SparseFlex: High-Resolution and Arbitrary-Topology 3D Shape Modeling

Xianglong He[1,2⋆] Zi-Xin Zou[2⋆] Chia-Hao Chen[1,2] Yuan-Chen Guo[2] Ding Liang[2]
Chun Yuan[1†] Wanli Ouyang[3] Yan-Pei Cao[2] Yangguang Li[2†]
[1]Tsinghua University [2]VAST [3]The Chinese University of Hong Kong

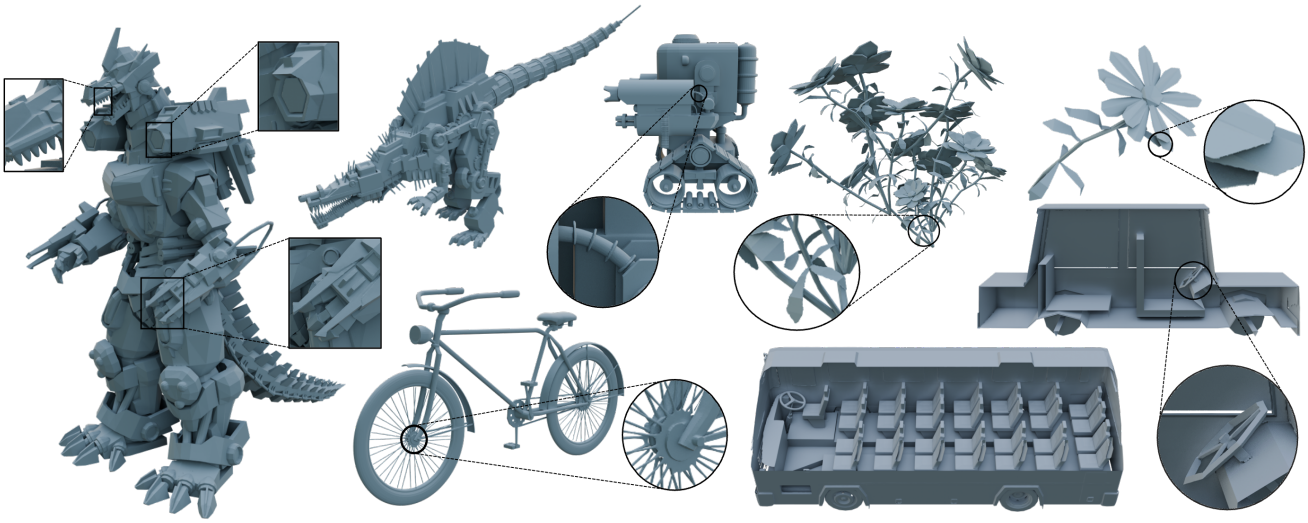

Figure 1. **SparseFlex VAE achieves high-fidelity reconstruction and generalization from point clouds.** Benefiting from a sparse-structured differentiable isosurface surface representation and an efficient frustum-aware sectional voxel training strategy, our SparseFlex VAE demonstrates the state-of-the-art performance on complex geometries (left), open surfaces (top right), and even interior structures (bottom right), facilitating the high-quality image-to-3D generation with arbitrary topology.

Abstract

Creating high-fidelity 3D meshes with arbitrary topology, including open surfaces and complex interiors, remains a significant challenge. Existing implicit field methods often require costly and detail-degrading watertight conversion, while other approaches struggle with high resolutions. This paper introduces SparseFlex, a novel sparse-structured isosurface representation that enables differentiable mesh reconstruction at resolutions up to 1024^3 *directly from rendering losses. SparseFlex combines the accuracy of Flexicubes with a sparse voxel structure, focusing computation on surface-adjacent regions and efficiently handling open surfaces. Crucially, we introduce a frustum-aware sectional voxel training strategy that activates only relevant voxels during rendering, dramatically reducing memory consumption and enabling high-resolution training. This also allows, for the first time, the reconstruction of mesh interiors using only rendering supervision. Building upon this, we demonstrate a complete shape modeling pipeline by training a variational autoencoder (VAE) and a rectified flow transformer for high-quality 3D shape generation. Our experiments show state-of-the-art reconstruction accuracy, with a ∼82% reduction in Chamfer Distance and a ∼88% increase in F-score compared to previous methods, and demonstrate the generation of high-resolution, detailed 3D shapes with arbitrary topology. By enabling high-resolution, differentiable mesh reconstruction and generation with rendering losses, SparseFlex significantly advances the state-of-the-art in 3D shape representation and modeling.*

1. Introduction

3D Generative AI is rapidly advancing, with applications spanning entertainment, design, and robotics. However, generating 3D content is fundamentally more challenging than 2D image or text generation due to the inherent complexity of representing and manipulating 3D geometry. Achieving *high fidelity* and supporting *arbitrary topology*—including open surfaces and complex interiors—presents particularly significant hurdles.

⋆ Equal contribution.
† Corresponding authors.

Recent progress in 3D generative models has explored various representations, including point clouds [43, 44, 49, 64], meshes [48, 60], 3DGS [18, 32, 74, 83], and implicit fields [8, 9, 34, 35, 56, 73, 74, 82, 86, 88, 90]. Implicit field representations, such as Signed Distance Functions (SDFs) and occupancy fields, have shown favorable results [31, 73, 74, 80, 82, 90]. However, creating training data for these methods typically involves a two-step process that limits their effectiveness. First, raw 3D mesh data must be converted into watertight representations [75, 86] to calculate SDF or occupancy values—a computationally expensive preprocessing step that often degrades fine details. Second, isosurfacing techniques (e.g., Marching Cubes [41], Dual Contouring [26]) are used to extract meshes from the learned continuous field, which can introduce further inaccuracies and artifacts. While Unsigned Distance Fields (UDFs) offer a potential way to model open surfaces [6, 10, 16, 38, 40, 79], they often suffer from instability and struggle to capture geometric fine details.

Rendering-based supervision offers a powerful, *differentiable* alternative for training 3D representations and 3D generative models [57, 58, 69, 70, 74, 76]. By directly comparing rendered images of a generated mesh to ground-truth data, rendering losses avoid the need for the initial watertight preprocessing step and better preserve fine details. However, a critical bottleneck arises: when used with dense implicit fields, rendering supervision requires extremely high memory consumption at high resolutions, severely limiting the achievable fidelity.

This paper introduces **SparseFlex**, a new sparse-structured isosurface representation that addresses these limitations and unlocks high-resolution, differentiable mesh reconstruction and generation using rendering supervisions. SparseFlex is built upon Flexicubes [58], providing accurate and differentiable isosurface extraction. The key design is the use of a *sparse* voxel structure instead of a conventional dense grid. This sparsity is crucial for two primary reasons: (1) it dramatically reduces memory consumption, enabling *high-resolution modeling*, and (2) it allows for the effective pruning of voxels near open boundaries, enabling the representation of *open surfaces*.

To fully leverage the capabilities of SparseFlex, we propose **frustum-aware sectional voxel training**. Inspired by techniques used in real-time rendering [1], this approach activates only the subset of SparseFlex voxels that reside within the camera's viewing frustum during each training iteration. We also introduce an adaptive strategy to control the frustum's parameters, further optimizing memory usage. This not only substantially reduces computational and memory overhead but also enables, for the first time, the reconstruction of mesh interiors using only rendering supervision by appropriately positioning the camera. Fig. 3 illustrates our SparseFlex representation and the efficient sectional voxel training strategy.

