

Nexus: Native Mesh Generation with Diffusion

HANXIAO WANG*, MAIS, Institute of Automation, Chinese Academy of Sciences; School of Artificial Intelligence, University of Chinese Academy of Sciences, China

YING-TIAN LIU*, VAST, China

YUAN-CHEN GUO, VAST, China

QI-YUAN FENG, Tsinghua University, China

ZI-XIN ZOU, VAST, China

DING LIANG, VAST, China

BIAO ZHANG†, Xi'an Jiaotong University, China

YAN-PEI CAO†, VAST, China



Fig. 1. **Meshes generated by Nexus.** By combining hierarchical octree diffusion with our novel spacetime topology diffusion, Nexus eliminates the need for vertex sorting, achieving robust and scalable 3D meshes generation.

*Both authors contributed equally to this research.

†Corresponding authors.

Authors' Contact Information: Hanxiao Wang, MAIS, Institute of Automation, Chinese Academy of Sciences; School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China, wanghanxiao18@mails.ucas.ac.cn; Ying-Tian Liu, VAST, Beijing, China, lytlogic@gmail.com; Yuan-Chen Guo, VAST, Beijing, China, imbenyguo@gmail.com; Qi-Yuan Feng, Tsinghua University, Beijing, China, fqy22@mails.tsinghua.edu.cn; Zi-Xin Zou, VAST, Beijing, China, zouzx1997@gmail.com; Ding Liang, VAST, Beijing, China, liangding1990@163.com; Biao Zhang, Xi'an Jiaotong University, Xi'an, China, biao.z@outlook.com; Yan-Pei Cao, VAST, Beijing, China, caoyanpei@gmail.com.



This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2026 Copyright held by the owner/author(s).
ACM 1557-7368/2026/7-ART71
<https://doi.org/10.1145/3811344>

Generating high-quality triangle meshes is essential for film, gaming, and interactive 3D applications. Mainstream methods rely on mesh serialization and autoregressive processes, which struggles in effective inference and is sensitive to error accumulation. In this paper, we present **Nexus**, a diffusion method that achieves holistic mesh generation via decoupled vertex and topology generation. First, we view mesh vertices as sparse voxels organized as an octree and adopt a diffusion model to generate the vertices in a coarse-to-fine manner. Second, for topology modeling, we propose *Space-time Interval*, as an extension of *Spacetime Distance* to encode arbitrary edge and face topology into continuous per-vertex embeddings. It allows for a global and efficient recovery of complex topology. We then employ a diffusion model to generate the continuous embeddings on the generated vertices. Extensive experiments on the Objaverse and Toys4K datasets and in-the-wild images demonstrate that our method outperforms state-of-the-art autoregressive and two-stage baselines, effectively circumventing the inherent limitations of sequential mesh modeling. A blind user study from 3D practitioners confirms strong perceptual preference for our results.

CCS Concepts: • **Networks**; • **Computing methodologies** → **Mesh models**;

Additional Key Words and Phrases: Mesh Generation, Deep Learning, Octree

ACM Reference Format:

Hanxiao Wang, Ying-Tian Liu, Yuan-Chen Guo, Qi-Yuan Feng, Zi-Xin Zou, Ding Liang, Biao Zhang, and Yan-Pei Cao. 2026. Nexus: Native Mesh Generation with Diffusion. *ACM Trans. Graph.* 45, 4, Article 71 (July 2026), 13 pages. <https://doi.org/10.1145/3811344>

1 Introduction

Meshes play a fundamental role in modern film and gaming industries for their expressiveness, compactness and compatibility. Recently, the community is thrilled to explore high-quality mesh generation by treating meshes as 3D field [Lai et al. 2025a; Li et al. 2025; Xiang et al. 2025; Zhang et al. 2024]. On the other hand, artistic meshes with good topologies are still favored for manipulation, articulation and animation. Recent advancements in artistic mesh generation [Chen et al. 2024b, 2025; Hao et al. 2024; Siddiqui et al. 2024; Tang et al. 2024] usually represent 3D meshes with 1D discrete tokens and adopt autoregressive transformers to predict the tokens one by one. They mainly focus on how to compress meshes in shorter sequences to ease the training and inference by exploring geometry-based encoding process [Chen et al. 2025; Lionar et al. 2025; Tang et al. 2024; Weng et al. 2025].

