	0.978±0.016	0.028±0.018	31.67±2.67	0.947±0.038	0.076±0.046	0.702±0.338
1-5	Ours	2	✓	✗	✗	34.47±2.45	0.978±0.016	0.028±0.018	31.74±2.73	0.947±0.039	0.077±0.047	0.709±0.334
1-6	Ours	3	✓	✗	✗	34.62±2.33	0.979±0.016	0.028±0.018	31.98±2.64	0.948±0.039	0.075±0.047	0.705±0.327
1-7	Ours	3	✗	✗	✗	34.66±2.62	0.979±0.016	0.029±0.018	31.83±2.89	0.948±0.039	0.077±0.047	0.730±0.320
1-8	Ours	3	✓	✓	✗	34.63±2.33	0.979±0.016	0.027±0.017	32.01±2.64	0.948±0.039	0.074±0.047	0.704±0.325
1-9	Ours	3	✓	✓	✓	34.63±2.33	0.979±0.016	0.027±0.017	32.01±2.64	0.948±0.039	0.074±0.047	0.556±0.341
1-10	Oracle	3	–	–	✓	34.64±2.31	0.979±0.016	0.027±0.017	32.07±2.62	0.949±0.038	0.074±0.046	0.541±0.315
						TRELLIS Lighting						
2-1	TRELLIS	0	✗	–	✓	29.69±2.59	0.958±0.025	0.044±0.023	26.03±2.50	0.904±0.053	0.118±0.058	0.616±0.277
2-2	Ours	0	✗	✗	✗	30.35±3.01	0.965±0.023	0.039±0.023	27.39±3.18	0.921±0.049	0.102±0.056	0.737±0.316
2-3	Ours	0	✓	✗	✗	30.37±3.04	0.965±0.023	0.040±0.023	27.41±3.21	0.919±0.050	0.102±0.056	0.735±0.320
2-4	Ours	1	✓	✗	✗	32.52±2.45	0.975±0.017	0.031±0.019	29.87±2.70	0.939±0.042	0.084±0.049	0.720±0.342
2-5	Ours	2	✓	✗	✗	32.47±2.45	0.975±0.018	0.031±0.019	29.90±2.73	0.940±0.042	0.083±0.049	0.715±0.323
2-6	Ours	3	✓	✗	✗	32.63±2.38	0.976±0.017	0.030±0.018	30.14±2.69	0.941±0.042	0.081±0.049	0.720±0.321
2-7	Ours	3	✗	✗	✗	32.56±2.72	0.975±0.018	0.031±0.019	29.89±2.97	0.939±0.042	0.084±0.049	0.731±0.317
2-8	Ours	3	✓	✓	✗	32.63±2.37	0.976±0.017	0.030±0.018	30.16±2.69	0.942±0.042	0.080±0.049	0.724±0.323
2-9	Ours	3	✓	✓	✓	32.63±2.37	0.976±0.017	0.030±0.018	30.16±2.69	0.942±0.042	0.080±0.049	0.569±0.332
2-10	Oracle	3	–	–	✓	32.61±2.37	0.976±0.017	0.029±0.018	30.20±2.69	0.942±0.042	0.080±0.048	0.558±0.316
						Random Lighting						
3-1	TRELLIS	0	✗	–	✓	26.29±3.56	0.939±0.038	0.052±0.030	22.74±3.37	0.869±0.075	0.134±0.070	0.691±0.365
3-2	Ours	0	✗	✗	✗	28.58±3.65	0.957±0.031	0.043±0.028	25.66±3.87	0.904±0.066	0.107±0.065	0.726±0.322
3-3	Ours	0	✓	✗	✗	28.88±3.61	0.956±0.032	0.043±0.028	25.93±3.81	0.903±0.067	0.109±0.066	0.732±0.331
3-4	Ours	1	✓	✗	✗	30.36±3.15	0.965±0.027	0.036±0.024	27.60±3.43	0.920±0.059	0.095±0.058	0.708±0.338
3-5	Ours	2	✓	✗	✗	30.39±3.08	0.965±0.027	0.036±0.024	27.65±3.39	0.920±0.060	0.095±0.059	0.717±0.335
3-6	Ours	3	✓	✗	✗	30.59±3.08	0.966±0.027	0.036±0.024	27.92±3.42	0.922±0.059	0.093±0.059	0.713±0.327
3-7	Ours	3	✗	✗	✗	30.11±3.48	0.964±0.027	0.037±0.024	27.27±3.75	0.917±0.060	0.096±0.059	0.729±0.325
3-8	Ours	3	✓	✓	✗	30.59±3.09	0.966±0.027	0.035±0.024	27.94±3.43	0.922±0.060	0.092±0.059	0.713±0.328
3-9	Ours	3	✓	✓	✓	30.59±3.09	0.966±0.027	0.035±0.024	27.94±3.43	0.922±0.060	0.092±0.059	0.552±0.329
3-10	Oracle	3	–	–	✓	30.59±3.07	0.966±0.026	0.035±0.024	27.97±3.42	0.922±0.059	0.092±0.058	0.541±0.312

