

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

000
001
002
003
004
005 **Anonymous authors**
006 Paper under double-blind review
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030



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

ABSTRACT

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

054 of VRAM. These results show that local, view-consistent sparsity is an effective
 055 route to higher-resolution, more efficient 3D VAEs.
 056

057 1 INTRODUCTION

060 3D AI-generated content (AIGC) is an emerging research direction with broad applications in em-
 061 bodied AI, digital content creation, gaming, 3D modeling, and AR/XR. Compared to its 2D coun-
 062 terpart, 3D AIGC is considerably more challenging due to the higher spatial dimensionality and the
 063 need to model complex geometry and topology. These challenges make achieving high-fidelity 3D
 064 generation difficult in a manner that is both accurate and efficient, and the problem remains largely
 065 unsolved despite recent progress.
 066

067 Following the success of 2D AIGC pipelines, most 3D AIGC approaches adopt a two-stage frame-
 068 work. In the first stage, a 3D variational auto-encoder (VAE) encodes a shape representation, such
 069 as a signed distance field (SDF), point cloud, or voxel grid, into a compact latent space. In the
 070 second stage, a generative model, often a latent diffusion model, operates in this reduced space to
 071 synthesize novel 3D assets. Within this pipeline, the 3D VAE plays a central role: its ability to faith-
 072 fully compress and reconstruct complex geometry directly determines the fidelity of the final results.
 073 Improving the efficiency and accuracy of 3D VAEs is therefore a critical step toward high-quality,
 074 scalable 3D AIGC.
 075

076 Since there is no single unified representation for 3D data, recent works have explored different
 077 designs for both the input/output and latent spaces of 3D VAEs. Methods such as VecSet Zhang
 078 et al. (2023) encode the input into a long tensor representation, but this introduces redundancy, since
 079 each latent dimension is correlated with all dimensions of the input. TRELLIS Xiang et al. (2025)
 080 addresses this by introducing Structured Latents, where the latent space is represented explicitly as
 081 sparse voxels. This structured design improves locality, geometric accuracy, and allows local editing
 082 by replacing specific voxels.
 083

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

095 TripoSF uses a point cloud as input and produces a mesh as output. This setup resembles the classical
 096 point cloud reconstruction pipeline, where high-quality meshes can be obtained from dense, oriented
 097 point clouds even without neural networks. Unlike TRELLIS, which requires global consistency
 098 to reproduce texture, a geometry-focused 3D VAE only needs local point distributions to recover
 099 surface patches.
 100

101 Motivated by this observation and the redundancy of existing approaches, we propose **Focusing**, a
 102 local 3D VAE training scheme built on simple yet effective voxel carving. Following the TripoSF
 103 framework, we supervise the VAE with depth and normal maps rendered from ground-truth views.
 104 Crucially, before decoding, we perform voxel carving as shown in Figure 2: voxels in the structured
 105 latent are compared with the rendered depth map, and only those consistent with the view are re-
 106 tained. This discards most irrelevant voxels, enabling the decoder to focus on fine-grained features
 107 while substantially reducing VRAM and unnecessary attention computations.
 108

109 To further improve detail capture, we introduce an adaptive camera adjustment strategy inspired
 110 by zooming. By dynamically adjusting the camera’s intrinsic parameters, we maintain the number
 111 of activated voxels within a controllable range. This mechanism not only stabilizes VRAM usage
 112 across diverse inputs but also allows us to expand the field of view (FoV) to capture surface details
 113 more effectively. Together, these strategies enable efficient high-resolution training while preserving
 114 geometric fidelity. See Figure 1 for examples of high-resolution 3D models produced by our method.
 115

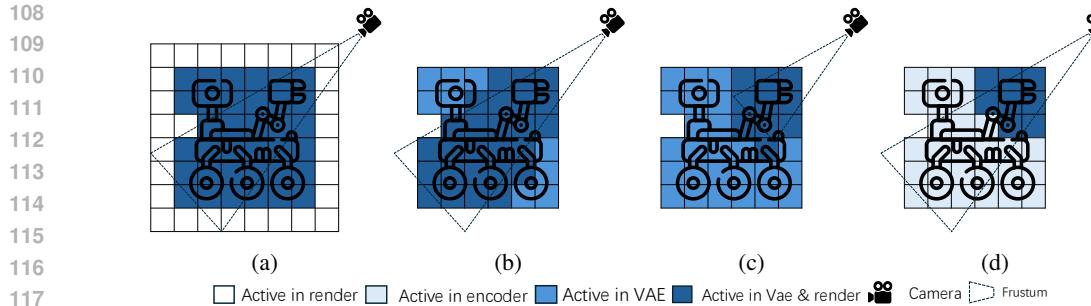


Figure 2: Conceptually illustration of sparse voxels in existing methods and in our approach. (a) TRELLO encodes 3D models using sparse voxels and generates a mesh on a dense voxel grid for rendering. (b) TripoSF employs SparseFlex to reduce the number of voxels used for mesh extraction during rendering. (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. (2017); 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 represent 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 produce meshes, an additional extraction algorithm (e.g., marching cubes Lorensen & Cline (1987) or its variants) is needed to recover surfaces. However, SDF- and occupancy-based approaches assume watertight geometry, which limits their ability to work with 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.