Despite the promising results they achieve, there remain challenges to make these approaches grounded. First, the mesh sequences usually grow linearly as the mesh complexity increases. It will take numerous auto-regressive steps to generate a single mesh. The generation time can be in minute-level despite being powered by state-of-the-art auto-regressive inference engine [Kwon et al. 2023; Zheng et al. 2024]. Second, the quality of the generated mesh is significantly influenced by error accumulation in the autoregressive process. Once some tokens are generated incorrectly, the subsequent mesh sequences would become meaningless. Moreover, from an intuitive perspective, a mesh does not bear any intrinsic order in nature, which means that it should be treated as a whole just as other signals, such as images, point clouds, etc. Thus, these key problems prompt us to think whether there exists an alternative approach to generate the 3D mesh in a holistic manner. Some current diffusion-based methods treat triangle faces in raw space [Alliegro et al. 2023] or latent space [He et al. 2025]. But they still struggles in modeling the distribution of a token set in variable length, which hinders flexible scaling. The fundamental challenge is obvious: is it possible to generate an artistic mesh as a holistic structure, while bypassing the difficulty in modeling a token set in variable length?

To overcome the challenge, we present **Nexus**, a novel framework that eliminates serialization entirely by leveraging diffusion processes for both geometry and topology. Our approach disentangles mesh generation into two parallelizable stages: hierarchical vertex generation and vertex-conditioned topology generation.

For geometry, we abandon the 1D sequential paradigm in favor of a spatial hierarchical representation. We treat vertices as sparse occupancy signals within an octree [Meagher 1982] and propose a coarse-to-fine vertex octree diffusion model. By defining the

diffusion process directly over the tree structure, we generate geometry layer-by-layer, establishing the global shape and progressively adding high-frequency details.

For topology, we introduce *Spacetime Interval*, which generalizes the pairwise spacetime distance from SpaceMesh [Shen et al. 2024] to a higher-order indicator. This allows us to encode not just edges, but also complex even non-manifold face topology directly into per-vertex embeddings. Instead of predicting a sequence of discrete face indices [Nash et al. 2020], we propose a topology autoencoder to encode vertices into a continuous geometric latent space. A diffusion model then generates these latents on the vertices. Leveraging the Spacetime Interval can easily extract edges and faces. Crucially, unlike previous metric-based methods restricted to manifolds, our formulation handles arbitrary artistic topology, including non-manifold geometry.

By eliminating the requirement for sorting and sequential prediction, Nexus achieves robust, scalable, and high-quality mesh generation. We validate our approach on the Objaverse and Toys4K datasets and in-the-wild images, demonstrating that Nexus outperforms state-of-the-art autoregressive and two-stage baselines in generation quality.

In summary, our contribution is as follows:

- We propose Nexus, a fully diffusion-based framework for artistic mesh generation that is inherently **sort-free**, eliminating the bottlenecks of autoregressive serialization.
- We design a hierarchical **octree diffusion** method to generate vertex structures efficiently in a coarse-to-fine and holistic manner.
- We present the **Spacetime Interval**, a geometric indicator designed to encode topology into per-vertex features. Utilizing this metric within a topology autoencoder, we obtain continuous latents that unlock the capability of diffusion-based topology generation.
- We demonstrate state-of-the-art performance on large-scale datasets, successfully generating complex meshes with arbitrary topologies while avoiding the error accumulation typical of sequence-based models.

2 Related Work

2.1 Learning on octree structures

Octree representations recursively partition 3D space, serving as a highly efficient data structure in network design.

Discriminative tasks. Foundational works like [Wang et al. 2017, 2018] established octree-based convolutions, while recent architectures have integrated Transformers (OctFormer [Wang 2023]) and State Space Models (Point Mamba [Liu et al. 2024]) to capture global dependencies. Furthermore, octrees have proven effective in neural rendering tasks, such as Octree-GS [Ren et al. 2024], by enabling level-of-detail management for real-time applications.

Octree-based 3D Generation. Octrees have become a critical representation due to their adaptive multi-scale capabilities. To address the fixed-resolution limitations of traditional VAEs, some works [Deng et al. 2025; Guo et al. 2025; Wei et al. 2025; Xiong et al. 2025] proposed to use octrees to model the latent space of 3d objects.

Table 1. **Comparison with recent mesh generation methods.** Existing methods typically generate triangles in the raw space using autoregressive models (R A), while our method generates vertices in the raw space and topology in the latent space, both using diffusion models (R D + L D). **Notation:** (R): raw space; (L): latent space; (A): autoregressive; (D): diffusion.

