FOCUSING: VIEW-CONSISTENT SPARSE VOXELS FOR EFFICIENT 3D VAE

Anonymous authors
Paper under double-blind review

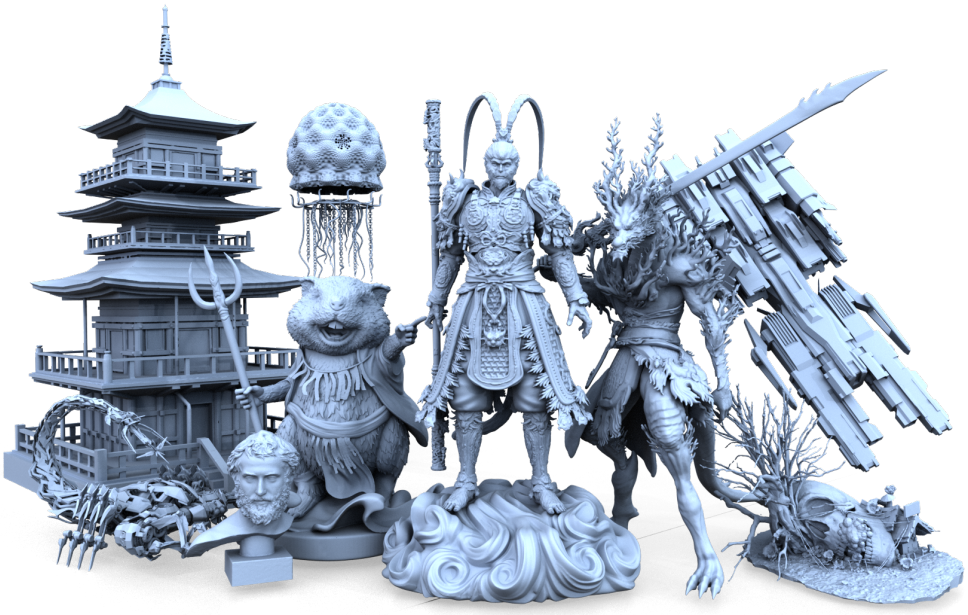

Figure 1: **Focusing** is an efficient and effective 3D VAE, capable of generating high-fidelity 3D models at 1024^3 resolution using less than 50GB of memory by using the render loss with an efficient voxel selection.

ABSTRACT

High-fidelity 3D generation remains difficult. Although some methods have proposed converting raw meshes to SDFs, it remains a lossy process. TripoSF presented a VAE training paradigm based on a rendering loss to circumvent this lossy SDF conversion, achieving high-precision surface reconstruction. However, because the rendering loss cannot supervise all the VAE outputs in the same way as SDF supervision, it limits detail and scalability.We present **Focusing**, a 3D VAE that improves efficiency by activating only the voxels that matter for a given view. Our key idea is a depth-driven voxel carving performed in the structured latent space: voxels inconsistent with the rendered depth are pruned before decoding. This concentrates learning on locally relevant geometry, reduces attention and decoding costs, and lowers video random access memory (VRAM) usage. To stabilize training and capture fine details, we further introduce an adaptive zooming strategy that adjusts camera intrinsics to keep the number of active voxels within a target range. The VAE is trained with a render-based loss on depth, normals, masks, and perceptual terms, and we add simple regularizers (e.g., sparse-voxel TV and a short warm-up with TSDF supervision) to reduce small holes and speed up convergence. Across standard reconstruction benchmarks, Focusing improves geometric accuracy (CD, F-score) over strong baselines while cutting VRAM consumption, which allows for training the 1024^3 resolution VAE on as little as 50GB

of VRAM. These results show that local, view-consistent sparsity is an effective route to higher-resolution, more efficient 3D VAEs.

1 INTRODUCTION

3D AI-generated content (AIGC) is an emerging research direction with broad applications in embodied AI, digital content creation, gaming, 3D modeling, and AR/XR. Compared to its 2D counterpart, 3D AIGC is considerably more challenging due to the higher spatial dimensionality and the need to model complex geometry and topology. These challenges make achieving high-fidelity 3D generation difficult in a manner that is both accurate and efficient, and the problem remains largely unsolved despite recent progress.

Following the success of 2D AIGC pipelines, most 3D AIGC approaches adopt a two-stage framework. In the first stage, a 3D variational auto-encoder (VAE) encodes a shape representation, such as a signed distance field (SDF), point cloud, or voxel grid, into a compact latent space. In the second stage, a generative model, often a latent diffusion model, operates in this reduced space to synthesize novel 3D assets. Within this pipeline, the 3D VAE plays a central role: its ability to faithfully compress and reconstruct complex geometry directly determines the fidelity of the final results. Improving the efficiency and accuracy of 3D VAEs is therefore a critical step toward high-quality, scalable 3D AIGC.