162 2.2 3D VAE ARCHITECTURES
163

164 Inspired by the success of 2D diffusion models Rombach et al. (2022), most current 3D generative
165 models follow a two-stage framework: a 3D VAE first compresses the input representation into
166 a compact latent space, and a 3D diffusion model then operates in this reduced space to generate
167 novel shapes. This design allows high-resolution synthesis while keeping diffusion tractable. Unlike
168 2D AIGC, however, there is no universally accepted compact representation for 3D data. The vast
169 majority of available assets are stored as non-compact meshes, which are not ideal for direct neural
170 network training due to irregular connectivity and variable topology.

171 3DShape2VecSet Zhang et al. (2023) adopts a VAE to learn a compact vector-set (vecset) repre-
172 sentation from point clouds sampled on meshes. The decoder is trained by supervising SDF values
173 at query points, providing implicit geometric supervision during reconstruction. Clay Zhang et al.
174 (2024) extends this approach to large-scale datasets and introduces an inflation-based preprocess-
175 ing pipeline to enforce watertight meshes. Dora Chen et al. (2025) and Huanyuan2 Team (2025)
176 enhance this framework with importance sampling, which improves the ability to capture sharp fea-
177 tures. Hi3DGen Ye et al. (2025) further incorporates normal-map supervision to boost the fidelity
178 of surface detail reconstruction. Despite these advances, vecset representations suffer from signifi-
179 cant information redundancy: each feature is correlated with the entire 3D model, making training
180 inefficient and hindering scalability.

181 Both XCube Ren et al. (2024) and TRELLIS Xiang et al. (2025) replace the global vecset with
182 sparse voxels. In this formulation, each voxel encodes local geometric information, leading to
183 higher-quality reconstructions and more structured latent representations. TRELLIS Xiang et al.
184 (2025) further demonstrates that selectively replacing certain voxels enables flexible local 3D edit-
185 ing. Direct3D-S2 Wu et al. (2025) introduces Spatial Sparse Attention to restrict computations to
186 local neighborhoods, thereby reducing overhead during the diffusion stage. Sparc3D Li et al. (2025)
187 proposes a new preprocessing strategy to improve geometric fidelity. TripoSF He et al. (2025) shifts
188 away from SDF-based supervision and instead employs a rendering loss that compares predicted
189 depth and normal maps with ground truth. This approach avoids the accuracy degradation caused by
190 lossy watertight conversion and allows the VAE to handle both internal structures and open bound-
191 aries. However, render-based supervision provides weaker constraints than SDF supervision, since
192 only voxels contributing to visible surfaces are directly trained. As a result, many latent voxels
193 remain under-regularized, which can limit reconstruction accuracy and consistency.

194 3 METHOD
195196 3.1 PRELIMINARIES
197

198 TripoSF He et al. (2025) introduced SparseFlex, a sparse version of FlexibleCubes Shen et al.
199 (2023), to train 3D AIGC models. SparseFlex is defined by a set of voxels \mathcal{V} . Each voxel $v_i \in \mathcal{V}$
200 contains both its spatial location (x_i, y_i, z_i) (the 3D coordinates of its center) and feature infor-
201 mation. Let the number of voxels be N_v and the number of corresponding corners be N_c . The
202 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
203 voxel's eight corners. In practice, the features for each corner are obtained by averaging the values
204 from surrounding relevant voxels. Additionally, the feature also contains the interpolation weights
205 $\{\alpha_i \in \mathbb{R}_{>0}^8, \beta_i \in \mathbb{R}_{>0}^{12} \mid 0 \leq i < N_v\}$ per voxel for Dual Marching Cubes (DMC) Nielson (2004).
206 Formally, the SparseFlex representation, S , is defined as:

$$207 \quad S = (\mathcal{V}, \mathcal{F}_c, \mathcal{F}_v), \quad \mathcal{F}_c = \{s_j, \delta_j\}, \quad \mathcal{F}_v = \{\alpha_i, \beta_i\}, \quad (1)$$

208 where \mathcal{F}_c contains the SDF values and deformations at the corner grids, and \mathcal{F}_v contains the inter-
209 polation weights for each voxel.