Method	Orderless	Vertex	Topology
PolyGen	✗	(R A)	(R A)
MeshGPT	✗		(L A)
PolyDiff	✓		(L D)
MeshXL	✗		(R A)
MeshAnything	✗		(R A)
EdgeRunner	✗		(L A D)
BPT	✗		(R A)
SpaceMesh	✓	(R D)	-
Meshtron	✗		(R A)
DeepMesh	✗		(R A)
FastMesh	✗	(R A)	-
MeshCraft	✗		(L D)
Ours	✓	(R D)	(L D)

We leverage the octree representation to exploit its spatial sparsity for computational efficiency and its adaptive nature, which allows the model to determine a variable number of vertices based on global geometric complexity.

2.2 Mesh Generation

We list recent important methods about mesh generation in Table 1. **Serialization Order.** Most mesh generation methods are autoregressive and rely heavily on a specific serialization order (i.e., traversal path) to tokenize the mesh [Chen et al. 2024a,b; Hao et al. 2024; Nash et al. 2020; Siddiqui et al. 2024]. Recent works [Chen et al. 2025; Weng et al. 2025] devote significant effort to optimizing these traversal sequences to facilitate convergence. In contrast, while some diffusion-based methods [Alliegro et al. 2023] obviate the need for ordering, they often struggle with the discrete nature of mesh topology and require a pre-defined or fixed number of elements, limiting their ability to generate meshes with variable complexity.

Decoupled Mesh Generation. Beyond serialization, the strategy for modeling vertex coordinates and face topology is another critical design choice. Most contemporary methods [Chen et al. 2024a,b; Siddiqui et al. 2024] adopt a unified, face-by-face generation paradigm, where vertices and their connectivity are predicted simultaneously. However, this coupled approach often leads to geometric inconsistencies, such as "cracks". Early seminal work like PolyGen [Nash et al. 2020] proposed a decoupled scheme that first generates the entire vertex set and then predicts the topology conditioned on these vertices. We follow this two-stage decomposition as it provides a more global perspective on the mesh structure. Distinct from PolyGen's autoregressive formulation, we implement a full diffusion-based framework for both vertex and topology generation, leveraging the generative power of diffusion models to produce high-quality meshes with complex and coherent structures.

Table 2. **Table of notations.**

	Symbol	Meaning
Train data	$\mathcal{M} = (\mathcal{V}, \mathcal{F})$	Mesh
	$\mathcal{V} = \{\mathbf{v} \in \mathbb{R}^3\}$	Vertices
	$\mathcal{F} = \{\mathbf{f} \in \mathcal{V}^3\}$	Faces
	C	Condition
Vertex	$\mathcal{O} = \{\mathcal{O}_d\}_{d=0}^D$	Octrees of all depths
	$\mathcal{O}_d = \{n_{d,k}\}_k$	Octree at depth d
	$n_{d,k} \in \{0, 1\}$	Node k at depth d
Topology	$\mathcal{Z} = \{\mathbf{z}_v \mid \mathbf{v} \in \mathcal{V}\}$	Spacetime embeddings
	$\mathbf{z}_v = [\mathbf{s}_v, \mathbf{t}_v] \in \mathbb{R}^c$	
	$\mathbf{t}_v \in \mathbb{R}^{c/2}$	Temporal component
	$\mathbf{s}_v \in \mathbb{R}^{c/2}$	Spatial component
	$d_{u,v} \in \mathbb{R}$	Edge indicator
	$d_{u,v,w} \in \mathbb{R}$	Face indicator

3 Methodology

We present **Nexus**, a generative framework designed to synthesize triangle meshes $\mathcal{M}$ conditioned on an input C (e.g., point clouds or images).

Formally, we denote a mesh as $\mathcal{M} = (\mathcal{V}, \mathcal{F})$, where $\mathcal{V}$ denotes the set of vertices and $\mathcal{F}$ denotes the faces. Our approach models the probability model $p(\mathcal{M}|C)$. The process is decomposed into two stages: vertex generation $p(\mathcal{V}|C)$, and vertex-conditioned topology generation $p(\mathcal{F}|\mathcal{V}, C)$, formulated as:

$$p(\mathcal{M}|C) = \underbrace{p(\mathcal{V}|C)}_{\text{vertices}} \cdot \underbrace{p(\mathcal{F}|\mathcal{V}, C)}_{\text{topology}}. \quad (1)$$

In Sec. 3.1, we describe how to model the vertex distribution $p(\mathcal{V}|C)$. The model operates in a coarse-to-fine manner: we first generate a coarse set of vertices and subsequently refine them level-by-level. Specifically, each level is modeled by a diffusion model operating on an octree representation.