Since there is no single unified representation for 3D data, recent works have explored different designs for both the input/output and latent spaces of 3D VAEs. Methods such as VecSet Zhang et al. (2023) encode the input into a long tensor representation, but this introduces redundancy, since each latent dimension is correlated with all dimensions of the input. TRELLIS Xiang et al. (2025) addresses this by introducing Structured Latents, where the latent space is represented explicitly as sparse voxels. This structured design improves locality, geometric accuracy, and allows local editing by replacing specific voxels.

Sparc3D Li et al. (2025) and Direct3D-S2 Wu et al. (2025) extend this line by using SDFs for both input and output, thereby enforcing a unified modality. However, since most raw meshes are not watertight, these methods require lossy preprocessing steps to obtain SDFs. To overcome this limitation, rendering-based supervision is adopted by TripoSF He et al. (2025), where the VAE is trained to match rendered depth and normal maps instead of precomputed SDF values. This avoids watertight conversion altogether, making the pipeline more flexible for open or complex training data. To further reduce computation, they adopt Frustum-aware Sectional Voxel Training, which prunes voxels outside the rendering frustum, thereby lowering training costs and enabling 1024^3 upsampling. While such strategies reduce redundant computation, they also raise a fundamental question: *What is the minimum amount of computation required to render an image from a given viewpoint?*

TripoSF uses a point cloud as input and produces a mesh as output. This setup resembles the classical point cloud reconstruction pipeline, where high-quality meshes can be obtained from dense, oriented point clouds even without neural networks. Unlike TRELLIS, which requires global consistency to reproduce texture, a geometry-focused 3D VAE only needs local point distributions to recover surface patches.

Motivated by this observation and the redundancy of existing approaches, we propose **Focusing**, a local 3D VAE training scheme built on simple yet effective voxel carving. Following the TripoSF framework, we supervise the VAE with depth and normal maps rendered from ground-truth views. Crucially, before decoding, we perform voxel carving as shown in Figure 2: voxels in the structured latent are compared with the rendered depth map, and only those consistent with the view are retained. This discards most irrelevant voxels, enabling the decoder to focus on fine-grained features while substantially reducing VRAM and unnecessary attention computations.

To further improve detail capture, we introduce an adaptive camera adjustment strategy inspired by zooming. By dynamically adjusting the camera's intrinsic parameters, we maintain the number of activated voxels within a controllable range. This mechanism not only stabilizes VRAM usage across diverse inputs but also allows us to expand the field of view (FoV) to capture surface details more effectively. Together, these strategies enable efficient high-resolution training while preserving geometric fidelity. See Figure 1 for examples of high-resolution 3D models produced by our method.

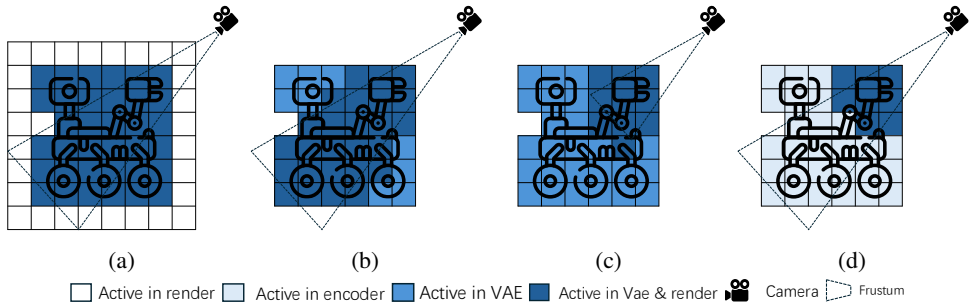

Figure 2: Conceptually illustration of sparse voxels in existing methods and in our approach. (a) TRELLIS encodes 3D models using sparse voxels and generates a mesh on a dense voxel grid for rendering. (b) TripoSF employs SparseFlex to reduce the number of voxels used for mesh extraction during rendering. (c) By adjusting the camera's far plane, TripoSF further reduces voxels that do not contribute to the final render. (d) Our method selects active voxels in the structured latent space based on the depth map. This strategy greatly reduces the computational overhead of both decoding and rendering, and it does not rely on the choice of the far plane.