210 SparseFlex significantly reduces memory consumption and cuts down on unnecessary computational
211 costs. Moreover, by supervising with a rendering loss instead of direct SDF values, TripoSF avoids
212 the lossy watertight mesh conversion process. To further reduce computational overhead, TripoSF
213 introduced Frustum-aware Sectional Voxel Training. This method applies SparseFlex to extract
214 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:

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

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

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

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. TripoSF **culls voxels that are far from the surface during the upsampling process in the decoder**. The \mathcal{L}_{occ} loss is used to guide the self-pruning upsampling module employed by TripoSF to accurately remove these distant voxels. Although this approach can reduce the number of voxels, it also creates more holes and makes convergence more difficult. \mathcal{L}_{KL} is the KL divergence between the learned latent distribution and a standard normal prior, which helps to regularize the latent space. $\mathcal{L}_{\text{flex}}$ is the regularization term from Flexicubes that promotes 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.

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 $\mathcal{Z}_{\text{carve}}^p$. To enhance robustness at the image boundary, the $3 \times 3 \times 3$ neighborhood of any remaining voxel is also retained.

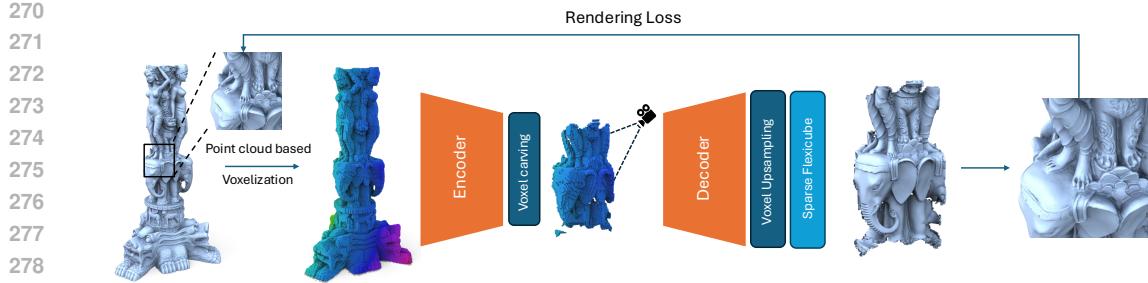


Figure 3: Overview of our framework. Starting from the input mesh, we first voxelize it and aggregate local features from sampled point clouds to form input voxels. A sparse transformer encoder-decoder then compresses these structured features into a latent space, followed by an upsampling module to 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.

3.3 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.

We randomly select a visibility ratio $\alpha \in [\alpha_{\min}, \alpha_{\max}]$ and retain only αN voxels to capture various levels of geometric structure. To limit the maximum number of voxels and avoid having too few, αN is clamped between N_{\min} and N_{\max} . We conduct a KNN search in image space to obtain the cropped voxel set $\mathcal{Z}_{\text{crop}}^p$ from $\mathcal{Z}_{\text{carve}}^p$. We then adjust the perspective matrix P using the new image bounding box $(x_{\min}^n, x_{\max}^n, y_{\min}^n, y_{\max}^n)$ as follows:

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

where s is the scaling rate used to prevent jagged noise from 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. (2017) 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 $\mathcal{Z}_{\text{crop}}^p$. The decoder takes $\mathcal{Z}_{\text{crop}}^p$ 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 the self-pruning block and do not use the output of the self-pruning block, O_{pred} , to filter voxels during training, in order to avoid unnecessary holes in the rendered images that negatively impact the rendering loss.**

324 To accelerate VAE convergence and provide initial VAE weights capable of extracting meshes, we
 325 supervise s_j using the TSDF s_j^{raw} calculated from raw meshes during the early stages of training:
 326

$$327 \quad \mathcal{L}_{tsdf} = \sum_{j=1}^{N_c} \|s_j - s_j^{raw}\| \quad (5)$$

331 To reduce the generation of holes, we introduce the Total Variation loss on sparse voxels to suppress
 332 the differences in SDF values between adjacent corners:
 333

$$334 \quad \mathcal{L}_{tv} = \sum_{d \in [D]} \sqrt{\Delta_x^2(V, d) + \Delta_y^2(V, d) + \Delta_z^2(V, d)} \quad (6)$$

337 where $\Delta_x^2(V, d)$ denotes the squared difference between the value of d -th channel in voxel $v :=$
 338 $(i; j; k)$ and the d th value in voxel $(i + 1; j; k)$, which can be analogously extended to $\Delta_y^2(V, d)$
 339 and $\Delta_z^2(V, d)$. We apply the TV term above to the SDF grid, denoted by \mathcal{L}_{tv} , which encourages a
 340 continuous and compact geometry.