D.2 Runtime and Memory Analysis

We analyze the runtime for both TRELLIS and our generative models in Tab. S8. Our model's latent sampling costs 9.3 seconds on while all decoders' feedforward passes cost less than 100 milliseconds on a single NVIDIA H100 80GB HBM3 GPU. In comparison, for TRELLIS, sampling SLAT (both coarse voxel and feature) takes 11.8 seconds. Utilizing one-step flow-matching models like MeanFlow (Geng et al., 2025) can further improve the speed of our generative model and is left as future work.

Table S5: **Ablation on number of input views for reconstruction during inference.** We choose TRELLIS lighting setup on Toys4k dataset. Our model is the same as "ours" in Tab. 1. Both TRELLIS and ours are trained with 150 views. For appearance evaluation, we render each model's output from 100 random cameras, varying difficulty by adjusting camera radius. We report in the format of `mean`$\pm$`std`, where the standard deviation is computed across objects. Note, we re-render the evaluation data for this ablation, thus row 1 (row 2) differs slightly from row 2-1 (row 2-9) in Tab. S2.

	Method	Simple, Camera Radius [3, 4]			Hard, Camera Radius [1, 3]			CD (100k)↓
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	
		150 input views						
1	TRELLIS	31.559±3.509	0.9740±0.0224	0.0361±0.0217	27.948±3.539	0.9408±0.0508	0.0928±0.0539	0.589±0.432
2	Ours	33.909±3.157	0.9841±0.0162	0.0260±0.0189	32.073±3.521	0.9658±0.0403	0.0585±0.0458	0.525±0.502
		120 input views						
3	TRELLIS	31.518±3.509	0.9738±0.0225	0.0363±0.0218	27.912±3.541	0.9404±0.0510	0.0932±0.0541	0.592±0.431
4	Ours	33.908±3.158	0.9841±0.0162	0.0260±0.0188	32.072±3.522	0.9658±0.0403	0.0585±0.0457	0.524±0.486
		90 input views						
5	TRELLIS	31.431±3.506	0.9734±0.0227	0.0366±0.0221	27.833±3.540	0.9397±0.0514	0.0938±0.0545	0.594±0.425
6	Ours	33.910±3.157	0.9841±0.0162	0.0260±0.0189	32.074±3.522	0.9658±0.0403	0.0585±0.0457	0.524±0.492
		60 input views						
7	TRELLIS	31.270±3.496	0.9726±0.0231	0.0372±0.0224	27.688±3.533	0.9383±0.0520	0.0952±0.0552	0.603±0.430
8	Ours	33.909±3.155	0.9841±0.0162	0.0260±0.0188	32.073±3.519	0.9658±0.0403	0.0585±0.0457	0.525±0.491
		30 input views						
9	TRELLIS	30.692±3.441	0.9699±0.0244	0.0396±0.0238	27.159±3.484	0.9336±0.0541	0.1002±0.0576	0.637±0.443
10	Ours	33.908±3.157	0.9841±0.0162	0.0260±0.0188	32.072±3.521	0.9658±0.0403	0.0585±0.0457	0.527±0.503

Table S6: **Single-image-conditioned generation on Toys4k with TRELLIS lighting.** KID is reported by $\times$100. CFG scale is 3.0. The best is highlighted.