2 Related Works

2.1 3D Representations

Meshes. Meshes are the most common and versatile representation for 3D assets, and they are directly applicable to many downstream applications, such as rendering, animation, and simulation. A line of work explores treating meshes as sequences and generating them with sequence-to-sequence models Nash et al. (2020); Siddiqui et al. (2024); Hao et al. (2024); Gao et al. (2025); Lionar et al. (2025). These methods can produce meshes with stylized or artist-like qualities, particularly quad meshes, but they are typically constrained by sequence length, which limits resolution and geometric fidelity. Other approaches attempt to directly predict mesh topology and geometry from images Wang et al. (2018) or point clouds Hanocka et al. (2020). More recently, differentiable rendering pipelines have been employed to supervise mesh generation with 2D projections. For example, TripoSF He et al. (2025) represents meshes using SparseFlex within a differentiable rendering pipeline, enabling efficient training and supporting higher-resolution mesh generation compared to prior mesh-based methods.

Point clouds. Point clouds, often obtained from scanning devices, have long served as input for 3D reconstruction algorithms. Their connectivity-free nature and flexibility make them appealing as neural network input Charles et al. (2017a); Qi et al. (2017). Other approaches generate point clouds directly as output distributions Achlioptas et al. (2018); Yang et al. (2019); Liu et al. (2021); Luo & Hu (2021), which allows flexible and high-precision modeling. However, converting point clouds into watertight meshes is both lossy and computationally expensive Kazhdan et al. (2006); Kazhdan & Hoppe (2013); Hou et al. (2022), which limits their use in many downstream tasks.

Implicit functions. Implicit functions encode 3D geometry as level sets of continuous fields, such as the zero level set of a signed distance function or the $1/2$-level set of an occupancy field. This formulation is robust and naturally compatible with neural networks, which explains its popularity in both reconstruction and generation tasks Park et al. (2019). Since implicit functions do not directly yield meshes, an additional extraction algorithm (e.g., marching cubes Lorensen & Cline (1987) or its variants) is required to recover surfaces. However, SDF- and occupancy-based approaches inherently assume watertight geometry, which limits their ability to handle open models. To relax this constraint, UDFs have been proposed Chibane et al. (2020), but existing UDF extraction methods often suffer from inefficiency and instability Zhou et al. (2022); Hou et al. (2023); Ren et al. (2022). A common workaround is to preprocess training data by inflating meshes into watertight versions, yet this conversion can introduce artifacts and reduce geometric fidelity. For example, Sparc3D Li et al. (2025) adopts a flood-filling and deformation-based repair pipeline inevitably leads to losing fine-scale details.

2.2 3D VAE ARCHITECTURES

Inspired by the success of 2D diffusion models Rombach et al. (2022), most current 3D generative models follow a two-stage framework: a 3D VAE first compresses the input representation into a compact latent space, and a 3D diffusion model then operates in this reduced space to generate novel shapes. This design allows high-resolution synthesis while keeping diffusion tractable. Unlike 2D AIGC, however, there is no universally accepted compact representation for 3D data. The vast majority of available assets are stored as non-compact meshes, which are not ideal for direct neural network training due to irregular connectivity and variable topology.

3DShape2VecSet Zhang et al. (2023) adopts a VAE to learn a compact vector-set (vecset) representation from point clouds sampled on meshes. The decoder is trained by supervising SDF values at query points, providing implicit geometric supervision during reconstruction. Clay Zhang et al. (2024) extends this approach to large-scale datasets and introduces an inflation-based preprocessing pipeline to enforce watertight meshes. Dora Chen et al. (2025) and Huanyuan2 Team (2025) enhance this framework with importance sampling, which improves the ability to capture sharp features. Hi3DGen Ye et al. (2025) further incorporates normal-map supervision to boost the fidelity of surface detail reconstruction. Despite these advances, vecset representations suffer from significant information redundancy: each feature is correlated with the entire 3D model, making training inefficient and hindering scalability.

Both XCube Ren et al. (2024) and TRELLIS Xiang et al. (2025) replace the global vecset with sparse voxels. In this formulation, each voxel encodes local geometric information, leading to higher-quality reconstructions and more structured latent representations. TRELLIS Xiang et al. (2025) further demonstrates that selectively replacing certain voxels enables flexible local 3D editing. Direct3D-S2 Wu et al. (2025) introduces Spatial Sparse Attention to restrict computations to local neighborhoods, thereby reducing overhead during the diffusion stage. Sparc3D Li et al. (2025) proposes a new preprocessing strategy to improve geometric fidelity. TripoSF He et al. (2025) shifts away from SDF-based supervision and instead employs a rendering loss that compares predicted depth and normal maps with ground truth. This approach avoids the accuracy degradation caused by lossy watertight conversion and allows the VAE to handle both internal structures and open boundaries. However, render-based supervision provides weaker constraints than SDF supervision, since only voxels contributing to visible surfaces are directly trained. As a result, many latent voxels remain under-regularized, which can limit reconstruction accuracy and consistency.