Building on SparseFlex and our frustum-aware training, we present a complete 3D shape modeling pipeline. We employ a variational autoencoder (VAE) architecture, drawing inspiration from TRELLIS [74] but with key modifications. Firstly, because our focus is on *high-fidelity geometry*, we use point clouds as input to the VAE, providing a direct and detailed representation of the shape's surface. Furthermore, we introduce a *self-pruning upsampling* module within the decoder to further refine the sparse voxel structure, which is particularly beneficial for representing open surfaces. A rectified flow transformer is then trained on the learned latent space for high-quality, image-conditioned 3D shape generation. Through extensive experiments on Toy4k [61], ABO [11], GSO [15], Meta [45], Objaverse [14], and Deepfashion3D [19], our method demonstrates state-of-the-art shape reconstruction accuracy with minimal detail degradation, and high-quality single-image 3D shape generation.

Our main contributions are:

- We propose SparseFlex, a new sparse-structured isosurface representation enabling efficient, high-resolution, and differentiable 3D shape modeling, with natural handling of open surfaces.
- We introduce a novel sectional voxel training strategy with adaptive view frustum control, dramatically reducing memory consumption and enabling high-resolution mesh reconstruction and generation, including interiors, using rendering losses.
- We demonstrate state-of-the-art reconstruction accuracy and the generation of high-resolution, detailed 3D shapes with arbitrary topology, representing a significant advance in the field.

2. Related Work

2.1. 3D Shape Representations for Generation

Point Cloud. Point clouds are a flexible 3D representation and can be easily acquired using depth sensors and LiDAR. Research has focused on point cloud processing [54, 55, 87] and generation [43, 44, 49, 64], with recent approaches treating point clouds as a distribution, directly sampling point clouds from noise via generative models [4, 84]. However, due to their limitation in representing solid surfaces, an additional surface reconstruction step [22, 27, 53] is required.

Triangle Mesh. Triangle meshes are the primary representation for 3D assets in modern industrial pipelines. Recent works employ auto-regressive models to generate triangle faces sequentially [48, 60, 63], improving resemblance to artist-created meshes. While effective for low-poly meshes, these methods struggle to produce high-quality meshes with a high face count.

Implicit Field. Implicit fields (SDF or occupancy) are widely used in geometry learning, particularly for 3D recon-

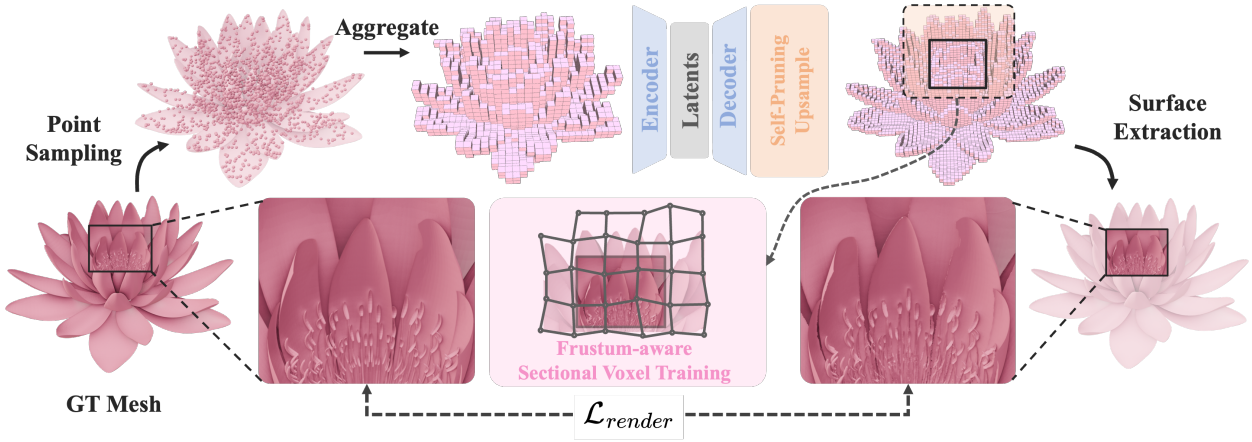

Figure 2. **Overview of the SparseFlex VAE pipeline.** SparseFlex VAE takes point clouds sampled from a mesh as input, voxelizes them, and aggregates their features into each voxel. A sparse transformer encoder-decoder compresses the structured feature into a more compact latent space, followed by a self-pruning upsampling for higher resolution. Finally, the structured features are decoded to SparseFlex through a linear layer. Using the frustum-aware section voxel training strategy, we can train the entire pipeline more efficiently by rendering loss.

struction [16, 22, 52, 58, 67, 91] and generation [8, 9, 34, 35, 56, 73, 74, 82, 86, 88, 90], producing high-quality meshes. Enhancements with triplane [73], vector set [80, 82, 88] sparse voxels [74] and sparse hierarchical voxels [56] allow neural networks to decode the field more effectively. These methods typically use the isosurface techniques [26, 41, 50] for surface extraction but struggle with open-surface shapes like clothing and flowers. Additionally, many require a time-consuming watertight conversion that degrades details, while rendering supervised approaches [74] suffer from high memory consumption when training at high resolutions.

Open Surfaces. Meshes with open surfaces are common but remain challenging to process. The Unsigned Distance Field (UDF) is widely used for open-surface modeling from point clouds [10] or multiview images [38, 40]. Surf-D [79] introduces a UDF-based diffusion model for generating meshes with arbitrary topology. However, UDF-based shape modeling is more difficult than SDF, and its surface extraction [17] is prone to instability due to inaccuracy in neural network gradient estimations. As a result, achieving high-quality open-surface meshes remain challenging. 3PSDF [6] introduces a three-pole sign to distinguish surface-adjacent regions but relies on binary occupancy grids for surface extraction, leading to discontinuities and artifacts on the surface. In this paper, we incorporate sparse structure into Flexicubes [58] to efficient high-quality isosurface representation for open faces.

2.2. 3D Generative Models and VAE

Existing 3D generation studies are primarily classified into two categories: large 3D reconstruction model following 2D multiview diffusion models and native 3D generation models. The first category uses multiview diffusion models to generate multiview images from text or an image [23, 24, 37, 59, 66], followed by a large 3D reconstruction model [21, 33, 62, 76, 77, 85, 92] is employed to reconstruct the 3D representation from these images in seconds. However, inconsistencies between generated multiview images significantly often degrade the result quality. The second category focuses on native 3D generation model that directly generate 3D models through generative models, including GAN [3, 7, 72, 89], auto-regressive [47, 81], diffusion [25, 64, 82, 86] and rectified flow [35, 74].

Due to the diverse and non-compact nature of 3D representation, many approaches use Variational Auto-Encoder (VAE) [29] or Vector Quantized VAE (VQ-VAE) [65] to encode 3D shapes into latent spaces for the generative models. Geometry-focused methods often input point clouds uniformly sampled from mesh surfaces into the VAE [5, 82]. When aim at decoding both geometry and texture, some methods [31, 32, 74] encode multiview image features into latent spaces. This two-stage process makes the VAE's reconstruction quality crucial for subsequent generation performance. Some works [5, 80] have improved generation quality by enhancing the VAE's shape encoding-decoding capabilities. In this paper, we aim to develop a foundational VAE that encodes 3D shapes into latent spaces and reconstructs them with arbitrary topology while preserving the raw 3D shape's details.