In Sec. 3.2, we detail how to learn the topology distribution $p(\mathcal{F}|\mathcal{V}, C)$. In this stage, we train two components, (1) a topology autoencoder that encodes the topological information of edges and faces into per-vertex latent features; (2) a diffusion model trained within the latent space.

Comparison with Existing Methods. Our design addresses the fundamental limitations of autoregressive mesh generation. Standard autoregressive methods [Chen et al. 2024a,b, 2025; Siddiqui et al. 2024; Weng et al. 2025] treat the mesh as an 1D sequence, which necessitates an arbitrary sorting order (e.g., Z-order). This formulation creates two issues: (1) sensitivity to permutation, where different sortings yield inconsistent results, and (2) error accumulation, where early mistakes lead to broken structures. While recent two-stage approaches like FastMesh [Kim et al. 2025] attempt to mitigate this, they still rely on sorting for autoregressive vertex generation and are limited to predicting wireframes without face information. In contrast, **Nexus** is a native mesh generation framework that is inherently *sort-free*. By adapting diffusion models for both vertex

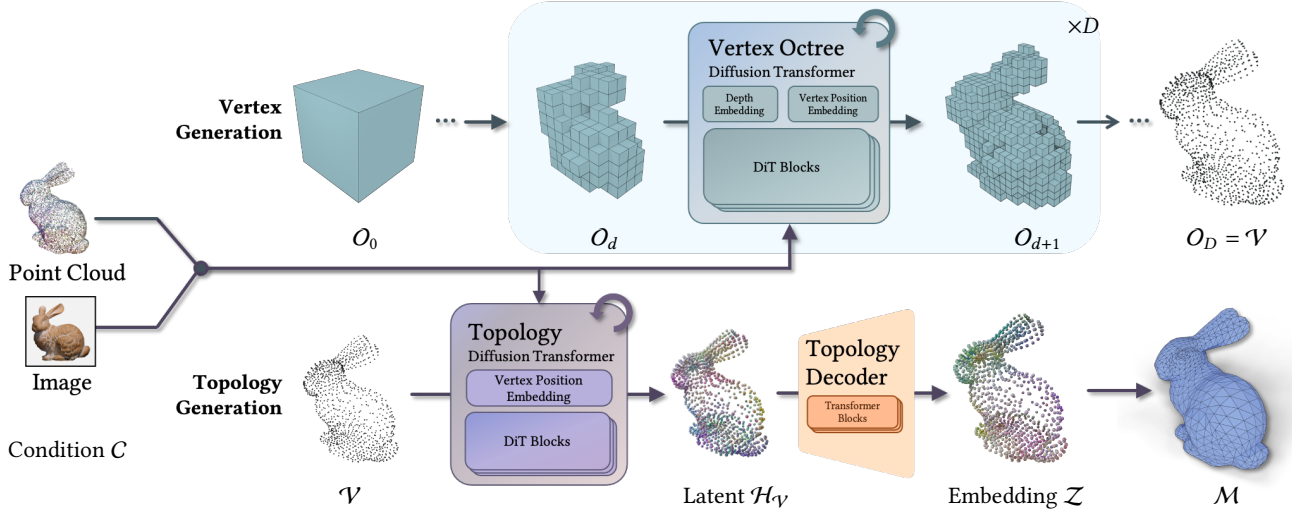


Fig. 2. **Pipeline of Nexus.** Given a condition (e.g., image or point cloud), we first generate vertices via a coarse-to-fine octree diffusion model. Subsequently, we synthesize per-vertex topology latents, which are decoded into embeddings to recover mesh edges and faces via spacetime intervals.

synthesis and topology modeling, we eliminate the need for serialization, enabling robust generation of complete surface meshes.