	Method	Train w/ Ray	Infer w/ GT Ray	Train Iters	CLIP↑	Conditioning View				Novel View			
						FID↓	KID↓	FID$_{\text{dino}}$↓	KID$_{\text{dino}}$↓	FID↓	KID↓	FID$_{\text{dino}}$↓	KID$_{\text{dino}}$↓
1	TRELLIS	✗	-	400k	0.899±0.045	12.84	0.088	84.692	2.311	7.600	0.100	67.458	3.166
2-1	Ours	✗	-	280k	0.906±0.040	8.193	0.012	48.117	0.461	6.648	0.064	75.814	4.321
2-2	Ours	✗	-	400k	0.906±0.041	7.741	0.010	44.555	0.392	6.413	0.064	71.436	3.997
2-3	Ours	✗	-	600k	0.905±0.041	6.219	0.009	41.621	1.333	6.216	0.058	66.530	3.522
3	Ours	✓	✗	290k	0.900±0.040	10.78	0.066	65.644	2.281	8.076	0.101	92.915	6.698
4	Ours	✓	✓	290k	0.904±0.039	10.13	0.053	61.342	1.665	7.831	0.097	86.091	5.826

Table S7: **Ablation on DiT sampler for single-image-conditioned generation.** The experiments are conducted on Toys4k with TRELLIS lighting. The generative model is trained for 600k iterations. Note, row 1 is copied from "ours" in Tab. 3 . KID is reported by $\times$100. CFG scale is 3.0. Our generative model's performance is robust across various numbers of sampling steps and numerical integration algorithms.

	Occ Pred	Data Type	Method	Step	CLIP↑	Conditioning View				Novel View			
						FID↓	KID↓	FID$_{\text{dino}}$↓	KID$_{\text{dino}}$↓	FID↓	KID↓	FID$_{\text{dino}}$↓	KID$_{\text{dino}}$↓
1	✗	float32	Heun	100	0.905±0.041	6.219	0.009	41.621	1.333	6.216	0.058	66.530	3.522
2	✓	float32	Heun	100	0.905±0.041	6.622	0.021	42.197	1.391	6.270	0.064	66.699	3.534
3	✓	bfloat16	Heun	100	0.905±0.041	6.661	0.020	43.992	1.741	6.270	0.063	68.025	3.906
4	✓	bfloat16	Heun	50	0.905±0.041	6.659	0.020	45.533	2.105	6.266	0.062	68.319	4.185
5	✓	bfloat16	Heun	25	0.904±0.041	6.644	0.019	54.231	4.011	6.251	0.060	77.148	5.879
6	✓	bfloat16	Euler	100	0.906±0.041	6.656	0.022	42.472	1.476	6.365	0.066	67.856	3.848
7	✓	bfloat16	Euler	50	0.905±0.041	6.688	0.023	42.363	1.430	6.384	0.066	68.987	3.958
8	✓	bfloat16	Euler	25	0.905±0.041	6.733	0.025	43.034	1.280	6.833	0.074	75.687	4.484

Table S8: **Generative model runtime analysis.** All results are reported with `torch.profiler` across three runs. TRELLIS uses 50 Euler steps for both its sparse structure and structured latent generations. We use 50 Euler steps for generating the latents, corresponding to row 7 in Tab. S7.

	Cond Proc (ms)	Structure Gen (s)	Latent Gen (s)	Occ Pred (ms)	3DGS Dec (ms)	Mesh Dec (ms)	Total (s)	Memory (GB)
NVIDIA A100-SXM4-80GB								
TRELLIS	68.90±0.49	4.89±0.80	7.720±5.10	–	18.70±5.46	67.33±13.98	12.76	12.70
Ours	68.78±0.41	–	17.32±1.50	36.07±3.67	35.32±14.2	90.78±29.75	17.55	15.95
NVIDIA H100 80GB HBM3								
TRELLIS	31.01±0.55	3.95±1.17	7.868±4.06	–	15.03±6.19	46.81±13.31	11.91	12.69
Ours	22.58±10.3	–	9.266±0.38	27.16±6.87	30.96±14.3	79.15±31.71	9.426	15.93

Table S9: **Original Geometric reconstruction evaluation**. We report Chamfer distances multiplied by 10^4 for readability, computed using 100k sampled points each from ground-truth and reconstruction. As 3DTopia-XL (Chen et al., 2025b) and TripoSG (Li et al., 2025a) can be sensitive to input geometry, we also list variants with their worst-performing 10% of objects removed. We separate our tested approaches based on those that require ground-truth coarse geometry for decoding the latent representation, and those that do not utilize this information. Our method outputs the best geometry among the approaches in the latter category, and it is competitive with the techniques in the former. Red highlights the best method in each category.