3. Method

We present a method for high-resolution 3D shape modeling based on a novel sparse-structured isosurface representation, SparseFlex. Our approach leverages differentiable rendering for training, enabling accurate reconstruction of complex geometries, including open surfaces and interiors (see Fig. 1. Fig. 2 illustrates our variational autoencoder (VAE) based shape modeling pipeline, which utilizes the SparseFlex representation to learn a compact latent space of 3D shapes. The VAE decoder outputs the parameters of a SparseFlex instance, facilitating high-resolution mesh reconstruction. A key component of our approach is *frustum-aware sectional*

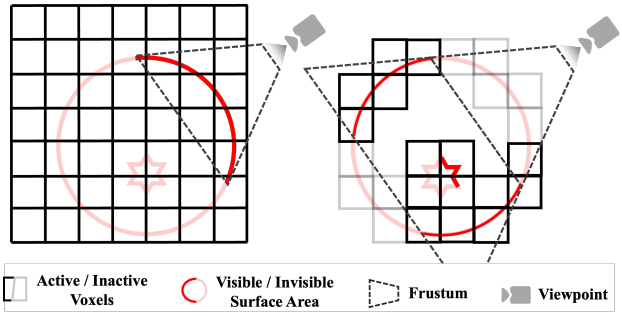

Figure 3. **Frustum-aware sectional voxel training.** The previous mesh-based rendering training strategy (left) requires activating the entire dense grid to extract the mesh surface, even though only a few voxels are necessary during rendering. In contrast, our approach (right) adaptively activates the relevant voxels and enables the reconstruction of mesh interiors only using rendering supervision.

voxel training, which significantly reduces memory consumption, allowing for training at resolutions up to 1024^3. Details of the SparseFlex representation are provided in Sec. 3.1, followed by a description of the VAE architecture in Sec. 3.2 and the training procedure in Sec. 3.3. Sec. 3.4 describes image-conditioned 3D shape generation using a rectified flow transformer on the learned latent space.

3.1. SparseFlex Representation

Preliminary. To achieve differentiable mesh extraction while preserving sharp features, we build upon Flexicubes [58], a method based on Dual Marching Cubes [50] (DMC). DMC places vertices at the center of voxels rather than edges/corners, leading to better feature preservation. Flexicubes constructs a *dense* voxel with a resolution of N_r^3, where an SDF scalar grid $s \in \mathbb{R}^{N_g^3}$ is assigned to the voxel corner points, where $N_g = N_r + 1$. For each voxel, it incorporates interpolation weights $\alpha \in \mathbb{R}_{>0}^{N_r^3 \times 8}, \beta \in \mathbb{R}_{>0}^{N_r^3 \times 12}$ to each voxel cell, and deformation vectors $\delta \in \mathbb{R}^{N_g^3}$ to each SDF grid. The underlying surface mesh can be effectively optimized via differentiable rendering [30].

Sparse Structured Flexicubes. The core design of SparseFlex is the introduction of a *sparse* voxel structure, enabling high-resolution shape representation while drastically reducing memory consumption. Instead of a dense grid, SparseFlex represents a shape using a significantly smaller set of voxels, $\mathcal{V}$, concentrated near the surface. This sparsity is crucial for two reasons: (1) it allows us to achieve much higher resolutions than would be possible with a dense grid, and (2) it enables the natural representation of open surfaces by simply omitting voxels in empty regions.

Specifically, the SparseFlex is defined by a set of N_v voxels, $\mathcal{V} = \{v_i = (x_i, y_i, z_i)\}$, where v_i represents the 3D coordinates of the *center* of the i-th voxel. Let N_c be the number of corner grids associated with these voxels, where $N_v = |\mathcal{V}|$. Each voxel is associated with interpolation weights $\{\alpha_i \in \mathbb{R}_{>0}^8, \beta_i \in \mathbb{R}_{>0}^{12} | 0 \leq i < N_v\}$. Each corner grid is associated with an SDF value $\{s_j | 0 \leq j < N_c\}$ and deformation vectors $\{\delta_j | 0 \leq j < N_c\}$. Due to the sparsity, $N_v \ll N_r^3$ and $N_c \ll N_g^3$, representing a significant reduction in memory usage compared to the dense Flexicubes representation. We only apply Dual Marching Cubes on these sparse voxels to extract the underlying surface. Formally, the SparseFlex representation, $\mathcal{S}$, is defined as:

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

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

SparseFlex inherits the differentiability of Flexicubes, allowing for end-to-end optimization using rendering losses. This eliminates the need for watertight mesh pre-processing, preserving fine details. Furthermore, the sparse structure, combined with the continuous and deformable nature of the SDF, allows for accurate and efficient representation of high-quality open-surface meshes. The sparsity also paves the way for our efficient frustum-aware training strategy, described in Sec. 3.3.

3.2. SparseFlex VAE for Shape Modeling

To learn a compact and disentangled latent space of 3D shapes, we employ a variational autoencoder (VAE) [29] that utilizes the SparseFlex representation. A VAE learns a probabilistic mapping between an input space (in our case, 3D shapes represented as point clouds) and a lower-dimensional latent space, enabling both reconstruction and generation of shapes. Fig. 2 also illustrates our VAE architecture. Our architecture draws inspiration from TRELLIS [74], but with key modifications to leverage the strengths of SparseFlex.

Encoder. The input to our encoder is a point cloud $\mathcal{P} = \{p_i \in \mathbb{R}^3\}_{i=1}^{N_p}$, uniformly sampled from the surface of a 3D mesh, along with corresponding normals $\mathcal{N} = \{n_i \in \mathbb{R}^3\}_{i=1}^{N_p}$. We first voxelize the point cloud to obtain the sparse structure $\mathcal{V}$ of the SparseFlex representation $\mathcal{S}$. We then employ a shallow PointNet [54] to aggregate local geometric features *within each voxel*. Specifically, for each voxel $v_i \in \mathcal{V}$, we apply a local max-pooling operation [52] to the points contained within that voxel, producing a feature vector f_i. These voxel features $\mathcal{F} = \{f_i\}$, along with the sparse structure $\mathcal{V}$, are then fed into a sparse transformer backbone. This backbone utilizes shifted window attention [39, 78], similar to TRELLIS [74], but is adapted to operate directly on the sparse voxel features $\mathcal{F}$ and structure $\mathcal{V}$. The transformer outputs a latent code $z \in \mathbb{R}^{d_z}$, which represents the encoded 3D shape.

Decoder. The decoder takes the latent code z as input and predicts the parameters of a SparseFlex instance, $\mathcal{S} = (\mathcal{V}, \mathcal{F}_c, \mathcal{F}_v)$. We use a series of transformer layers, culminating in a final linear layer, to predict the SDF values (s_j) and deformations (δ_j) for each corner grid, as well as the interpolation weights (α_i, β_i) for each voxel.

Upsampling Modules. To achieve high-resolution reconstructions, we incorporate two convolutional, self-pruning upsampling modules within the decoder, following the transformer. These modules progressively increase the resolution of the SparseFlex representation. Each upsampling module *subdivides* existing voxels into smaller voxels (increasing the resolution by a factor of 4). Crucially, each module also $prunes$ redundant voxels based on a predicted occupancy value. A voxel is considered occupied if, after the subdivision, it contains any points from the input point cloud $\mathcal{P}$. This pruning process, inspired by [56], is essential for maintaining the sparsity of the SparseFlex representation and is particularly beneficial for accurately representing open surfaces, as it removes unnecessary voxels in empty regions.

3.3. Training SparseFlex VAE

We train our SparseFlex VAE end-to-end using rendering losses, leveraging the differentiability of the SparseFlex representation and a new training strategy called *frustum-aware sectional voxel training*. This strategy dramatically reduces memory consumption during training, enabling us to achieve high resolutions (up to 1024^3) that would be infeasible with traditional approaches.

Frustum-aware Sectional Voxel Training. Even with the sparse structure of SparseFlex, directly rendering the entire representation at high resolutions can be computationally expensive. Furthermore, standard rendering supervision typically focuses only on the visible surface, neglecting the interior of the shape. Besides, recent methods [36, 69, 71, 74, 76] relying on rendering supervision from mesh typically require extracting the entire mesh because a dense representation doesn't trivially allow for partial extraction. In contrast, our sparse representation naturally enables partial extraction. To address these issues, we introduce frustum-aware sectional voxel training. Inspired by techniques used in real-time rendering for efficient visibility culling [1], this approach activates only the voxels within the camera's viewing frustum during each training iteration. *"Activating"* a voxel means including it in the isosurface extraction and rendering process. This *"sectional"* approach means we only process a *portion* of the 3D space at a time, markedly reducing memory usage.