3 METHOD

3.1 PRELIMINARIES

TripoSF He et al. (2025) introduced SparseFlex, a sparse version of FlexibleCubes Shen et al. (2023), to train 3D AIGC models. SparseFlex is defined by a set of voxels $\mathcal{V}$. Each voxel $v_i \in \mathcal{V}$ contains both its spatial location (x_i, y_i, z_i) (the 3D coordinates of its center) and feature information. Let the number of voxels be N_v and the number of corresponding corners be N_c. The feature includes the SDF values $\{s_j \mid 0 \leq j < N_c\}$ and deformations $\{\delta_j \mid 0 \leq j < N_c\}$ for the voxel's eight corners. In practice, the features for each corner are obtained by averaging the values from surrounding relevant voxels. Additionally, the feature also contains the interpolation weights $\left\{\alpha_i \in \mathbb{R}^8_{>0}, \beta_i \in \mathbb{R}^{12}_{>0} \mid 0 \leq i < N_v\right\}$ per voxel for Dual Marching Cubes (DMC) Nielson (2004). Formally, the SparseFlex representation, S , is defined as:

$$\mathcal{S} = (\mathcal{V}, \mathcal{F}_c, \mathcal{F}_v), \quad \mathcal{F}_c = \{s_j, \delta_j\}, \quad \mathcal{F}_v = \{\alpha_i, \beta_i\}, \tag{1}$$

where $\mathcal{F}_c$ contains the SDF values and deformations at the corner grids, and $\mathcal{F}_v$ contains the interpolation weights for each voxel.

SparseFlex significantly reduces memory consumption and cuts down on unnecessary computational costs. Moreover, by supervising with a rendering loss instead of direct SDF values, TripoSF avoids the lossy watertight mesh conversion process. To further reduce computational overhead, TripoSF introduced Frustum-aware Sectional Voxel Training. This method applies SparseFlex to extract meshes only from voxels that are within the current camera's Normalized Device Coordinates (NDC)

space. By adjusting the camera's intrinsic parameters, this cropping operation also enables the learning of a model's internal structure.

TripoSF then trains a VAE with following losses:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{render}} + \lambda_2 \mathcal{L}_{\text{occ}} + \lambda_3 \mathcal{L}_{\text{KL}} + \lambda_4 \mathcal{L}_{\text{flex}} \tag{2}$$

$\mathcal{L}_{\text{render}}$ is the rendering supervision loss including the following items:

$$\mathcal{L}_{\text{render}} = \lambda_d \mathcal{L}_d + \lambda_n \mathcal{L}_n + \lambda_m \mathcal{L}_m + \lambda_{ss} \mathcal{L}_{ss} + \lambda_{lp} \mathcal{L}_{lp} \tag{3}$$

where $\mathcal{L}_d$, $\mathcal{L}_n$, and $\mathcal{L}_m$ denote the L_1 loss for depth maps, normal maps, and mask maps, respectively. $\mathcal{L}_{ss}$ and $\mathcal{L}_{lp}$ denote SSIM loss and LPIPS loss, and are only applied to normal maps. $\mathcal{L}_{KL}$ is the KL divergence between the learned latent distribution and a standard normal prior, regularizing the latent space. $\mathcal{L}_{flex}$ is the regularization term from Flexicubes to encourage smooth SDF values.

While SparseFlex avoids the accuracy loss from computing SDFs, its training still faces several issues:

- **Irrelevant voxels in the NDC space.** A significant number of invisible voxels participate in mesh extraction and require large VRAM, which has significant influence for sparse voxel resolution scaling up. Although controlling the near and far planes can help, this approach is often suboptimal. To address this, we propose a plug-in-and-play visibility-based voxel carving strategy in Section 3.2 to more effectively reduce the number of active voxels. We also show that this pruning can be performed in the latent space, which greatly reduces the decoder's workload on irrelevant voxels.
- **Rendering blurriness.** Methods based on rendering loss can suffer from blurriness due to the resolution of the rendered image. Capturing accurate detail often requires a higher resolution or a closer camera view, with the former significantly increasing computational cost. In Section 3.3, we introduce an adaptive camera adjustment strategy based on the dolly zoom effect. By controlling the number of voxels to be activated within the view, we can flexibly adjust camera parameters to better supervise the model's fine details.
- **Generating unnecessary holes.** Unlike direct SDF supervision, which provides a strict signal for maintaining watertightness, small holes are difficult to effectively supervise with a rendering loss. In Section 3.4, we introduce our VAE framework and propose a new regularization loss to reduce these holes.