3.1 Vertex Generation

We represent the vertex set $\mathcal{V}$ as an octree structure $\mathcal{O}$, denoted as

$$\mathcal{V} \rightarrow \mathcal{O} = \{\mathcal{O}_0, \mathcal{O}_1, \dots, \mathcal{O}_D\},$$

where each coordinate is quantized into D -bit and $\mathcal{O}_d$ represents the occupancy status of nodes at depth d (see Fig. 2). The final vertex set $\mathcal{V}$ is obtained by extracting the 3D coordinates of all occupied leaf nodes in $\mathcal{O}_D$, i.e., the centers of the occupied grid cells at the finest octree level. We formulate the conditional generation of vertices as a sequential probabilistic process, factorized across the tree depths:

$$p(\mathcal{V}|\mathcal{C}) = p(\mathcal{O}|\mathcal{C}) = \prod_{d=1}^D p(\mathcal{O}_d|\mathcal{O}_{d-1}, \mathcal{C}). \quad (2)$$

In other words, each level of octree model $\mathcal{O}_d$ is conditioned on the previous level $\mathcal{O}_{d-1}$.

3.1.1 Training. We train a unified network capable of modeling the probabilities $p(\mathcal{O}_d|\mathcal{O}_{d-1}, \mathcal{C})$ across levels simultaneously. Specifically, an octree state $\mathcal{O}_d$ at depth d consists of a set of nodes $\{\mathbf{n}_{d,k}\}_k$, where the value of a node $\mathbf{n}_{d,k} \in \{0, 1\}$ indicates whether a node is occupied (1) or empty (0). Given the state at the previous level $\mathcal{O}_{d-1}$, we focus solely on the occupied nodes and we aim to train a probabilistic model to predict the occupancy of their corresponding child nodes at level d (see Fig. 2).

In our octree formulation, each node subdivides into 8 child nodes. Consequently, the training target is represented by a binary vector of size $|\sum_k \mathbf{n}_{d-1,k}| \times 8$ where $|\sum_k \mathbf{n}_{d-1,k}|$ gives the count of occupied nodes at the previous level $d-1$.

We train a unified network across all depths to predict the occupancy of the 8 child nodes for each occupied parent node. To apply continuous generative modeling, we cast the binary targets into real

values $\{0, 1\}$ and employ flow matching [Lipman et al. 2023; Liu et al. 2023] with velocity prediction parameterization. The backbone is a diffusion transformer that treats the 8-value occupancy pattern as a single token. We utilize 3D RoPE [Su et al. 2023] for position encoding and a learnable depth embedding before all transformer blocks. The input condition $\mathcal{C}$, is encoded using jointly trained VecSet [Zhang et al. 2023] encoder (for point clouds) or pretrained DINOv3 [Siméoni et al. 2025] (for images), and integrated into the denoising network with cross-attention mechanisms (see Fig. 2).

3.1.2 Inference. The inference process operates in a coarse-to-fine manner. We start with the root level octree $\mathcal{O}_0$ containing a single occupied node. For subsequent depths, we iteratively extract the occupied nodes from the formerly predicted or given (for the root) $\mathcal{O}_{d-1}$ and employ the trained diffusion model to predict the occupancy of the next level $\mathcal{O}_d$ conditioned on the same input $\mathcal{C}$.

3.2 Topology Generation

To enable the generation of mesh topology, we design a topology autoencoder and employ a diffusion model conditioned on the vertices, formulated as

$$p(\mathcal{F}|\mathcal{V}, \mathcal{C}).$$

We model edge and face topology separately. This decomposition is motivated by computational efficiency: directly recovering faces from all vertex triplets requires $O(|\mathcal{V}|^3)$ computation, whereas our edge-first strategy reduces this to $O(|\mathcal{V}|^2)$ for edge recovery followed by triangle enumeration on the sparse edge graph. We encode the mesh topology (i.e. edges and faces) into latent representations. Specifically, we learn per-vertex embeddings to infer whether two vertices form an edge and three vertices form a face (illustrated in Fig. 3). Let $\mathcal{Z} = \{\mathbf{z}_v\}_{v \in \mathcal{V}}$ denote the per-vertex embeddings. We begin with modeling edge connectivity and extend it to faces later.

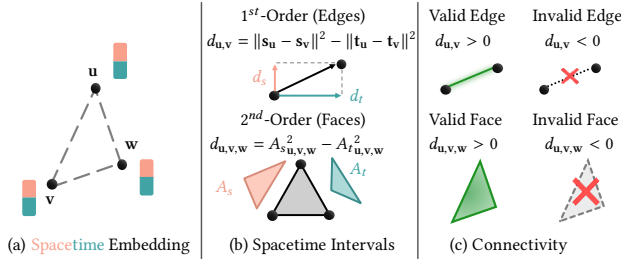


Fig. 3. **Illustration of topological indicators.** We define connectivity using spacetime intervals. The 1st-order interval (top) determines edge existence based on squared distance differences, while the 2nd-order interval (bottom) determines face validity based on the difference between temporal and spatial squared areas.