As illustrated in Fig. 3, given the camera's extrinsic π, intrinsics K, and the near (n) and far (f) clipping planes of the viewing frustum, we compute the Model-View-Projection (MVP) matrix $\mathbf{MVP}$. We then use a boolean operator to check whether the center of each voxel v_i lies within the viewing frustum defined by the MVP matrix. We use $I(v_i \in \text{Frustum}(\mathbf{MVP}))$ to represent this check, where $I(\cdot)$ is an indicator function. The set of active voxels, $\mathcal{V}_{\text{active}}$, is then defined as:

$$\mathcal{V}_{\text{active}} = \{v_i | I(v_i \in \text{Frustum}(\mathbf{MVP})) = 1, v_i \in \mathcal{V}\}. \quad (2)$$

Adaptive Frustum and Interior Reconstruction. We introduce a visibility ratio α ($0 < \alpha \leq 1$) controlling the *proportion* of active voxels in SparseFlex. We adaptively adjust the near and far clipping planes to ensure that approximately αN_v voxels are within the frustum. This is achieved through an iterative process: we initially set the near and far planes and iteratively adjust them based on the number of active voxels until the desired proportion is reached.

This adaptive frustum also enables a novel capability: reconstructing mesh *interiors* using only rendering supervision. By positioning a virtual camera *inside* the object or adjusting the near clipping plane to intersect the mesh, we can render and supervise the internal structure (see Fig. 4). Moreover, the zoom-in camera viewpoint can render the mesh surface with greater details, making it better for higher-resolution training. This is a significant advantage over methods that rely on watertight representations, which cannot capture interior details.

Loss Function. We train our VAE in an end-to-end manner, with an objective function comprising four components:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{render}} + \lambda_2 \mathcal{L}_{\text{prune}} + \lambda_3 \mathcal{L}_{\text{KL}} + \lambda_4 \mathcal{L}_{\text{flex}}, \quad (3)$$

where λ_1, λ_2, λ_3, and λ_4 are weighting coefficients that balance the different loss terms.

$\mathcal{L}_{\text{render}}$ is the rendering supervision loss. We employ a combination of losses commonly used in differentiable rendering [28, 46]:

$$\mathcal{L}_{\text{render}} = \lambda_d \mathcal{L}_d + \lambda_n \mathcal{L}_n + \lambda_m \mathcal{L}_m + \lambda_{ss} \mathcal{L}_{ss} + \lambda_{lp} \mathcal{L}_{lp}, \quad (4)$$

where $\mathcal{L}_d$, $\mathcal{L}_n$, and $\mathcal{L}_m$ denote the L1 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 on normal maps. Please refer to the supplementary material for a detailed definition of these losses and their weighting coefficients.

$\mathcal{L}_{\text{prune}}$ is the structure loss, formulated as a binary cross-entropy (BCE) loss, supervising the construction of sparse voxels:

$$\mathcal{L}_{\text{prune}} = \text{BCE}\left(V, \hat{V}\right), \quad (5)$$

where V represents the ground-truth occupancy of voxels derived from the input point cloud and $\hat{V}$ represents the predicted occupancy by the upsampling modules.

$\mathcal{L}_{\text{KL}}$ is the KL divergence between the learned latent distribution and a standard normal prior, regularizing the latent space. $\mathcal{L}_{\text{flex}}$ is the regularization term from Flexicubes [58] to encourage smooth SDF values.

3.4. Image-to-3D Generation with Rectified Flow

Building upon the trained SparseFlex VAE, we develop a pipeline for high-quality, image-conditioned 3D shape generation, following a similar approach to TRELLIS [58]. Our approach consists of two main components: *Structure Flow Model* and *Structured Latent Flow Model*.

Method	Toys4k			Dora Benchmark		
	CD ↓	F1(0.001) ↑	F1(0.01) ↑	CD ↓	F1(0.001) ↑	F1(0.01) ↑
Craftsman [34]	13.08/4.63	10.13/15.15	56.51/85.02	13.54/2.06	6.30/11.14	73.71/91.95
Dora [5]	11.15/2.13	17.29/26.55	81.54/93.84	16.61/1.08	13.65/25.78	78.73/96.40
Trellis [74]	12.90/11.89	4.05/4.93	59.65/64.05	17.42/9.83	3.81/6.20	62.70/71.95
XCube [56]	4.35/3.14	1.61/13.49	74.65/79.62	4.74/2.37	1.31/0.84	75.64/86.50
3PSDF* [6]	4.51/3.69	11.33/14.10	81.70/86.13	7.45/1.68	7.52/12.50	79.43/91.17
Ours_{256}	2.56/1.25	18.31/27.23	85.35/92.01	1.93/0.53	16.24/28.37	88.76/97.31
Ours_{512}	1.67/0.84	23.74/34.10	90.39/95.60	1.36/0.23	21.85/36.03	91.55/98.51
Ours_{1024}	**1.33/0.60**	**25.95/35.69**	**92.30/96.22**	**0.86/0.12**	**25.71/39.50**	**94.71/99.14**

Table 1. **Quantitative comparison for VAE reconstruction quality on the Toys4K dataset (left) and Dora benchmark (right).** The '/' symbol separates the results computed over the entire dataset from those obtained exclusively on the watertight subset.

Method	CD ↓	F1(0.001) ↑	F1(0.01) ↑
Surf-D [79]	63.79	0.80	23.17
3PSDF* [6]	0.26	8.14	99.35
$\text{Ours}_{256}^{\dagger}$	0.55	6.35	94.88
Ours_{256}	0.08	18.60	99.99
$\text{Ours}_{512}^{\dagger}$	0.18	11.31	99.93
Ours_{512}	<u>0.05</u>	<u>31.60</u>	100.00
$\text{Ours}_{1024}^{\dagger}$	0.05	24.80	<u>100.00</u>
Ours_{1024}	**0.04**	**37.22**	**100.00**

Table 2. **Reconstruction results on open-surface dataset Deepfashion3D.** † indicates the absence of the self-pruning upsampling module.

Structure Flow Model. First, a separate, simple fully 3D convolutional structure VAE is employed to compress dense voxels representing 3D shapes into a low-resolution ($1/4$ scale) space. Subsequently, the image condition features are extracted using DINOv2 [51] and injected into the transformer model via cross-attention, after which a rectified flow model is trained within this low-resolution space. During inference, given an input image, the trained structure flow model generates the corresponding low-resolution 3D space, which is then decoded by the structure VAE to produce the sparse structure of the generated 3D shape.

Structured Latent Flow Model. Based on the proposed SparseFlex VAE, the point cloud and the corresponding voxelized sparse structure of a 3D shape are encoded into a *structured latent space*. Subsequently, the image condition feature obtained via DINOv2 is injected into the sparse transformer model through cross-attention, followed by training a rectified-flow model within this structured latent space. During inference, given an input image, the corresponding sparse structure is first generated using the structure flow model and its structure VAE. Then, both the sparse structure and the input image are provided to the structured latent flow model to generate the corresponding structured latent representation. Finally, the SparseFlex VAE decodes this latent representation to produce the final 3D shape.