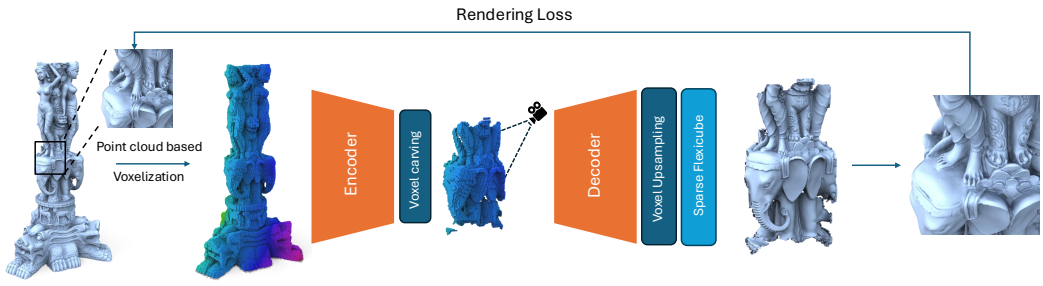

Figure 3: Overview of our framework. Starting from the input mesh, we first voxelize it and aggregate local features from sampled point clouds to form input voxels. A sparse transformer encoder-decoder then compresses these structured features into a latent space, followed by an upsampling module to increase resolution. In the latent space, we perform visibility-based voxel carving to retain only view-consistent voxels that contribute to the visible mesh in the rendered image. The refined structured features are decoded into SparseFlex for final mesh extraction. Supervision is provided by a rendering loss, which compares the rendered images of the input and reconstructed meshes.

3.2 DEPTH-BASED VOXEL CARVING

Our key insight is that the local surface is determined by local points, independent of distant points. As shown in Figure 3, selecting only a subset of voxels in the latent space and feeding them to the decoder can still generate a locally complete mesh, except for some jagged noise at the boundary. Since a rendering loss based VAE does not require a globally complete mesh, we can minimize the number of voxels in the network without affecting the rendering result by aligning the input camera with the filtered voxels.

Specifically, given a camera with extrinsic π, intrinsics K, and the near (n) and far (f) clipping planes of the viewing frustum, we compute the Model-View-Projection matrix to render the depth map D from the ground truth mesh. For each voxel z_i in the latent code $\mathcal{Z}$, we also project its center to obtain the projected point $z_i^p = \{u_i^p, v_i^p, d_i^p\}$ in the image space and regard d_i^p as the depth of each voxel z_i. Using a threshold r, we then filter all voxels where $d_i^p > D(u_i^p, v_i^p) - r$ to obtain the view-consistent voxels $\mathcal{Z}_{carve}^p$. To enhance robustness at the image boundary, the $3 \times 3 \times 3$ neighborhood of any remaining voxel is also retained.

3.3 ADAPTIVE ZOOMING

The significant variation in the number of visible voxels leads to significant VRAM fluctuations during training. TRELLIS addresses this by removing voxels exceeding a fixed quota, but this can impact the final render if essential voxels are lost. TripoSF utilizes a visibility ratio α, to control the number of active voxels, primarily by adjusting the near and far planes. This approach relies on the camera being preset close to the object's surface to capture key details.

In contrast, since we've already removed most invisible voxels, adjusting the far plane has little influence on the number of voxels. We introduce a zooming-based voxel count adjustment method. This approach effectively controls the number of active voxels while enabling flexible zoom-in operations to capture finer model details.

Specifically, we pick a random projected voxel in $\mathcal{Z}^p$ as the center. We randomly pick a number between N_{min} and N_{max} to perform the KNN search in image space to get the crop voxel set $\mathcal{Z}_{crop}^p$ from $\mathcal{Z}_{carve}^p$. We then modify the perspective matrix P with the new image bounding box $(x_{min}^n, x_{max}^n, y_{min}^n, y_{max}^n)$ as follows:

$$P_{new} = \begin{pmatrix} \frac{2s}{x_{max}^n - x_{min}^n} & 0 & 0 & -\frac{x_{max}^n + x_{min}^n}{x_{max}^n - x_{min}^n} \\ 0 & \frac{2s}{y_{max}^n - y_{min}^n} & 0 & -\frac{x_{max}^n + x_{min}^n}{x_{max}^n - x_{min}^n} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} P \tag{4}$$

where s is the scaling rate to avoid jagged noise appearing at the boundary of the image.

3.4 VAE STRUCTURE

Following TripoSF, we use a variational autoencoder that compresses the input oriented dense point cloud into a sparse latent code $\mathcal{Z}$ without relying on computationally expensive global attentions. A frozen PointNet Charles et al. (2017b) adapted from TripoSF is used to aggregate local geometric features within each voxel. A sparse transformer then utilizes the shifted window attention proposed by TRELLIS to learn relationships between voxels. This process outputs a fused structured latent feature $\mathcal{Z}$, where each voxel possesses local geometric information. The voxel carving and adaptive zooming is then used to filter the voxels in $\mathcal{Z}$ to attain $\mathcal{Z}_{crop}^p$. The decoder takes $\mathcal{Z}_{crop}^p$ as input and uses a series of transformer layers to generate the final output. To support high-resolution output, two self-pruning upsampling modules are then employed to obtain a 2x upsampled result from the initial output, ultimately yielding a 4x resolution output. We only use $\mathcal{L}_{occ}$ to supervise self-pruning block and do not use O_{pred} to filter voxels in training, as this may create unnecessary holes, thereby negatively impacting the rendering loss.