Edge Topology. Naive approach is to measure some distance between two vertex embeddings, like Euclidean distance or cosine similarity. Taking Euclidean distance as an example,

$$d_{u,v}^{\text{Eld}} = \|z_u - z_v\|^2.$$

the existence of an edge connecting u, v can be determined by whether the distance $d_{u,v}^{\text{Eld}}$ is greater than a threshold. To enable the distance to represent complex connectivity, SpaceMesh [Shen et al. 2024] proposes a spacetime distance that converges much faster than the two naive choices,

$$d_{u,v} = \|s_u - s_v\|^2 - \|t_u - t_v\|^2, \quad \text{where } z_u = [s_u, t_u], z_v = [s_v, t_v]. \quad (3)$$

Here $[]$ denotes concatenation. $s_v \in \mathbb{R}^{c/2}$ and $t_v \in \mathbb{R}^{c/2}$ denote the spatial and temporal subvectors of the vertex embedding z_v , respectively. $d_{u,v} > 0$ corresponds to a space-like separation, which we interpret as the existence of an edge between the two vertices. Conversely, a time-like separation implies no connection. We refer $d_{u,v}$ as **1st Order Spacetime Interval**.

There is a strong theoretical motivation for favoring spacetime distance over standard Euclidean distance. Euclidean metrics inherently impose a form of transitive proximity: if a point is close to two other points, those two points are also encouraged to be close to each other. This implicit clustering bias makes it difficult to faithfully represent complex topologies, where nodes may share local neighborhoods without being mutually connected. Our spacetime formulation bypasses this bottleneck by utilizing an indefinite metric signature (Minkowski space). The temporal component acts as a learned slack variable to dynamically absorb such topological distortions. Intuitively, the subtractive nature of the spacetime interval ($\|s\|^2 - \|t\|^2$) provides higher expressive power than purely positive-definite metrics, enabling the network to cleanly isolate unconnected vertex pairs that merely appear close in the ambient space. This perspective is consistent with the viewpoint presented in SpaceMesh [Shen et al. 2024]. Furthermore, this theoretical advantage is strongly validated by our ablation study (Tab. 9), where confining the latent space to Euclidean (L_2) or spherical metrics causes massive degradation in topological accuracy.

Face Topology. Although SpaceMesh [Shen et al. 2024] proposes an alternative representation for faces, its face recovery relies heavily on half-edge data structures, restricting its application to manifold meshes. To accommodate arbitrary topologies, we propose a novel indicator for face connectivity that extends the spacetime concept to higher dimensions. Analogous to edge modeling, a straightforward Euclidean indicator for face existence is the squared area of the triangle formed by three vertices. Specifically, let $z_{uw} = z_u - z_w$, $z_{vw} = z_v - z_w$. We define

$$d_{u,v,w}^{\text{Eld}} = \|z_{uw}\|^2 \|z_{vw}\|^2 - \langle z_{uw}, z_{vw} \rangle^2 \propto A_{u,v,w}^2 \quad (4)$$

which represents the squared area of triangles (scaled by a factor of 4). Inspired by the 1st Order Interval in Eq. 3, we propose the spacetime counterpart as:

$$d_{u,v,w} = A_{s,u,v,w}^2 - A_{t,u,v,w}^2 \quad (5)$$

which is the interval of squared areas between temporal and spatial components. We refer $d_{u,v,w}$ as **2nd Order Spacetime Interval**.

3.2.1 Training. We train two networks: (1) a topology autoencoder encodes topological information into per-vertex latents; (2) a diffusion model that generates these latents conditioned on the input C and the vertex set $\mathcal{V}$ generated from the previous stage.

Topology Autoencoder. Formally, the KL-regularized autoencoder are defined as

$$\begin{aligned} h &= \text{Enc}(\mathcal{V}, \mathcal{F}) && \text{encoder} \\ \mu, \sigma &= \text{Linear}(h) && \text{KL bottleneck} \\ \mathcal{H}_{\mathcal{V}} &= \mu + \sigma\epsilon, \epsilon \sim \mathcal{N}(0, I) && \\ \mathcal{Z} &= \text{Dec}(\mathcal{H}_{\mathcal{V}}) && \text{decoder} \end{aligned} \quad (6)$$

As shown in Fig. 4, in the encoding process, we treat each mesh as a graph $\mathcal{G}$, with vertices and faces as the nodes and connections between incident vertices and faces as the edges. To process the graph topology, we employ a hybrid architecture with interleaved graph convolutions and standard transformer blocks. Specifically, the encoder consists of interleaved SAGEConv [Hamilton et al. 2017] graph convolution layers and standard Transformer blocks. The graph layers encode topology connection effectively, while the Transformer blocks enable global information exchange across disconnected mesh components. The input node features are the vertex positions and face centroids.