	Method	Appearance	Latent size	PBR-Objaverse	Toys4k	GSO
0	GT	-	-	0.479±0.247	0.445±0.364	0.531±0.309
Requires coarse geometry oracle:						
1	TripoSF (He et al., 2025)	✗	$\approx$ 244k × 11	0.621±0.546	0.595±0.659	0.697±0.751
2	TRELLIS (Xiang et al., 2025)	✓	$\approx$ 20k × 11	0.639±0.405	0.604±0.486	0.725±0.293
3-1	3DTopia-XL (Chen et al., 2025b)	✓	2048 × 64	77.58±406.5	30.16±238.4	0.664±0.321
3-2	(worst 10% removed)	✓	2048 × 64	3.966±8.884	0.980±1.480	0.596±0.157
Does not utilize coarse geometry oracle:						
4-1	TripoSG (Li et al., 2025a)	✗	2048 × 64	32.19±71.07	36.11±62.42	44.52±87.03
4-2	(worst 10% removed)	✗	2048 × 64	13.84±17.96	19.51±24.92	18.73±28.63
5	Shape Tokens (Chang et al., 2024)	✗	1024 × 16	1.120±0.447	1.061±0.565	1.221±0.433
6	Ours	✓	8192 × 32	0.935±0.328	0.890±0.361	1.007±0.331

E Implementation Details

E.1 Architectures

We provide detailed network architectures in Fig. S2 to S7. These include our encoder (Sec. 3.3) in Fig. S2, velocity decoder and Gaussian decoder (Sec. 3.4) in Fig. S3 and S4, mesh decoder in Fig. S5, occupancy decoder in Fig. S6, and generative model's DiT (Sec. 3.5) in Fig. S7.

E.2 Position Encoding

We have the following position encoding function applied on *each channel* of the input data:

$$\{\sin(u_0), \ldots, \sin(u_{F-1}), \cos(u_0), \ldots, \cos(u_{F-1})\}, \tag{S1}$$

$$\text{where } u_i = x \cdot 2^{\left(M_{\text{min}} + i \cdot \frac{M_{\text{max}} - M_{\text{min}}}{F-1}\right)}, \tag{S2}$$

x is the value at the corresponding channel where the position encoding is applied. We use $F = 32$, $M_{\text{min}} = 0$, $M_{\text{max}} = 12, 8,$ and 8 in position encoding functions for 3D location $\mathbf{x}_i$, viewing direction $\hat{\mathbf{d}}_i$, color $\mathbf{c}_i$ in Eq. (1) respectively. For time step t in flow matching (Eq. (2)), we use $F = 16$, $M_{\text{min}} = \log_2 2\pi$, and $M_{\text{max}} = M_{\text{min}} + F - 1$.

E.3 3D Gaussian Prediction

In Fig. S4, the output position of 3D Gaussian is predicted with respect to a normalized space centered around the occupied voxel's world coordinates, and is then translated to the world coordinate system using the voxel's information. Specifically, we predict 3D Gaussian's position as $\mathbf{x}_{\text{output}} \in [-1, 1]^3$. Assume the corresponding voxel's center is located at $\mathbf{x}_{\text{voxel}} \in \mathbb{R}^3$ in the world coordinate system. The final 3D Gaussian's position in the world coordinate system is computed as $\mathbf{x}_{\text{3DGS}} = \mathbf{x}_{\text{voxel}} + s \cdot \mathbf{x}_{\text{output}}$, where s is a hyperparameter to define the size of the normalized space mentioned above. In our experiments, we set $s = 0.05$. Note, $s = 0.05$ is actually larger than the voxel size we consider. This is intentional as it provides more flexibility, such that the predicted 3D Gaussian can go across the voxel boundaries.