4. Experiments

4.1. Implementation Details

We develop the implementation of SparseFlex based on the official code[1] provided by FlexiCubes [58]. We train SparseFlex VAE and structured latent flow model using approximately 400K high-quality 3D meshes filtered from large-scale datasets, Objaverse (-XL) [13, 14]. Since incorrect normals in raw data can significantly degrade the performance of both VAE reconstruction and image-to-3D generation, we apply a mesh preprocessing step to correct these flipped normals, ensuring that all normals are consistently oriented outwards. Please refer to the supplementary for more details.

Building on the success of progressive training in recent works [86], we train SparseFlex VAE progressively, increasing final resolution from low to high (256, 512, and 1024). For the structure VAE and structure flow model, we adopt the model from Trellis [74] and finetune their pre-trained weights to our task. We train SparseFlex VAE on 64 A100 GPUs with a batch size of 64 and train structured latent flow models with a batch size of 256. We use the AdamW [42] optimizer with an initial learning rate of $1e-4$ and the weight decay as 0.01. At inference, we generate the results with 3.5 CFG and 50 sampling steps.

4.2. Dataset, Baselines, and Metrics

Dataset. We evaluate the reconstruction quality of VAE across different methods on a diverse set of datasets, including 1) universal datasets ABO [11], GSO [15], Meta [45], Objaverse [14], Toys4k [61] and 2) open-surface dataset Deepfashion3D [19]). The test list of ABO, GSO, Meta, and Objaverse is derived from Dora benchmark [5] after excluding our training data which includes about 2.7k assets. For Toys4k, we use the full set following Trellis [74]. For the image-to-3D generation, we evaluate the methods on 200 random assets from Toys4k [61] and some images in the wild, showcasing the superior potential on generation tasks of SparseFlex VAE.

Baselines. We compare our VAE with previous state-of-the-art methods, including Craftsman [34], Trellis [74], Dora [5], XCubes [56], Surf-D [79] and 3PSDF [6], with Surf-D and 3PSDF specially designed for open surfaces. We directly use the available pre-trained weights provided by these baselines, except for 3PSDF, which we re-implement and train on our dataset. We only compare Surf-D on the Deepfashion3D dataset due to the lack of available pre-trained weights trained on large-scale datasets. For the generation results, we compare our method with InstantMesh [76], Direct3D [73], and TRELLIS [74].

Metrics. We evaluate the reconstruction performance of VAE by using the commonly used metrics, including Chamfer Distance (CD) and F-score with thresholds of 0.01 and 0.001. The metrics are multiplied by 10^4 and 10^2, respectively. For generation results, four orthogonal views of normal maps for each shape are rendered for quantitative comparisons. We report the Fréchet Inception Distance (FID) [20] and Kernel Inception Distance (KID) [2].

[1] https://github.com/nv-tlabs/FlexiCubes

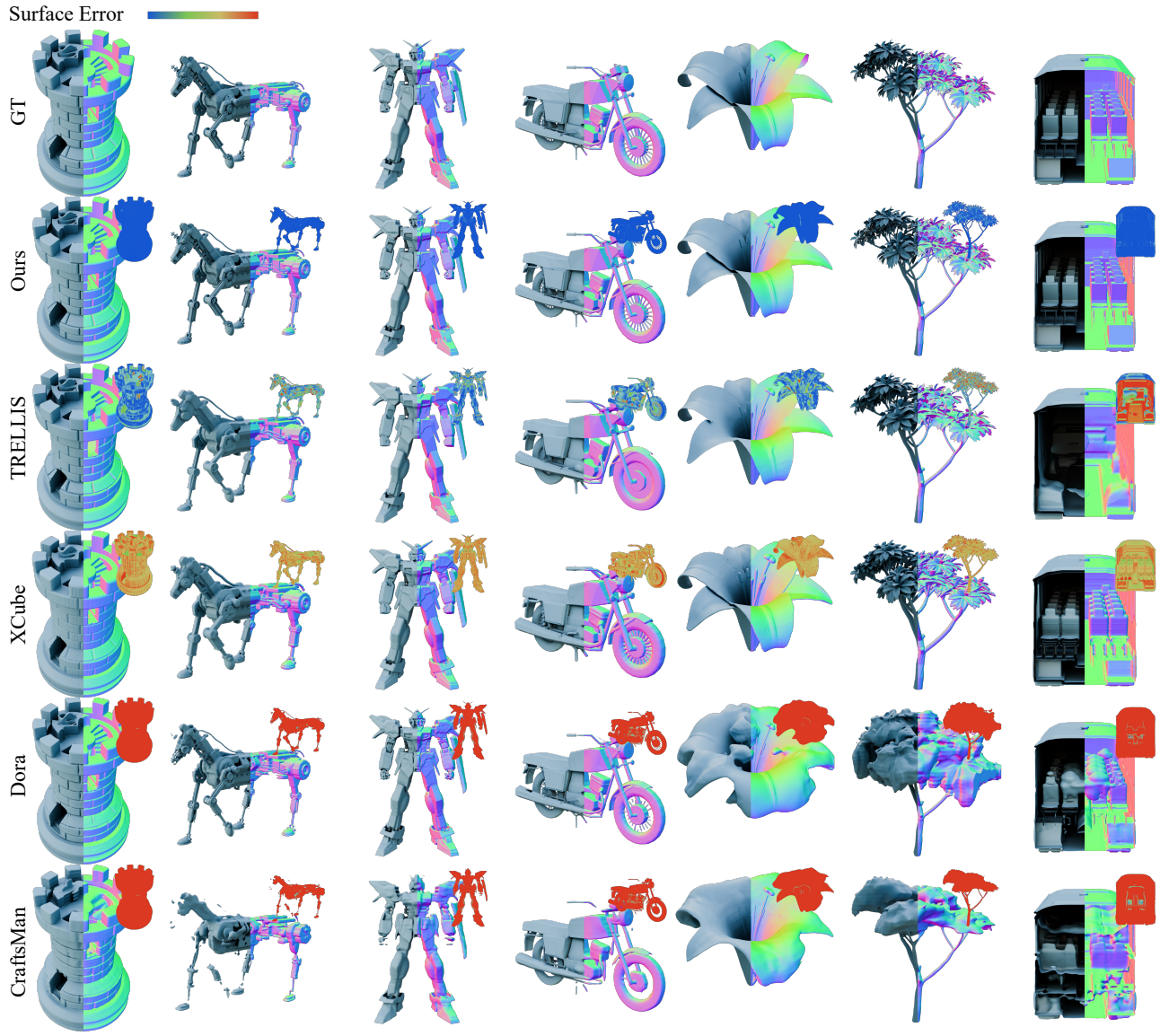

Figure 4. **Qualitative comparison of VAE reconstruction between ours and other state-of-the-art baselines.** Our approach demonstrate superior performance in reconstructing complex shapes, open surfaces, and even interior structures.

4.3. VAE Reconstruction Evaluation

We conduct extensive experiments to evaluate the quantitative results of VAE reconstruction from different methods in Table Tab. 1. Since Dora [5] and Craftsman [34] are trained on watertight data, we separate the results exclusively on the watertight subset from the results computed over the entire dataset for the clear demonstration. These methods often perform poorly on non-watertight meshes, especially open surfaces like flowers. Ours$_{256}$ already outperforms other baselines in terms of CD and F-score. As the resolution increases, our method achieves even better, ultimately achieving a $\sim$82% reduction on CD and $\sim$88% increase in F-score. Fig. 4 demonstrate the superiority of our method for complex shapes, open surfaces, and interior structures. Tab. 2 compares our method with those designed for open surfaces on the Deepfashion3D [19] dataset, and our method still achieves the best performance among them.

4.4. Image to 3D Genration

We also validate the effectiveness of our SparseFlex VAE as a foundation model for generation. Tab. 4 validates the effectiveness of our generation. Visualizations are also demonstrated in Fig. 6, which includes image-to-3D results using wild images. The generated shapes, which preserves sharp edges and fine details, highly match the given images and showcase the generalization of our method.