To accelerate VAE convergence and provide initial VAE weights capable of extracting meshes, we supervise s_j using the TSDF s_j^{raw} calculated from raw meshes during the early stages of training:

$$\mathcal{L}_{tsdf} = \sum_{j=1}^{N_c} ||s_j - s_j^{raw}|| \tag{5}$$

To reduce the generation of holes, we introduce the Total Variation loss on sparse voxels to suppress the differences in SDF values between adjacent corners:

$$\mathcal{L}_{tv} = \sum_{d\in[D]} \sqrt{\nabla_x^2(V,d) + \nabla_y^2(V,d) + \nabla_z^2(V,d)} \tag{6}$$

The overall loss function is:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{render}} + \lambda_2 \mathcal{L}_{\text{occ}} + \lambda_3 \mathcal{L}_{\text{KL}} + \lambda_4 \mathcal{L}_{\text{flex}} + \lambda_5 \mathcal{L}_{tv} + \lambda_6 \mathcal{L}_{tsdf} \tag{7}$$

3.5 Rectified Flow based Image to 3D Generation

Drawing inspiration from TRELLIS Xiang et al. (2025) and TripoSF He et al. (2025), we employ a two-stage rectified-flow generation model, which consists of a sparse structure flow model and a structured latents flow model.

Sparse structure flow model. Following the approach of TRELLIS, we first train a Sparse Structure VAE based on voxel carving. This VAE compresses a 3D shape into a smaller resolution using 3D convolutions. Thanks to our preliminary voxel carving process, we can train a high-resolution Sparse Structure VAE with relative ease. Subsequently, we extract DINO features from the condition image and inject them into a Diffusion Transformer (DiT) via cross-attention. We then employ a rectified flow model to denoise the noised latents.

Structured latents flow model. Based on our proposed VAE, we first encode the sampled point cloud and its corresponding sparse structure into the latent space. Similar to the sparse structure flow model, we inject the DINO features of the condition image into the DiT (Diffusion Transformer) via cross-attention. We then use a rectified flow model for denoising. Finally, the denoised latents are decoded into a 3D shape by the VAE's decoder.

4 Experiments

4.1 Experiment Settings

Implementation Details. Following TRELLIS Xiang et al. (2025), we train both the VAE and its latent flow model on 183K high-quality assets from Objaverse-XL Deitke et al. (2023). We employ a progressive training scheme for our VAE. The 512^3 resolution VAE runs on 32 A800 GPUs (batch size 32) with AdamW (initial LR 1 X 10^{-4}, weight decay 0.01) for two days. A cosine annealing learning rate schedule with 40K steps is used to adjust the learning rate. We then train the 1024^3 resolution VAE with 32 A100 GPUs using the same setting.

Hyperparameter Settings. We use the same weight configuration as TRELLIS to train our VAE. For our two newly introduced losses, we set $\lambda_5 = 0.001$ and $\lambda_6 = 1$. The $\mathcal{L}_{tsdf}$ is only used for the first 12K steps. We set the threshold r equal to 2/resolution to carve voxels. We use a resolution of 518^2 to render images and use $s = 1.1$ to calculate the new perspective matrix.

4.2 VAE Reconstruction Evaluation

We primarily compare with TripoSF, as currently only TripoSF supports raw mesh reconstruction. Our method demonstrates superior VAE reconstruction performance with quantitative results detailed in Table 1. Our model achieves better metrics than TripoSF at the same resolution in terms of the L2 norm of Chamfer Distance (CD) and F-score with a threshold 0.005. Since even the Level 4 data in the Dora Benchmarks lack sufficient detail, we selected several detail-rich models from online websites to further test our method's ability to capture model details, as shown in Figure 5.

Our method demonstrates results comparable to TripoSF's 1024^3 resolution output even at 512^3 resolution and captures geometry details better at 1024^3 resolution.

We further tested the extraction results of the 1024^3 resolution network weights at different resolutions in Figure 4. We found that our model has the ability to complete 1536^3 reconstruction without being trained at the corresponding resolution.