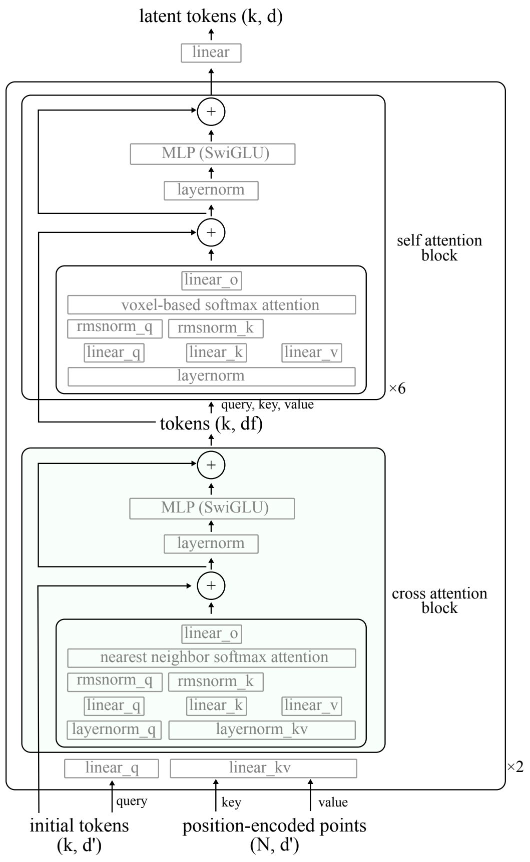
F More Studies

F.1 Studying Spherical Harmonics Degrees

Our Gaussian decoder outputs Gaussians with spherical harmonics up to degree three. We study what information is captured by individual spherical harmonics degrees. In Fig. S8 and Fig. S9, we render the 3D Gaussians from both reconstruction and generation by clipping the degree of the spherical harmonics (*i.e.*, we use only the $\ell \leq 3$ degrees during rendering). We observe that zeroth-degree renderings are mostly view-independent and have little lighting baked in, whereas higher-degree renderings illustrate lighting effects. This is in contrast to TRELLIS's results whose zeroth-degree renderings contain both baked lighting and inaccurate view-dependent appearance produced using micro-surface geometry (Walter et al., 2007). The results suggest that our model is able to represent view-dependent effects using the higher-degree spherical harmonics, and to use the zeroth-degree rendering for view-independent, diffused, appearance. This separation is an interesting finding, and it provides potential opportunity for future investigation of relighting using our representation.

F.2 NeRF Datasets

For RefNeRF dataset's object of car, we visualize the ground-truth mesh as well as unprojected depths from all training views as in Sec. S10. As can be seen, the provided depth maps are not accurate.



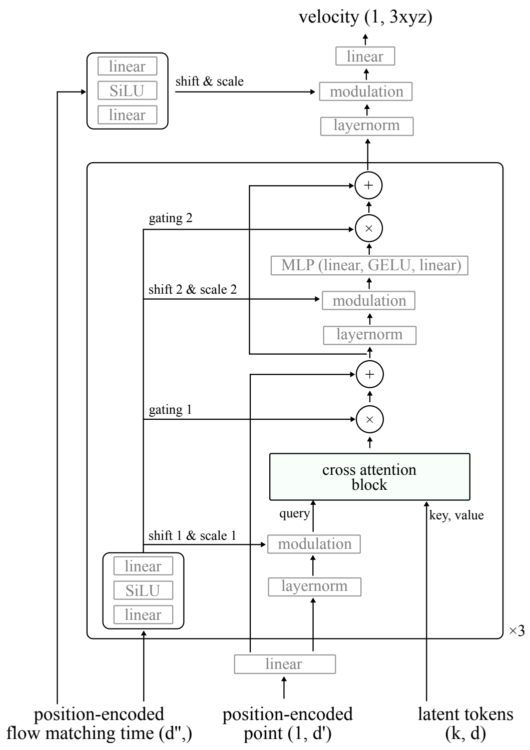


Figure S2: **Encoder architecture.** The model uses a feature dimension of $df = 512$, while the hidden layer in MLP uses a feature dimension of 2048. The number of heads for cross-attention and self-attention is 16. The input dimension $d' = 396$, which includes 3D location, position-encoded 3D location, RGB, position-encoded RGB, and Plucker coordinates. Our latent has $k = 8192$ and $d = 32$. Please refer to Sec. E for position encoding details.

Figure S3: **Velocity decoder architecture.** The model uses a feature dimension of 512, while the hidden layer in MLP uses a feature dimension of 2048. The number of heads for cross-attention is 8. Our latent has $k = 8192$ and $d = 32$. We have $d' = 195$, which includes 3D location and position-encoded 3D location. Meanwhile, $d'' = 64$, which is obtained by applying a linear layer to time-step position encoding in Eq. (S1). Please refer to Sec. E for position encoding details.