4.5. Ablation Studies

Self-Pruning Upsampling. Tab. 2 demonstrates that the self-pruning upsampling module plays an important role in the reconstruction quality of open-surface shapes, as it

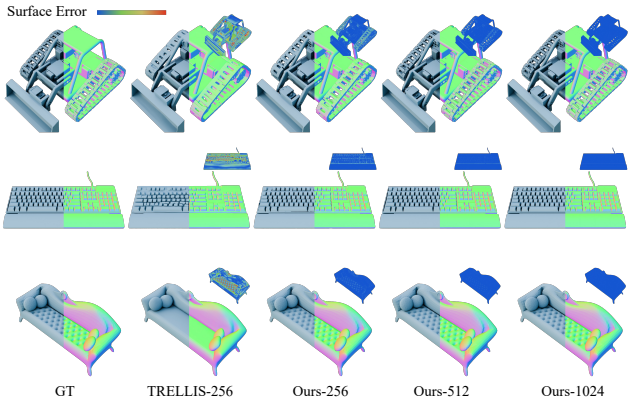

Figure 5. **Qualitative comparison of VAE reconstruction quality** between our method with different resolution and TRELLIS.

	Feed-Forward Time (ms)↓			GPU Memory Cost (MB)↓		
Resolution	256^3	512^3	1024^3	256^3	512^3	1024^3
Ours ($\alpha = 0.1$)	333	620	1151	35515	40183	55441
Ours ($\alpha = 0.3$)	357	697	1475	37403	45675	69991
w/o FSV	390	958	OOM	40703	62029	OOM
w/o FSV & Sp.	418	OOM	OOM	45505	OOM	OOM

Table 3. **Feed-Forward time and GPU memory cost comparisons.** α stands for the visibility ratio of voxels. 'OOM' means Out Of Memory and 'FSV' means frustum-aware sectional voxel training strategy. 'Sp' means SparseFlex.

Method	InstantMesh [76]	Direct3D [73]	TRELLIS [74]	Ours
FID↓	68.74	50.84	47.66	**44.95**
KID ($\times 10^3$)↓	9.68	2.04	1.28	**1.05**

Table 4. **Quantitative generation results on Toys4k.**

allows for effective pruning of voxels near open boundaries, enabling the reconstruction boundary to better align with input. Supplementary shows the visual effects of this module in detail.

SparseFlex and Frustum-aware Sectional Voxel Training. To demonstrate the effectiveness of SparseFlex and frustum-aware sectional voxel training strategy, we evaluate the runtime of the feed-forward and GPU memory consumption with different settings, as shown in Tab. 3. It demonstrates that SparseFlex effectively reduces the required GPU memory and runtime during network feed-forward. However, it is not efficient to scale to higher resolution. The frustum-aware sectional voxel training strategy eliminates the reliance on the entire surface extraction during rendering, significantly reducing the memory requirements during training.

Sparse Voxel Resolutions. Higher resolution leads to better VAE reconstruction quality as shown in Tab. 1. Fig. 5 illustrates the qualitative comparison of SparseFlex VAE with different resolutions, along with TRELLIS with a resolution of 256. Thanks to geometry encoding, our approach achieves better geometry reconstruction with the same resolution as TRELLIS. Benefiting from efficient training, more details of complex structures are revealed, such as the tank track in the first row, as the resolution increases.

[1]The original images in Fig. 6 are sourced from various 3D generation platforms, benchmarks (such as Tripo3D, Rodin, Meshy, Trellis, etc.).

Figure 6. **Single image-to-3D generations with in-the-wild images.**[1] The geometry of generated assets accurately preserves sharp edges and fine details.

5. Conclusion

In this paper, we present SparseFlex, a new sparse-structured isosurface representation for differentiable mesh reconstruction with high resolution using rendering supervision, enabling the reconstruction of open surfaces. Based on SparseFlex, we propose a novel frustum-aware sectional voxel training strategy with adaptive frustum control to efficiently train SparseFlex VAE with high-resolution, dramatically reducing memory consumption. This strategy also allows our method to reconstruction the interiors only using rendering loss. Finally, we develop the image-to-3D generation pipeline following TELLIS [74]. Experiments demonstrate state-of-the-art reconstruction accuracy and high-quality generation with open surfaces.

Limitations: Despite the strong performance of SparseFlex VAE in both reconstruction and image-to-3D generation, some limitations remain. 1) Open surface boundaries, while handled effectively by voxel pruning, may exhibit minor

artifacts at lower resolutions. 2) High-resolution generation remains computationally demanding. 3) Enhanced control over the generation of interior structures is an area for future work.

References

[1] Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. *Real-time rendering*. AK Peters/crc Press, 2019. 2, 5

[2] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018. 6

[3] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*, pages 16123–16133, 2022. 3

[4] Jen-Hao Rick Chang, Yuyang Wang, Miguel Angel Bautista Martin, Jiatao Gu, Josh Susskind, and Oncel Tuzel. 3d shape tokenization. *arXiv preprint arXiv:2412.15618*, 2024. 2

[5] 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. *arXiv preprint arXiv:2412.17808*, 2024. 3, 6, 7, 13

[6] Weikai Chen, Cheng Lin, Weiyang Li, and Bo Yang. 3psdf: Three-pole signed distance function for learning surfaces with arbitrary topologies. In *CVPR*, pages 18522–18531, 2022. 2, 3, 6

[7] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, pages 5939–5948, 2019. 3

[8] 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. *arXiv preprint arXiv:2409.12957*, 2024. 2, 3

[9] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G Schwing, and Liang-Yan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In *CVPR*, pages 4456–4465, 2023. 2, 3

[10] Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *NeurIPS*, 33: 21638–21652, 2020. 2, 3

[11] Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. Abo: Dataset and benchmarks for real-world 3d object understanding. In *CVPR*, pages 21126–21136, 2022. 2, 6

[12] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 13

[13] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *NeurIPS*, 36:35799–35813, 2023. 6

[14] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, pages 13142–13153, 2023. 2, 6

[15] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *ICRA*, pages 2553–2560, 2022. 2, 6

[16] Benoît Guillard, Federico Stella, and Pascal Fua. Meshudf: Fast and differentiable meshing of unsigned distance field networks. In *ECCV*, pages 576–592, 2022. 2, 3

[17] Benoit Guillard, Federico Stella, and Pascal Fua. Meshudf: Fast and differentiable meshing of unsigned distance field networks. In *ECCV*, pages 576–592, 2022. 3

[18] Xianglong He, Junyi Chen, Sida Peng, Di Huang, Yangguang Li, Xiaoshui Huang, Chun Yuan, Wanli Ouyang, and Tong He. Gvgen: Text-to-3d generation with volumetric representation. In *ECCV*, pages 463–479. Springer, 2024. 2

[19] Zhu Heming, Cao Yu, Jin Hang, Chen Weikai, Du Dong, Wang Zhangye, Cui Shuguang, and Han Xiaoguang. Deep fashion3d: A dataset and benchmark for 3d garment reconstruction from single images. In *ECCV*, pages 512–530, 2020. 2, 6, 7

[20] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 6

[21] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: large reconstruction model for single image to 3d. In *ICLR*, 2024. 3

[22] Jiahui Huang, Zan Gojcic, Matan Atzmon, Or Litany, Sanja Fidler, and Francis Williams. Neural kernel surface reconstruction. In *CVPR*, pages 4369–4379, 2023. 2, 3