341 The overall loss function is:
 342

$$344 \quad \mathcal{L} = \lambda_1 \mathcal{L}_{\text{render}} + \lambda_2 \mathcal{L}_{\text{occ}} + \lambda_3 \mathcal{L}_{\text{KL}} + \lambda_4 \mathcal{L}_{\text{flex}} + \lambda_5 \mathcal{L}_{tv} + \lambda_6 \mathcal{L}_{tsdf} \quad (7)$$

346 3.5 RECTIFIED FLOW BASED IMAGE TO 3D GENERATION

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

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

357 **Structured latents flow model.** Based on our proposed VAE, we first encode the sampled point
 358 cloud and its corresponding sparse structure into the latent space. **Similar to the sparse structure**
 359 **flow model, we use the DINO features of the condition image as the condition to train the DiT via**
 360 **cross-attention.** We then use a rectified flow model for denoising. Finally, the denoised latents are
 361 decoded into a 3D shape by the VAE’s decoder.

363 4 EXPERIMENTS

365 4.1 EXPERIMENT SETTINGS

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

373 **Hyperparameter Settings.** We use the same weight configuration as TRELLIS to train our VAE.
 374 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
 375 first 12K steps. We set the threshold r equal to $2/\text{resolution}$ to carve voxels. We use a resolution of
 376 518^2 to render images and use $s = 1.1$ to calculate the new perspective matrix. **For zooming, we set**
 377 **$\alpha_{\min} = 0.1$ and $\alpha_{\max} = 0.3$ to train our model. We set $N_{\min} = 8192$ and $N_{\max} = 15360$ to limit the**
number of voxels.

378 4.2 VAE RECONSTRUCTION EVALUATION
379380
381 Table 1: Comparison of Different Methods
382

Method	Input Type	Preprocess	Network Backbone	Texture	Max Resolution
TRELLIS	Visible Only	No	Sparse Voxel	Required	256
Dora	Watertight Only	Yes	VecSet	No	256
Direct3D-S2	Watertight Only	Yes	Sparse Voxel	No	1024
TripoSF	Inside support	No	Sparse Voxel	No	1024
Ours	Inside support	No	Sparse Voxel	No	1024

383
384 As shown in Table 1, although Dora-VAE, TRELLIS, and Direct3d-S2 all provide VAE weights, they
385 can only reconstruct watertight models. Therefore, it is unfair to directly compare them with our
386 method on raw meshes. We specifically show the comparison with these methods in Section A of the
387 appendix. In the main page, we primarily compare with TripoSF, as currently only TripoSF supports
388 raw mesh reconstruction. Our method demonstrates superior VAE reconstruction performance with
389 quantitative results detailed in Table 2. Our model achieves better metrics than TripoSF at the
390 same resolution in terms of the L2 norm of Chamfer Distance (CD) and F-score with a threshold
391 0.005. Since even the Level 4 data in the Dora Benchmarks lack sufficient detail, we selected several
392 detail-rich models from online websites to further test our method’s ability to capture model details,
393 as shown in Figure 11. Our method demonstrates results comparable to TripoSF’s 1024 resolution
394 output. In some high-fidelity models, our 512 resolution result captures geometry details better than
395 TripoSF’s 1024 resolution result, as shown in Figure 11.
396

402 Table 2: We sample 100K point cloud to measure the Chamfer Distance and F-score on the Dora
403 Benchmark Chen et al. (2025) in different geometry details Levels. Low Chamfer Distance ensures
404 overall shape fidelity, while high F-score ensures local details are accurately covered within an
405 acceptable error radius.
406

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

425 4.3 IMAGE-TO-3D GENERATION
426

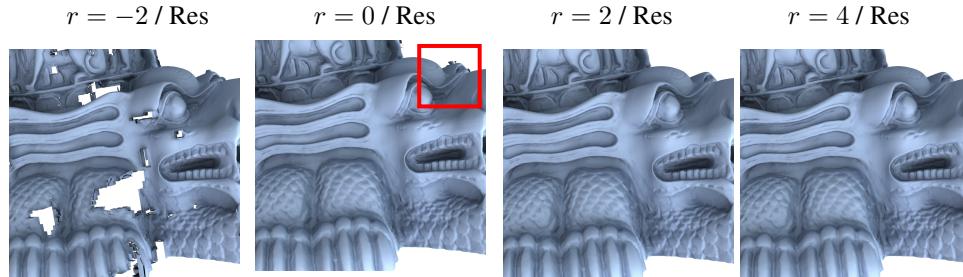
427 We further validate our VAE’s utility as a generative foundation model. Visualizations in Figure 7,
428 including image-to-3D results from in-the-wild images, highlight the generalization of our method.
429 The generated 3D shapes maintain sharp edges and rich details while exhibiting high fidelity to the
430 corresponding input images.
431