Table 1: We sample 100K point cloud to measure the Chamfer Distance and F-score on the Dora Benchmark Chen et al. (2025) in different geometry details Levels. Low Chamfer Distance ensures overall shape fidelity, while high F-score ensures local details are accurately covered within an acceptable error radius.

	Chamfer Distance (10^{-5})							
	L1		L2		L3		L4	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
TripoSF_{512}	1.382	0.985	1.600	1.189	2.184	1.368	3.107	2.126
Ours_{512}	1.353	0.995	1.513	1.008	2.116	1.351	2.806	1.771
TripoSF_{1024}	1.315	0.937	1.456	0.936	2.007	1.197	2.431	1.390
Ours_{1024}	1.294	0.886	1.429	0.873	1.901	1.011	2.264	1.055
	F-score							
	L1		L2		L3		L4	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
TripoSF_{512}	0.951	0.078	0.939	0.089	0.892	0.115	0.831	0.146
Ours_{512}	0.953	0.080	0.945	0.086	0.897	0.117	0.841	0.147
TripoSF_{1024}	0.957	0.074	0.951	0.078	0.907	0.106	0.873	0.121
Ours_{1024}	0.958	0.070	0.953	0.073	0.916	0.091	0.886	0.097

4.3 Image-to-3D Generation

We further validate our VAE's utility as a generative foundation model. Visualizations in Figure 6, including image-to-3D results from in-the-wild images, highlight the generalization of our method. The generated 3D shapes maintain sharp edges and rich details while exhibiting high fidelity to the corresponding input images.

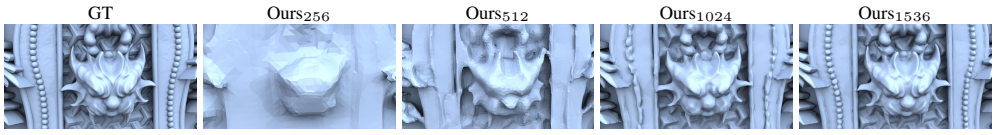

Figure 4: We used the 1024^3 resolution checkpoint to test the reconstruction results at different resolutions. Our method demonstrated generalization capabilities across different resolutions.

5 Conclusions & Limitaion

We present an efficient voxel carving scheme for 3D VAE training and 3D model generation. Only the visible voxels of the structured latent are sent to the VAE decoder for generation. This efficiently removes redundant computation, thereby enabling high-resolution and detailed mesh generation. Experiments demonstrate that our method outperforms SOTA works in terms of efficiency and accuracy.

Currently, our pipeline uses point cloud features as input and is highly dependent on the normal vectors of the point cloud to predict the final result. When the object to be reconstructed is very thin, such as the hair in the last example of Figure 5, it is difficult to obtain a locally consistent normal

vector to capture these fine details. In the future, we hope to find a better input feature to overcome this limitation.