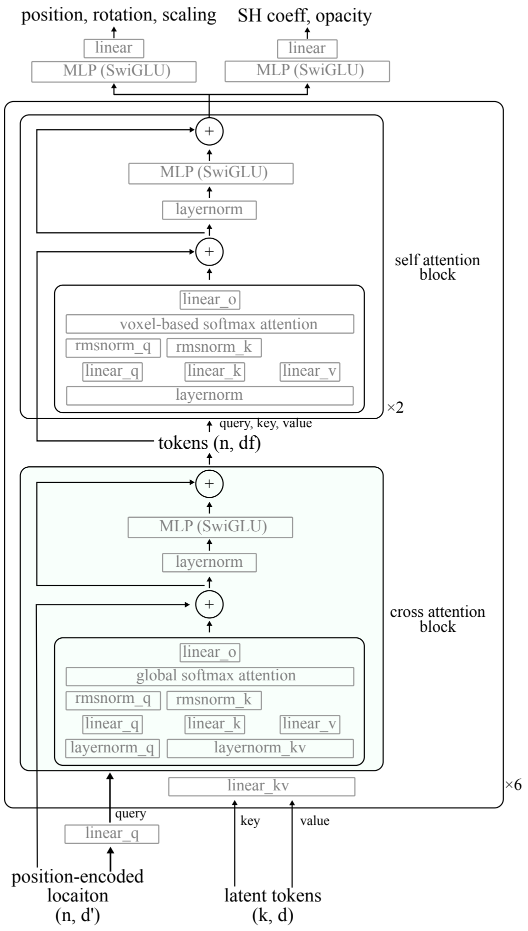


Figure S4: **3D Gaussian decoder architecture.** The model uses a feature dimension of $df = 512$, while the hidden layer in MLP uses a feature dimension of 2048. The number of heads for cross-attention and self-attention is 8. Our latent has $k = 8192$ and $d = 32$. We have $d' = 195$, which includes 3D location and position-encoded 3D location. Please refer to Sec. E for position encoding details.

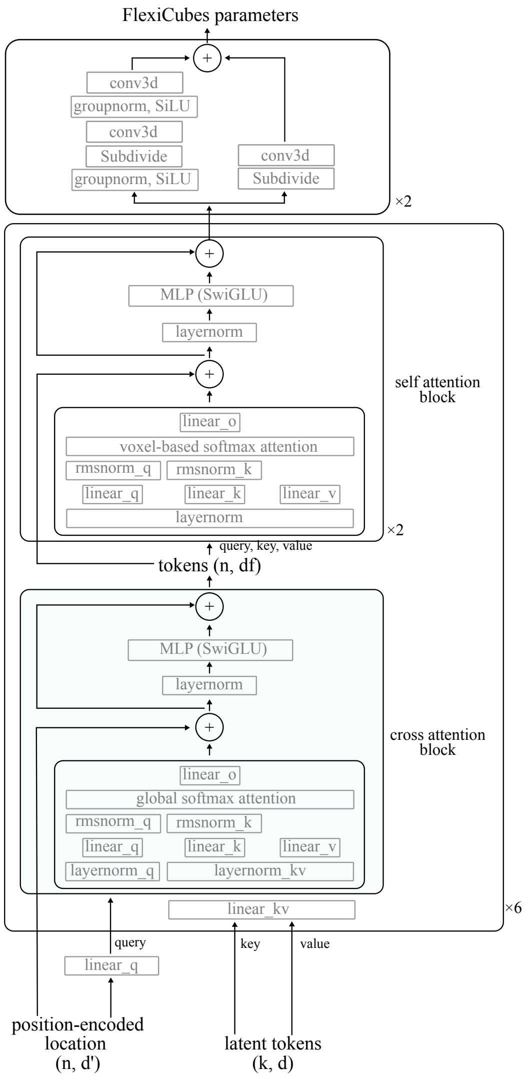
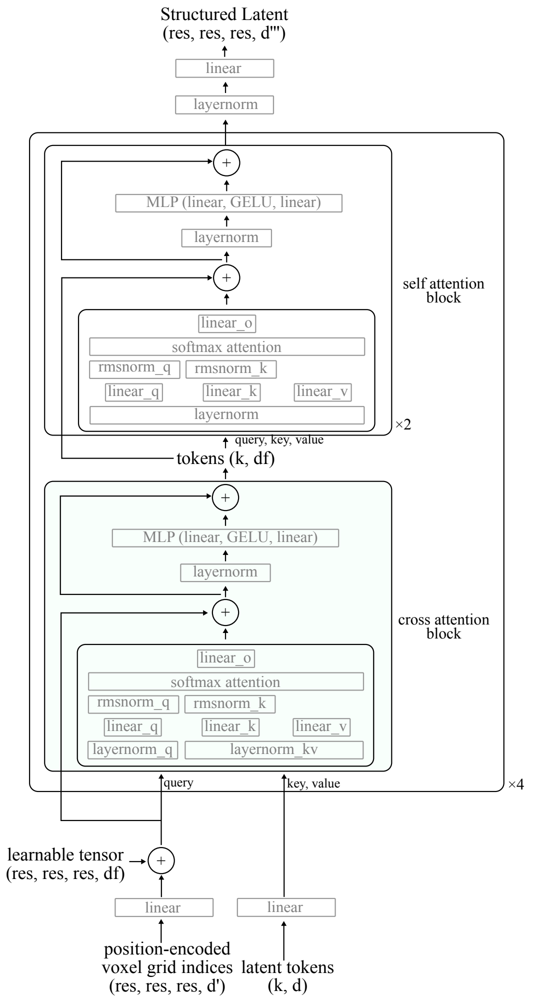