432 Table 3: Comparison of training efficiency with 512^3 resolution in A100 GPU.
433

	w/o Carving & Zooming	w/o Zooming	Ours ($\alpha_{min} = 0.3$)	Ours ($\alpha_{min} = 0.6$)	Ours ($\alpha_{min} = 0.15$)
Training Speed (steps/h)	2652	5337	8640	7590	9238
GPU Memory Peak (GB)	64	50	32	35	28

438
439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485

444 **Depth-based voxel carving and adaptive zooming.** In this paper, we employ depth-based carving
445 and adaptive zooming to replace the Frustum-aware Sectional method used by TripoSF. This sub-
446 stitution allows for a more reasonable voxel cropping operation. We further find that this cropping
447 operation can be directly applied in the latent space, which further reduces the computational cost.
448 We present the impact of each component on the training speed and memory footprint in Table 3.
449 It demonstrates that performing voxel carving based on visibility in the latent space significantly
450 reduces training cost. Simultaneously, we can further adjust the computational cost by tuning α_{max} .
451 However, considering that the encoder still needs to encode global information, an excessively low
452 α_{max} does not provide a linear reduction in cost. Furthermore, because the encoder still encode
453 global information, our method cannot be directly scaled up to train at 1536^3 resolution. One ap-
454 proach is to increase the upsampling rate from 4 to 8, thereby reducing the computational pressure on
455 the encoder. Additionally, exploring the possibility of performing voxel culling within the encoder
456 would also be an interesting direction.

464 Figure 4: Since the centers of some voxels may lie behind the surface, we use $r = 2 / \text{resolution}$ as
465 the depth-carving threshold to avoid missing these voxels. A smaller r can lead to geometric errors,
466 while a larger r has almost no effect on geometry but slightly increases the number of retained
467 voxels.468 **VAE resolution.** Higher resolution always leads to better VAE reconstruction quality, as shown in
469 Table 2. We further tested the extraction results of the network weights trained at 1024^3 resolution
470 at different resolutions in Figure 5. We found that our model has the ability to complete 1536^3
471 reconstruction without being trained at the corresponding resolution. At the 1536 resolution, chain-
472 beads that were previously not captured were successfully reconstructed.

473 474 475 476 477 478 479 480 481 482 483 484 485

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 11, 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.

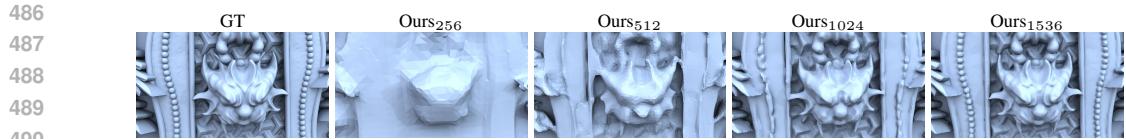


Figure 5: We used the 1024^3 resolution checkpoint to test the reconstruction results at different resolutions. Our method demonstrated generalization capabilities across different resolutions.