[23] Zehuan Huang, Yuan-Chen Guo, Haoran Wang, Ran Yi, Lizhuang Ma, Yan-Pei Cao, and Lu Sheng. Mv-adapter: Multi-view consistent image generation made easy. *arXiv preprint arXiv:2412.03632*, 2024. 3

[24] Zehuan Huang, Hao Wen, Junting Dong, Yaohui Wang, Yangguang Li, Xinyuan Chen, Yan-Pei Cao, Ding Liang, Yu Qiao, Bo Dai, et al. Epidiff: Enhancing multi-view synthesis via localized epipolar-constrained diffusion. In *CVPR*, pages 9784–9794, 2024. 3

[25] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation. In *SIGGRAPH Asia Conference*, pages 1–9, 2022. 3

[26] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, pages 339–346, 2002. 2, 3

[27] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013. 2

[28] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 42(4):139–1, 2023. 5

[29] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013. 3, 4

[30] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM TOG*, 39(6), 2020. 4

[31] Yushi Lan, Fangzhou Hong, Shuai Yang, Shangchen Zhou, Xuyi Meng, Bo Dai, Xingang Pan, and Chen Change Loy. Ln3diff: Scalable latent neural fields diffusion for speedy 3d generation. In *ECCV*, pages 112–130, 2024. 2, 3

[32] Yushi Lan, Shangchen Zhou, Zhaoyang Lyu, Fangzhou Hong, Shuai Yang, Bo Dai, Xingang Pan, and Chen Change Loy. Gaussiananything: Interactive point cloud latent diffusion for 3d generation. *arXiv preprint arXiv:2411.08033*, 2024. 2, 3

[33] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. In *ICLR*, 2024. 3

[34] Weiyu Li, Jiarui Liu, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long. Craftsman: High-fidelity mesh generation with 3d native generation and interactive geometry refiner. *arXiv preprint arXiv:2405.14979*, 2024. 2, 3, 6, 7

[35] 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 preprint arXiv:2502.06608*, 2025. 2, 3

[36] Minghua Liu, Chong Zeng, Xinyue Wei, Ruoxi Shi, Linghao Chen, Chao Xu, Mengqi Zhang, Zhaoning Wang, Xiaoshuai Zhang, Isabella Liu, et al. Meshformer: High-quality mesh generation with 3d-guided reconstruction model. *NeurIPS*, 37:59314–59341, 2025. 5

[37] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. In *ICLR*, 2024. 3

[38] Yu-Tao Liu, Li Wang, Jie Yang, Weikai Chen, Xiaoxu Meng, Bo Yang, and Lin Gao. Neudf: Leaning neural unsigned distance fields with volume rendering. In *CVPR*, pages 237–247, 2023. 2, 3

[39] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *CVPR*, pages 10012–10022, 2021. 4

[40] Xiaoxiao Long, Cheng Lin, Lingjie Liu, Yuan Liu, Peng Wang, Christian Theobalt, Taku Komura, and Wenping Wang. Neuraludf: Learning unsigned distance fields for multi-view reconstruction of surfaces with arbitrary topologies. In *CVPR*, pages 20834–20843, 2023. 2, 3

[41] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, pages 163–169, 1987. 2, 3, 13

[42] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[43] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *CVPR*, pages 2837–2845, 2021. 2

[44] Luke Melas-Kyriazi, Christian Rupprecht, and Andrea Vedaldi. Pc2: Projection-conditioned point cloud diffusion for single-image 3d reconstruction. In *CVPR*, pages 12923–12932, 2023. 2

[45] META. Digital twin catalog, 2024. 2, 6

[46] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 5

[47] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation. In *CVPR*, pages 306–315, 2022. 3

[48] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *ICML*, pages 7220–7229, 2020. 2

[49] 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. 2

[50] Gregory M Nielson. Dual marching cubes. In *IEEE visualization*, pages 489–496, 2004. 3, 4

[51] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 6

[52] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, pages 523–540, 2020. 3, 4

[53] Songyou Peng, Chiyu Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. *NeurIPS*, 34:13032–13044, 2021. 2

[54] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. 2, 4

[55] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 30, 2017. 2

[56] 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*, pages 4209–4219, 2024. 2, 3, 5, 6

[57] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *NeurIPS*, 34: 6087–6101, 2021. 2

[58] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. *ACM TOG*, 42(4): 37:1–37:16, 2023. 2, 3, 4, 5, 6

[59] Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. In *ICLR*, 2024. 3

[60] 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 *CVPR*, pages 19615–19625, 2024. 2

[61] Stefan Stojanov, Anh Thai, and James M Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *CVPR*, pages 1798–1808, 2021. 2, 6

[62] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *ECCV*, pages 1–18. Springer, 2024. 3

[63] Jiaxiang Tang, Zhaoshuo Li, Zekun Hao, Xian Liu, Gang Zeng, Ming-Yu Liu, and Qinsheng Zhang. Edgerunner: Autoregressive auto-encoder for artistic mesh generation. *arXiv preprint arXiv:2409.18114*, 2024. 2

[64] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. *NeurIPS*, 35:10021–10039, 2022. 2, 3

[65] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *NeurIPS*, 30, 2017. 3

[66] Peng Wang and Yichun Shi. Imagedream: Image-prompt multi-view diffusion for 3d generation. *arXiv preprint arXiv:2312.02201*, 2023. 3

[67] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021. 3

[68] Peng-Shuai Wang, Yang Liu, and Xin Tong. Dual octree graph networks for learning adaptive volumetric shape representations. *ACM Transactions on Graphics (TOG)*, 41(4): 1–15, 2022. 13

[69] Zhengyi Wang, Yikai Wang, Yifei Chen, Chendong Xiang, Shuo Chen, Dajiang Yu, Chongxuan Li, Hang Su, and Jun Zhu. Crm: Single image to 3d textured mesh with convolutional reconstruction model. In *ECCV*, pages 57–74, 2024. 2, 5

[70] Xinyue Wei, Fanbo Xiang, Sai Bi, Anpei Chen, Kalyan Sunkavalli, Zexiang Xu, and Hao Su. Neumanifold: Neural watertight manifold reconstruction with efficient and high-quality rendering support. *arXiv preprint arXiv:2305.17134*, 2023. 2

[71] Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Meshlrm: Large reconstruction model for high-quality meshes. *arXiv preprint arXiv:2404.12385*, 2024. 5

[72] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 29, 2016. 3

[73] 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. *arXiv preprint arXiv:2405.14832*, 2024. 2, 3, 6, 8

[74] 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. *arXiv preprint arXiv:2412.01506*, 2024. 2, 3, 4, 5, 6, 8

[75] Hongyi Xu and Jernej Barbič. Signed distance fields for polygon soup meshes. In *Graphics Interface*, pages 35–41. 2014. 2

[76] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024. 2, 3, 5, 6, 8

[77] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. In *ECCV*, pages 1–20. Springer, 2024. 3

[78] Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. Swin3d: A pretrained transformer backbone for 3d indoor scene understanding. *arXiv preprint arXiv:2304.06906*, 2023. 4

[79] Zhengming Yu, Zhiyang Dou, Xiaoxiao Long, Cheng Lin, Zekun Li, Yuan Liu, Norman Müller, Taku Komura, Marc Habermann, Christian Theobalt, Xin Li, and Wenping Wang. Surf-d: Generating high-quality surfaces of arbitrary topologies using diffusion models. In *ECCV*, pages 419–438, 2024. 2, 3, 6

[80] Biao Zhang and Peter Wonka. Lagem: A large geometry model for 3d representation learning and diffusion, 2024. 2, 3

[81] Biao Zhang, Matthias Nießner, and Peter Wonka. 3dilg: Irregular latent grids for 3d generative modeling. *NeurIPS*, 35: 21871–21885, 2022. 3