Figure S5: **Mesh decoder architecture.** The model uses a feature dimension of $df = 512$, while the hidden layer in MLP uses a feature dimension of 2048. Our latent has $k = 8192$ and $d = 32$. The number of heads for cross-attention and self-attention is 16. We have $d' = 195$, which includes 3D location and position-encoded 3D location. Please refer to Sec. E for position encoding details.



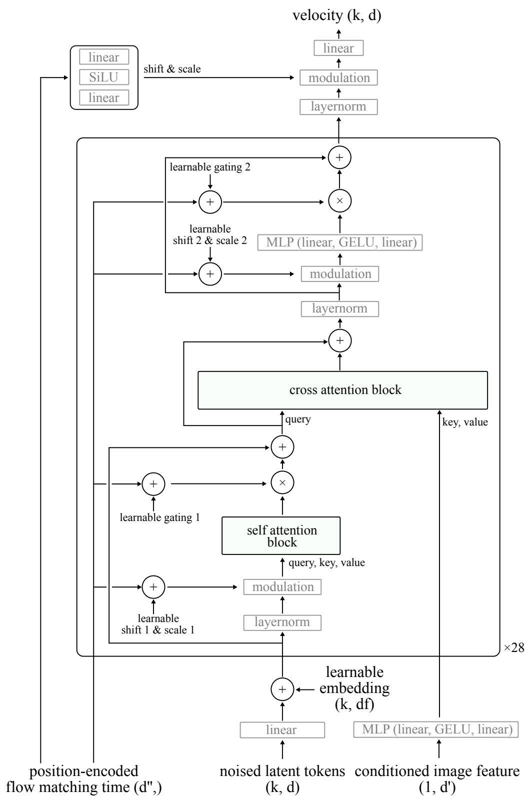

velocity (k, d)
linear
modulation
layernorm
linear
SiLU
linear
shift & scale
learnable gating 2
learnable shift 2 & scale 2
MLP (linear, GELU, linear)
modulation
layernorm
cross attention block
query
key, value
learnable gating 1
self attention block
query, key, value
modulation
layernorm
learnable shift 1 & scale 1
×28
learnable embedding (k, df)
linear
MLP (linear, GELU, linear)
position-encoded flow matching time (d'',)
noised latent tokens (k, d)
conditioned image feature (1, d')

Figure S6: **Occupancy decoder architecture.** The model uses a feature dimension of $df = 512$, while the feature dimension for QKV in cross/self-attention is 1024. The hidden layer in MLP uses a feature dimension of 2048. Our latent has $k = 8192$ and $d = 32$. The number of heads for cross-attention and self-attention is 8. We have $d' = 771$, with $M_{\text{min}} = 0$, $M_{\text{max}} = 5$, and $F = 128$ in Eq. (S1) for encoding 3D location. We use resolution of 16, *i.e.*, res $= 16$. Please refer to Sec. E for position encoding details.

Figure S7: **Generative model DiT architecture.** The model uses a feature dimension of $df = 1152$, while the hidden layer in MLP uses a feature dimension of 4608. Our latent has $k = 8192$ and $d = 32$. The number of heads for self-attention and cross-attention is 16. The feature dimension for conditioning image $d' = 2048$. We have $d'' = 64$, with $M_{\text{min}} = 0$, $M_{\text{max}} = 12$, and $F = 32$ in Eq. (S1) for encoding flow matching time step. Please refer to Sec. E for position encoding details.