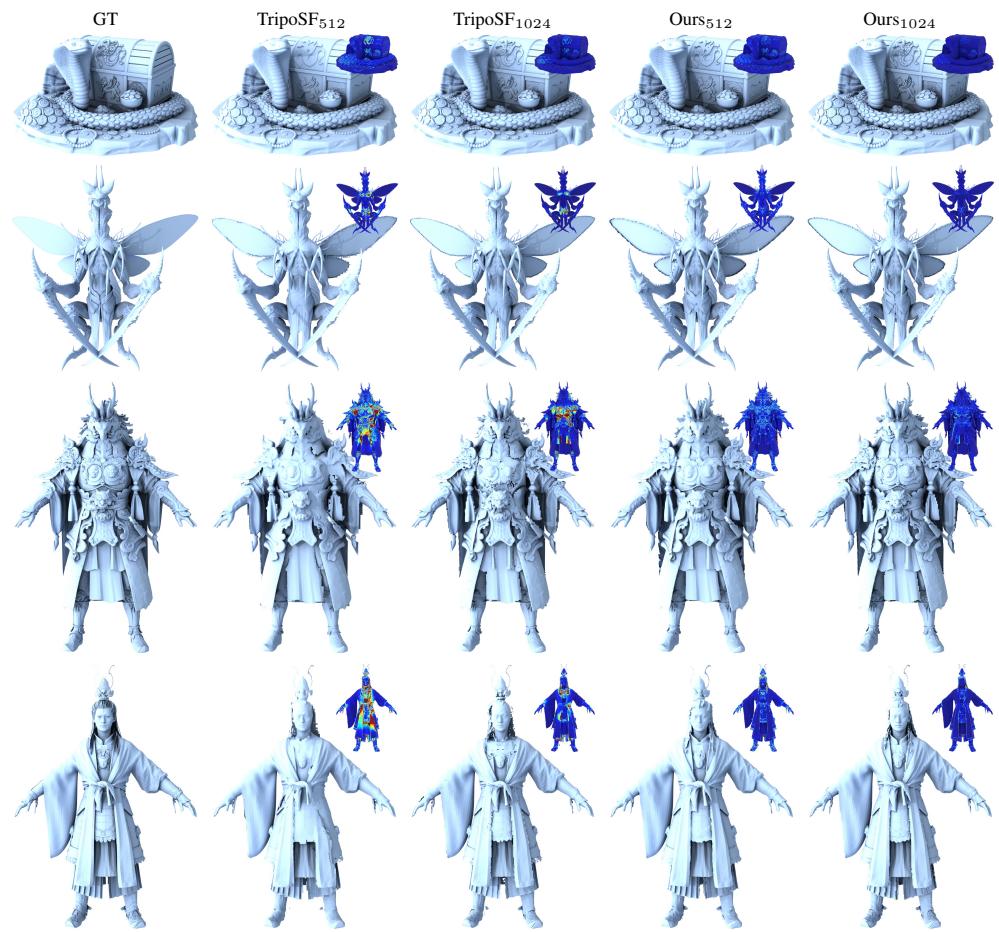


Figure 6: Qualitative comparison of VAE reconstruction between ours and TripoSF with different resolution. Our approach demonstrate superior performance in reconstructing geometry details.



Figure 7: Single image-to-3D generations with in-the-wild images which are collected from Geminis or Dora.

540

6 ETHICS STATEMENT

541
542 All datasets and models used in this study are publicly available and have been used in accordance
543 with their respective licenses and terms of use.
544545

7 REPRODUCIBILITY STATEMENT

546 We have clarified our experiment setting in Section 4.1. We will open-source the code and release
547 the trained model.
549550

REFERENCES

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

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

560 Rui Chen, Jianfeng Zhang, Yixun Liang, Guan Luo, Weiyu Li, Jiarui Liu, Xiu Li, Xiaoxiao Long,
561 Jiashi Feng, and Ping Tan. Dora: Sampling and benchmarking for 3d shape variational auto-
562 encoders. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*,
563 pp. 16251–16261, June 2025.

564 Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit
565 function learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, December
566 2020.

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

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

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

576 Zekun Hao, David W Romero, Tsung-Yi Lin, and Ming-Yu Liu. Meshtron: High-fidelity, artist-like
577 3d mesh generation at scale. *arXiv preprint arXiv:2412.09548*, 2024.

578 Xianglong He, Zi-Xin Zou, Chia-Hao Chen, Yuan-Chen Guo, Ding Liang, Chun Yuan, Wanli
579 Ouyang, Yan-Pei Cao, and Yangguang Li. Sparseflex: High-resolution and arbitrary-topology
580 3d shape modeling. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

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

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

585 Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*,
586 32(3), 2013.

587 Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Pro-
588 ceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP ’06, pp. 61–70,
589 2006.

594 Zhihao Li, Yufei Wang, Heliang Zheng, Yihao Luo, and Bihan Wen. Sparc3d: Sparse representation
 595 and construction for high-resolution 3d shapes modeling. *arXiv preprint arXiv:2505.14521*, 2025.
 596

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

600 Yanchao Liu, Jianwei Guo, Bedrich Benes, Oliver Deussen, Xiaopeng Zhang, and Hui Huang.
 601 Treepartnet: neural decomposition of point clouds for 3d tree reconstruction. *ACM Trans. Graph.*,
 602 40(6), December 2021.

603

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

607 Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Pro-
 608 ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.
 609 2837–2845, 2021.

610

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

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

615

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

619

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

625

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

629

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

633

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

637

638 Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan
 639 Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-
 640 based mesh optimization. *ACM Trans. Graph.*, 42(4), jul 2023. ISSN 0730-0301.

641

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

646

647 Tencent Hunyuan3D Team. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured
 648 3d assets generation, 2025.

649

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

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

651

652 Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen,
653 Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In
654 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21469–21480,
655 2025.

656

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

660

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

664

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

668

669 Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan
670 Xu, and Jingyi Yu. Clay: A controllable large-scale generative model for creating high-quality 3d
671 assets. *ACM Transactions on Graphics (TOG)*, 43(4):1–20, 2024.