[82] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM TOG*, 42(4): 92:1–92:16, 2023. 2, 3

[83] Bowen Zhang, Yiji Cheng, Jiaolong Yang, Chunyu Wang, Feng Zhao, Yansong Tang, Dong Chen, and Baining Guo. Gaussiancube: A structured and explicit radiance representation for 3d generative modeling. *arXiv preprint arXiv:2403.19655*, 2024. 2

[84] Biao Zhang, Jing Ren, and Peter Wonka. Geometry distributions. *arXiv preprint arXiv:2411.16076*, 2024. 2

[85] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *ECCV*, pages 1–19. Springer, 2024. 3

[86] 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 TOG*, 43(4):1–20, 2024. 2, 3, 6, 13

[87] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *CVPR*, pages 16259–16268, 2021. 2

[88] 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. *NeurIPS*, 36: 73969–73982, 2023. 2, 3

[89] Xinyang Zheng, Yang Liu, Pengshuai Wang, and Xin Tong. Sdf-stylegan: implicit sdf-based stylegan for 3d shape generation. In *Computer Graphics Forum*, pages 52–63, 2022. 3

[90] Xin-Yang Zheng, Hao Pan, Peng-Shuai Wang, Xin Tong, Yang Liu, and Heung-Yeung Shum. Locally attentional sdf diffusion for controllable 3d shape generation. *ACM TOG*, 42 (4):1–13, 2023. 2, 3

[91] Zi-Xin Zou, Shi-Sheng Huang, Yan-Pei Cao, Tai-Jiang Mu, Ying Shan, Hongbo Fu, and Song-Hai Zhang. Gp-recon: Online monocular neural 3d reconstruction with geometric prior. *IEEE TVCG*, 2024. 3

[92] Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. In *CVPR*, pages 10324–10335, 2024. 3

A. VAE Training Losses

For training our SparseFlex VAE, we use the losses described below:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{render}} + \lambda_2 \mathcal{L}_{\text{prune}} + \lambda_3 \mathcal{L}_{\text{KL}} + \lambda_4 \mathcal{L}_{\text{flex}}, \quad (6)$$

where $\lambda_1 = 1.0$, $\lambda_2 = 0.2$, $\lambda_3 = 0.001$, and $\lambda_4 = 1.0$.

We use $\mathcal{L}_{\text{render}}$ to supervise the rendered depth maps, normal maps and mask maps, which is defined as:

$$\mathcal{L}_{\text{render}} = \lambda_d \mathcal{L}_d + \lambda_n \mathcal{L}_n + \lambda_m \mathcal{L}_m + \lambda_{ss} \mathcal{L}_{ss} + \lambda_{lp} \mathcal{L}_{lp}, \quad (7)$$

where $\lambda_d = 10.0$, $\lambda_n = 4.0$, $\lambda_m = 1.0$, $\lambda_{ss} = \lambda_{lp} = 0.5$. $\mathcal{L}_{ss}$ and $\mathcal{L}_{lp}$ denote SSIM loss and LPIPS loss, and are only applied on normal maps. L1 supervision is applied for the rest of losses.

For the structure loss $\mathcal{L}_{\text{prune}}$, we use it after two upsampled blocks, which leads to cross-level self-pruning operations, enhancing more accurate reconstruction results for local details.

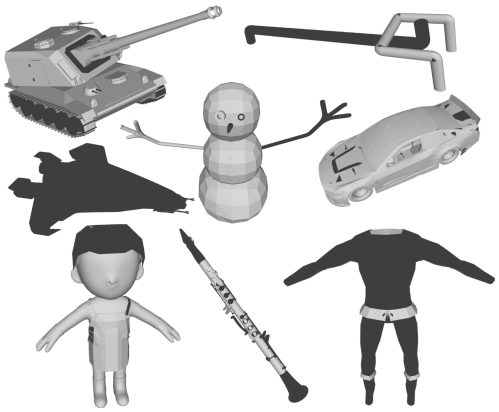

Figure 7. **Examples with inconsistent normal orientations in the large-scale dataset.**

B. Mesh Normal Repair

The datasets contain a large number of meshes with inconsistent normal orientations, as shown in Figure 7. Since we aim at open-surface reconstruction and primarily use rendering supervision, these inconsistencies have a significant negative impact on the training of SparseFlex VAE and structured latent flow model. To address this issue, we employ data preprocessing to repair the expected face normals. Specifically, we utilize Blender [12] to recalculate the face normals, ensuring consistency within each connected component and oriented them outward. Due to the ambiguity in determining the outward orientation, certain open surface components from the mesh still have flipped normals. We then compute the ambient occlusion for each component to determine their visibility ratio. According to this, we flip components whose visibility increases after flipping their normals.

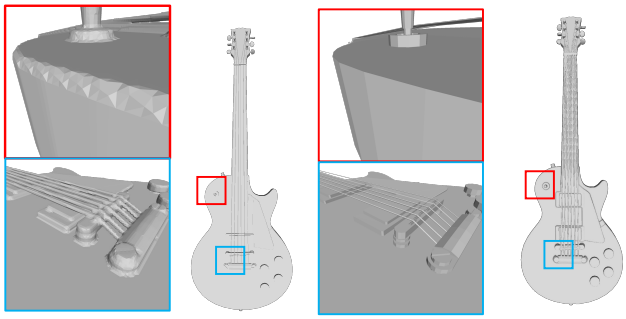

Figure 8. **Comparison of mesh processed with watertight conversion (right) and raw mesh (left).**

C. Degradation in Watertight Conversion

Approaches based on SDF/occupancy field often require a time-consuming watertight conversion for constructing 3D ground-truth supervision. Most extract the double-side mesh from the UDF field computed from ground-truth mesh, retaining the maximum connected component [68] or removing the interior by calculating visibility [86]. This watertight conversion pipeline requires applying Marching Cubes [41] to extract the surface with a small iso-value, which introduces further inaccuracies and artifacts. Furthermore, the dilation introduced by Marching Cubes applied to UDF makes it challenging to preserve the sharp features of raw data. Fig. 8 demonstrates the comparison between the mesh converted by code scripts from Dora [5] and raw mesh, it exhibits significant degradation of details in watertight mesh (right).

D. Self-Pruning for Open Surface

After dense upsampling, voxels near the open boundaries are often redundant and cause noticeable jagged artifacts for the open surface, leading to degraded perceptual quality. In that case, we incorporate self-pruning for SparseFlex VAE. Figure 9 shows the comparisons with and without the apply self-pruning. The visualization demonstrates that self-pruning effectively reduces the artifacts around open boundaries.

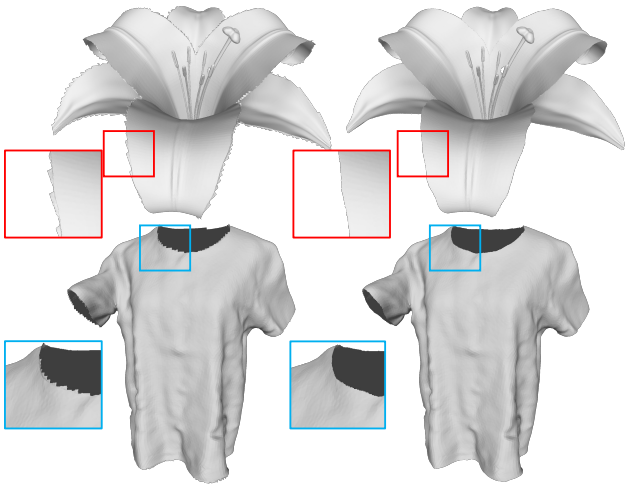

Figure 9. **Effects of self-pruning for open surface shapes.** It exhibits noticeable jagged artifacts without applying self-pruning in upsampling module.