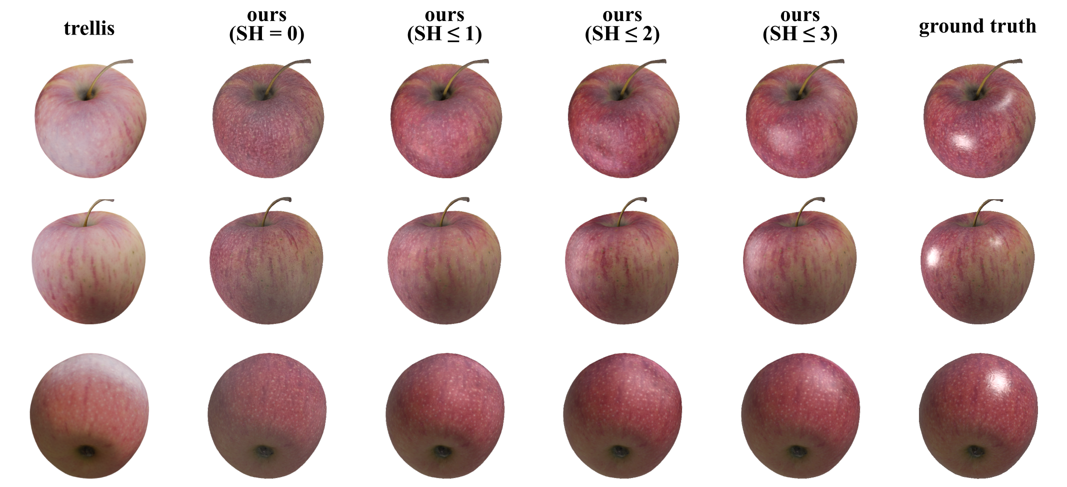


Figure S8: **Rendering with various spherical harmonics degrees in reconstruction.** When restricted to zeroth-order spherical harmonics, our 3D Gaussians produce a view-independent appearance and avoid the over-exposed regions observed in TRELLIS's renderings. As we progressively incorporate higher-order spherical harmonics, our method yields increasingly pronounced view-dependent effects. Mesh credit: DigitalSouls (2019).

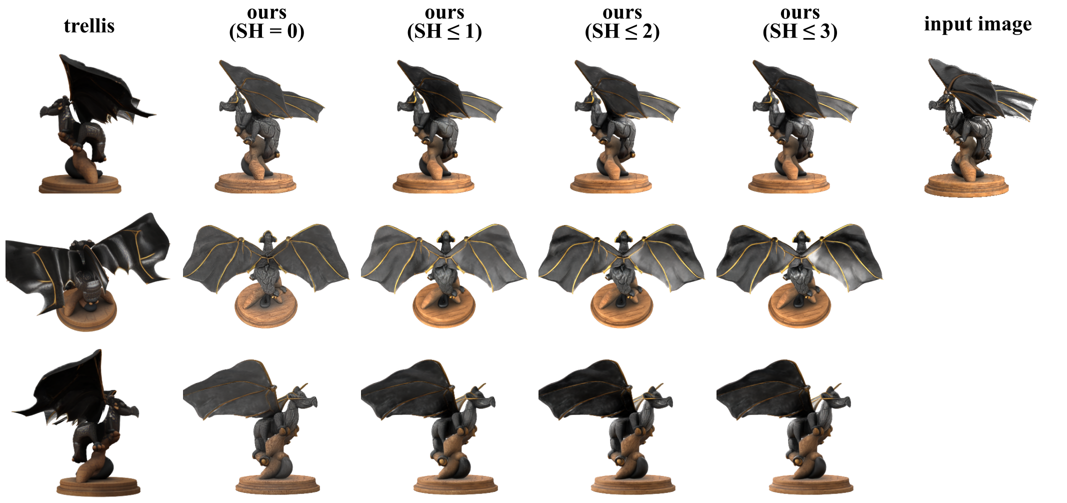


Figure S9: **Rendering with various spherical harmonics degrees in generation.** When restricted to zeroth-order spherical harmonics, our 3D Gaussians produce a view-independent appearance and avoid the over-exposed regions observed in TRELLIS's renderings. As we progressively incorporate higher-order spherical harmonics, our method yields increasingly pronounced view-dependent effects. Mesh credit: Vetech82 (2021).

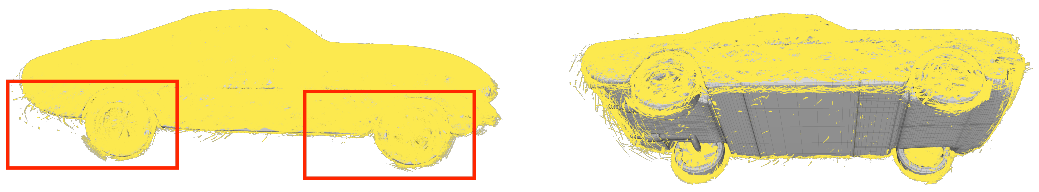


Figure S10: **RefNeRF dataset issues: inaccurate depths.**