672

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

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

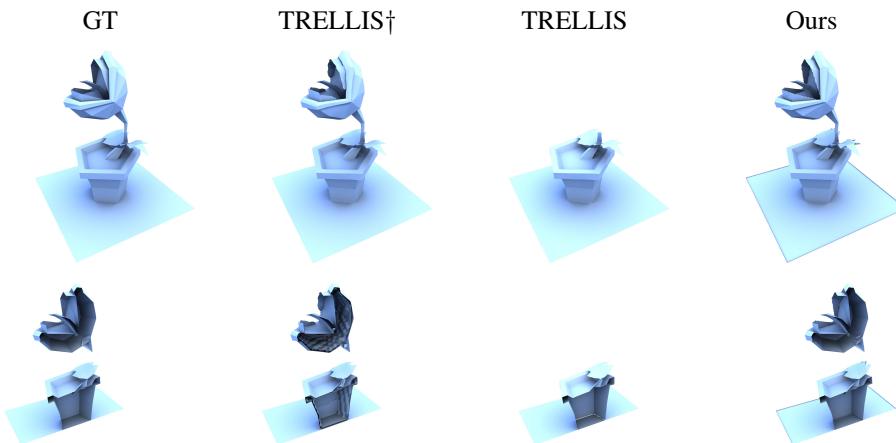
697

698

699

700

701

702 A COMPARISION WITH MORE METHODS
703704 We sample 100K points to measure the Chamfer Distance (CD) and F-score on the Dora Benchmark
705 Chen et al. (2025) across different geometry details Levels. Comparison is conducted within each
706 group as shown in Table 4.
707708 **Visible Only.** TRELLIS uses image features as input, and thus can only achieve better reconstruction
709 for the visible regions. Furthermore, since it densifies the network’s sparse output using grid points
710 with an SDF of 1. Such operation generates a redundant layer of faces on the inner side of the
711 object, as shown in Figure 8. Although TRELLIS proposed a post-processing method to remove
712 these internal faces, it may lead to incorrect deletions, which could actually worsen the metrics.
713 To avoid misunderstanding and potentially unfair comparisons, we use TRELLIS’s visibility-based
714 point cloud sampling results to calculate the CD and F-score.
715716 **Watertight Only.** Sparc3d, Dora, and Direct3d all require the input mesh to be a watertight mesh to
717 compute the SDF. Dora uses the ϵ -level set of the raw mesh as the watertight representation of the
718 original model. Sparc3d proposed a method based on flood fill and optimization to further reduce
719 the loss of precision caused by this conversion, but this introduces self-intersections in the output.
720 Since Sparc3d has not open-sourced any code, we use Dora’s method for data processing: we set
721 $\epsilon = 2/\text{resolution}$ for each resolution to extract the surface. This processed data is then used as the
722 GT mesh and input to each method to obtain the reconstruction results. Since the ϵ -level set method
723 results in the loss of object surface details, and this loss is particularly severe at lower resolutions
724 (higher ϵ), our method achieves the best results only at Level 4 on the Watertight remesh results.
725 Furthermore, at Level 1, the low-resolution results appear better than the high-resolution results due
726 to the sheer lack of details that require high resolution to capture.
727728 **Raw mesh.** The comparison on raw meshes best reflects a method’s ability to reconstruct geometry.
729 However, as mentioned earlier, methods other than ours and TripoSF’s cannot be directly applied
730 to raw meshes, which causes the metrics for these methods to fall significantly behind. To avoid
731 unnecessary misunderstanding, we mark these methods with a asterisk (*). Although we do not
732 consider internal viewpoints during training, we do not filter any voxels during inference, allowing
733 our method to still generate objects with internal structures, as shown in Figure 9.
734748 Figure 8: The inner redundant faces generated by TRELLIS.
749
750751 B FAIL CASE
752753 3D VAEs that rely on voxelized point cloud features inherently struggle to capture thin structures
754 whose thickness is smaller than one voxel. As a result, such structures may be omitted during
755 reconstruction. As illustrated in Figure 10, both our method and TripoSF fail to recover extremely
756

756
 757 Table 4: We sample 100K point cloud to measure the Chamfer Distance and F-score on the Dora
 758 Benchmark Chen et al. (2025) in different geometry details Levels. Comparison is conducted within
 759 each group (Visible Only, Watertight and Raw Mesh respectively).