The latent bottleneck is regularized by KL divergence [Kingma and Welling 2022]. The decoder consists of a stack of transformer blocks operating only on the vertex nodes and a linear head projecting the transformer output to per-vertex embeddings. In practice, we adopt separate embeddings for edge and face indicators. Then we sample vertex pairs or triplets to calculate their edge or face indicators in Eq. 3 and Eq. 5, and apply binary cross entropy loss on it. Since the edges and faces are extremely sparse in vertex pairs and triplets, we employ a simple loss balancing strategy to achieve better training stability and convergence. For edges, we divide all vertex pairs into TP, TN, FP, FN (T-true, F-false, P-positive, N-negative) groups and average the binary cross entropy losses within each group before aggregating them:

$$\mathcal{L}_{\text{edge}} = \frac{1}{4} \sum_{g \in \{\text{TP, TN, FP, FN}\}} \mathbb{E}_{(v_i, v_j) \in g} [\text{BCE}(d_{v_i, v_j}, e_{ij})] \quad (7)$$

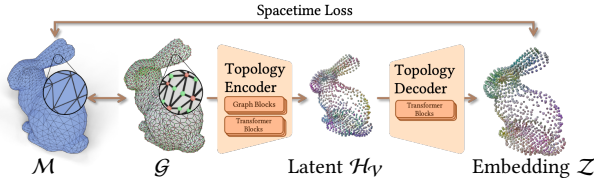


Fig. 4. **Architecture of the Topology VAE.** The input mesh $\mathcal{M}$ is first converted into a graph $\mathcal{G}$. The encoder, consisting of interleaved Graph and Transformer blocks, captures both local and global geometric information to produce per-vertex latents $\mathcal{H}_V$. The decoder then maps these latents to spacetime embeddings $\mathcal{Z}$, supervised by the Spacetime Loss.

where $e_{ij} = 1$ indicates that there exists an edge (v_i, v_j) and otherwise $e_{ij} = 0$. At each step, all vertex pairs are supervised with edge labels. For faces, we adopt a similar balanced loss except that we supervise all positive triplets and sampled negative triplets because vertex triplets are numerous.

Topology Generative Model. We train a diffusion model to generate the per-vertex latents $\mathcal{H}_V$ conditioned on the vertex set $\mathcal{V}$ and the input condition $\mathcal{C}$, employing flow matching [Lipman et al. 2023; Liu et al. 2023] with velocity prediction parameterization. The vertex position is incorporated via 3D RoPE. The input condition is injected into the main network with the same approach as the vertex generation stage.

3.2.2 Inference. Upon obtaining the vertex embeddings using the diffusion sampling and decoding process, we reconstruct the mesh topology through a hierarchical inference process. Since face recovering over all possible vertex triplets requires $O(|\mathcal{V}|^3)$ computation which is hard to handle, we start from edges. First, we recover the edge set by computing the pairwise spacetime interval for all vertex pairs. Then we identify *candidate faces* by searching for all closed 3-cycles (i.e., triangles) formed by the recovered edges and verify each candidate (u, v, w) using the 2nd order spacetime interval. This “edges-first” strategy ensures topologically consistency while significantly reducing computational overhead. By construction, a face can only be validated on edges that have already been confirmed, preventing face-edge inconsistency and ensuring topological coherence.

4 Experiments

4.1 Implementation Details

Vertex Diffusion. We train a diffusion transformer [Peebles and Xie 2023] (DiT) with approximately 2 billion parameters to recover vertex positions conditioned on point clouds and images. For the condition encoder, we employ a VecSet [Zhang et al. 2023] architecture comprising 8 layers with a hidden dimension of 2048. The input to the encoder consists of 8192 points sampled from the mesh surface with their surface normals. These features are projected into 1024 tokens with a dimension of 2048 and injected into the DiT. For the image encoder, we utilize a pre-trained DINOv3 [Siméoni et al. 2025] encoder to get image feature and injected into the DiT. For point cloud conditioning, we uniformly sample 8,192 points from