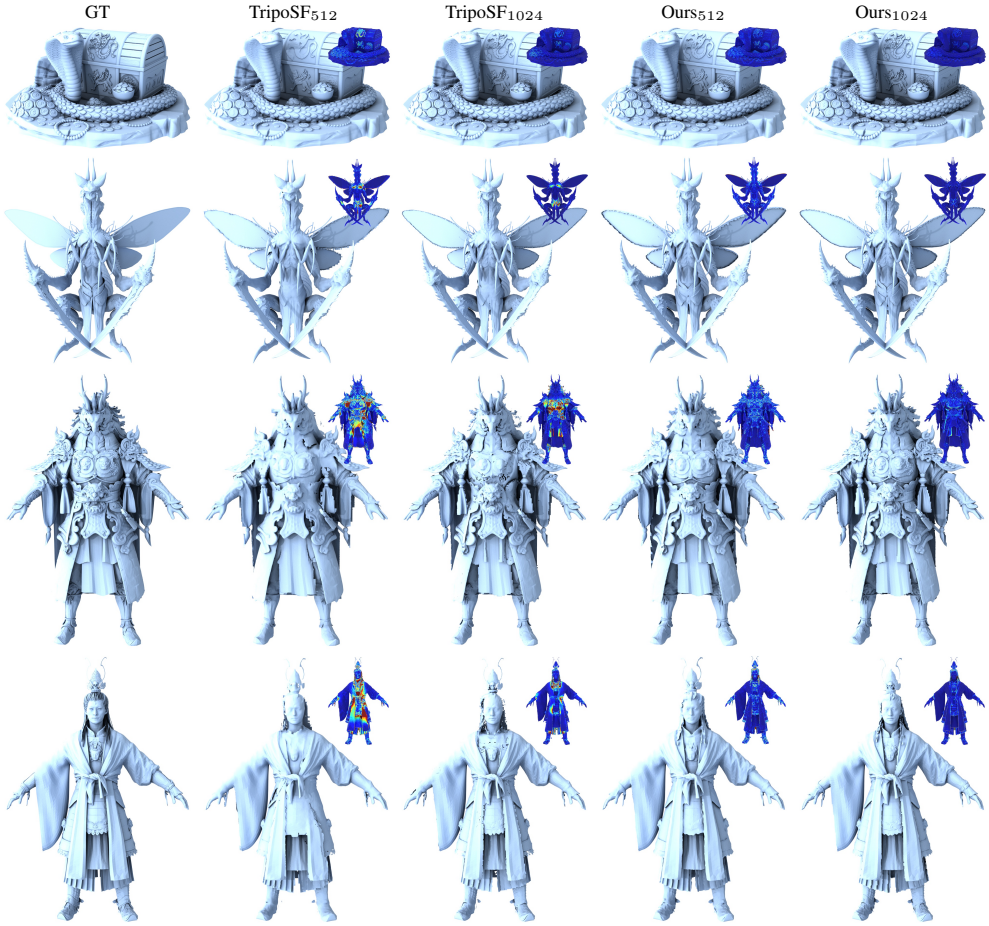

Figure 5: Qualitative comparison of VAE reconstruction between ours and TripoSF with different resolution. Our approach demonstrate superior performance in reconstructing geometry details.

Figure 6: Single image-to-3D generations with in-the-wild images which are collected from Geminis or Dora.

6 Ethics Statement

All datasets and models used in this study are publicly available and have been used in accordance with their respective licenses and terms of use.

7 Reproducibility Statement

We have clarified our experiment setting in Section 4.1. We will open-source the code and release the trained model.

References

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3d point clouds. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 40–49. PMLR, 2018.

R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017a.

R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017b.

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 *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 16251–16261, June 2025.

Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, December 2020.

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.

Daoyi Gao, Yawar Siddiqui, Lei Li, and Angela Dai. Meshart: Generating articulated meshes with structure-guided transformers. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 618–627, 2025.

Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *arXiv preprint arXiv:2005.11084*, 2020.

Zekun Hao, David W Romero, Tsung-Yi Lin, and Ming-Yu Liu. Meshtron: High-fidelity, artist-like 3d mesh generation at scale. *arXiv preprint arXiv:2412.09548*, 2024.

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. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Fei Hou, Chiyu Wang, Wencheng Wang, Hong Qin, Chen Qian, and Ying He. Iterative poisson surface reconstruction (ipsr) for unoriented points. *ACM Trans. Graph.*, 41(4), 2022.

Fei Hou, Xuhui Chen, Wencheng Wang, Hong Qin, and Ying He. Robust zero level-set extraction from unsigned distance fields based on double covering. *ACM Trans. Graph.*, 42(6), 2023.

Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3), 2013.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pp. 61–70, 2006.

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*, 2025.

Stefan Lionar, Jiabin Liang, and Gim Hee Lee. Treemeshgpt: Artistic mesh generation with autoregressive tree sequencing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 26608–26617, 2025.

Yanchao Liu, Jianwei Guo, Bedrich Benes, Oliver Deussen, Xiaopeng Zhang, and Hui Huang. Treepartnet: neural decomposition of point clouds for 3d tree reconstruction. *ACM Trans. Graph.*, 40(6), December 2021.

William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987.

Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2837–2845, 2021.

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. Polygen: an autoregressive generative model of 3d meshes. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20, 2020.

Gregory M Nielson. Dual marching cubes. In *IEEE visualization 2004*, pp. 489–496. IEEE, 2004.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5099–5108, 2017.

Siyu Ren, Junhui Hou, Xiaodong Chen, Ying He, and Wenping Wang. Geoudf: Surface reconstruction from 3d point clouds via geometry-guided distance representation. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14168–14178, 2022.

Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

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

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 Trans. Graph.*, 42(4), jul 2023. ISSN 0730-0301.

Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19615–19625, 2024.

Tencent Hunyuan3D Team. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation, 2025.

Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

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

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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21469–21480, 2025.

Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge J. Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 4540–4549, 2019.

Chongjie Ye, Yushuang Wu, Ziteng Lu, Jiahao Chang, Xiaoyang Guo, Jiaqing Zhou, Hao Zhao, and Xiaoguang Han. Hi3dgen: High-fidelity 3d geometry generation from images via normal bridging. *arXiv preprint arXiv:2503.22236*, 2025.

Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Trans. Graph.*, 42(4), jul 2023. ISSN 0730-0301.

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. *ACM Transactions on Graphics (TOG)*, 43(4):1–20, 2024.

Junsheng Zhou, Baorui Ma, Yu-Shen Liu, Yi Fang, and Zhizhong Han. Learning consistency-aware unsigned distance functions progressively from raw point clouds. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.