760 761 762 Type	763 764 765 766 Method	Chamfer Distance (10^{-5}) ↓							
		767 768 L1		769 770 L2		771 772 L3		773 774 L4	
		775 Mean	776 Std	777 Mean	778 Std	779 Mean	780 Std	781 Mean	782 Std
Visible only	TRELLIS	1.633	1.088	2.098	2.095	5.300	68.56	4.882	11.16
	TRELLIS [†]	1.571	0.933	1.895	1.117	2.391	1.369	3.253	4.778
Watertight remeshing	Dora	1.886	0.771	2.182	0.832	2.753	1.487	3.217	1.278
	Direct3d-S2 ₁₀₂₄	1.128	0.946	1.402	2.181	1.640	0.893	2.308	1.123
	TripoSF ₅₁₂	1.232	0.812	1.304	0.763	1.838	1.053	2.517	1.322
	Ours ₅₁₂	1.218	0.810	1.305	0.765	1.840	1.049	2.518	1.334
	TripoSF ₁₀₂₄	1.243	0.825	1.321	0.741	1.844	1.007	2.426	1.184
	Ours ₁₀₂₄	1.242	0.824	1.324	0.741	1.785	0.898	2.258	0.906
Raw mesh	TRELLIS*	266.8	2270	23.00	92.21	35.13	213.2	90.26	867.3
	Dora*	300.5	1808	225.7	1512	157.7	934.1	200.9	1447
	Direct3d-S2 ₁₀₂₄ *	335.3	1725	363.8	2161	415.2	3224	425.4	2336
	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 ↑									
783 784 Type	785 786 787 Method	788 789 L1		790 791 L2		792 793 L3		794 795 L4	
		796 Mean	797 Std	798 Mean	799 Std	800 Mean	801 Std	802 Mean	803 Std
Visible only	TRELLIS	0.946	0.075	0.928	0.078	0.897	0.080	0.861	0.074
	TRELLIS [†]	0.945	0.076	0.928	0.078	0.896	0.081	0.862	0.074
Watertight remeshing	Dora	0.959	0.069	0.937	0.081	0.887	0.108	0.838	0.117
	Direct3d-S2 ₁₀₂₄	0.974	0.048	0.969	0.055	0.941	0.074	0.887	0.106
	TripoSF ₅₁₂	0.964	0.061	0.963	0.061	0.920	0.094	0.862	0.121
	Ours ₅₁₂	0.965	0.061	0.963	0.061	0.920	0.093	0.862	0.122
	TripoSF ₁₀₂₄	0.963	0.062	0.963	0.059	0.922	0.089	0.874	0.112
	Ours ₁₀₂₄	0.963	0.062	0.962	0.059	0.927	0.079	0.889	0.087
Raw mesh	TRELLIS*	0.430	0.221	0.719	0.149	0.665	0.150	0.646	0.189
	Dora*	0.379	0.077	0.408	0.079	0.337	0.068	0.292	0.084
	Direct3d-S2 ₁₀₂₄ *	0.370	0.361	0.339	0.277	0.240	0.175	0.269	0.166
	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

thin geometric elements. Handling these sub-voxel structures remains an open challenge for voxel-based representations and is an important direction for future work.

810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828

Figure 9: Our method is able to produce shapes with inner structures.

830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845

846
847
848
849
850
851
852
853
854

855
856
857
858
859
860
861
862
863

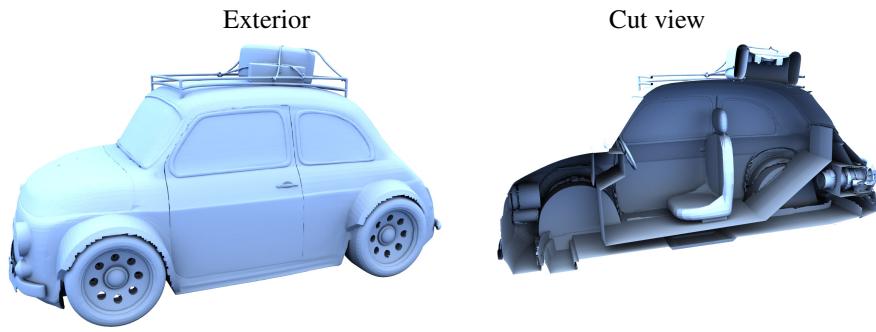


Figure 9: Our method is able to produce shapes with inner structures.

830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845

846
847
848
849
850
851
852
853
854

855
856
857
858
859
860
861
862
863

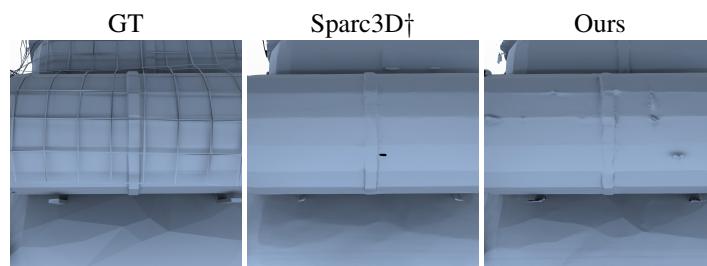


Figure 10: Fail case. Neither our method nor TripoSF can reconstruct extremely thin structures.



Figure 11: More qualitative comparison of VAE reconstruction between ours, TripoSF and Direct3d-S2 with 1024^3 resolution. Our approach demonstrate superior performance in reconstructing geometry details.