Table 3. **Quantitative comparison of mesh reconstruction quality on multiple datasets.**

Method	Objaverse		Toys4K	
	Hausdorff ↓	Chamfer ↓	Hausdorff ↓	Chamfer ↓
MeshAnything	0.327	0.128	0.264	0.110
MeshAnythingV2	0.265	0.089	0.238	0.090
TreeMeshGPT	0.183	0.055	0.133	0.047
BPT	0.126	0.043	0.091	0.037
FastMesh	0.145	0.040	0.100	0.033
Ours	0.104	0.031	0.071	0.027

the mesh surface along with their surface normals. For image conditioning, we render images at 512×512 resolution with the camera uniformly sampled in front of the object. We set the depth $D = 9$ which means the vertex positions are normalized and quantized into integers within the range $[0, 511]$. The model is trained using the Adam optimizer on 32 NVIDIA A100 GPUs for 400K steps. We use a learning rate of 1×10^{-4} and a weight decay of 0.01. For inference, we employ the DPM-Solver [Lu et al. 2022] sampler, performing 20 steps for each octree level.

Topology Autoencoder. We train a network with around 100 million parameters to encode topological information. The encoder processes vertices and face centroids using a hybrid architecture of interleaved graph blocks (containing graph convolutions) and transformer blocks. It consists of totally 24 layers with a hidden dimension of 512. The latent bottleneck dimension is set to 64, regularized by a KL divergence loss. The decoder employs a pure attention-based architecture with 16 layers and a hidden dimension of 1024. The model is trained with the Adam optimizer on 32 NVIDIA A100 GPUs for 50K steps, with a learning rate of 1×10^{-4} .

Topology Diffusion. We train another DiT with 2 billion parameters to generate topology latents on the vertex conditioned on $\mathcal{C}$. This model shares the architecture and hyperparameters as the vertex diffusion model. The model is trained using the Adam optimizer on 32 NVIDIA A100 GPUs for 200K steps, with a learning rate of 1×10^{-4} . Similar to the geometry stage, we use the DPM-Solver [Lu et al. 2022] with 20 inference steps. For inference, generating a mesh with up to 20,000 faces takes approximately 60 seconds on a single 24GB GPU.

Dataset. For training of all stages, we curate a dataset from Objaverse [Deitke et al. 2022] and ObjaverseXL [Deitke et al. 2023] with meshes with fewer than 20,000 faces. This results in a subset of approximately 1 million meshes. To ensure reliable evaluation, we perform testing on two diverse datasets that are excluded from the training phase:

- **Objaverse:** We collect a test set of 500 models drawn randomly from the non-training portion of Objaverse.
- **Toys4K:** To evaluate performance on low-poly shapes, we select 900 examples from the Toys4K dataset [Stojanov et al. 2021] with a face count threshold of 4,000.

4.2 Point-Cloud-Conditioned Mesh Generation

In this section, we compare our point-to-mesh results with state-of-the-art methods. Our evaluation includes one-stage methods

Table 4. **Additional mesh quality metrics on the Toy4k test set.** We report Edge Chamfer Distance (ECD), Normal Consistency (NC), average vertex count (#V), and average face count (#F) as complementary metrics to Tab. 3.

Method	ECD ↓	NC ↓	#V	#F
MeshAnything [Chen et al. 2024b]	0.1413	0.6106	242.5	431.4
MeshAnythingV2 [Chen et al. 2025]	0.1129	0.6238	667.0	1223.6
TreeMeshGPT [Lionar et al. 2025]	0.0670	0.2861	842.6	1592.2
BPT [Weng et al. 2025]	0.0471	0.6177	599.7	1148.5
FastMesh [Kim et al. 2025]	0.0357	1.9162	600.0	4758.7
Ours	0.0233	1.5441	1081.8	2287.0

such as MeshAnything v1/v2 [Chen et al. 2024b, 2025], BPT [Weng et al. 2025], and TreeMeshGPT [Lionar et al. 2025]. Furthermore, we include FastMesh [Kim et al. 2025] as a representative baseline for two-stage pipelines. We normalize all meshes to a $[-1, 1]$ bounding box and sample point clouds for evaluation. Quantitative evaluation is performed using Hausdorff Distance (HD) for maximum surface deviation and Chamfer Distance (CD) for average reconstruction error. Both metrics are computed using 50,000 points uniformly sampled from the mesh surface.

4.2.1 Quantitative Analysis. We compare Nexus with state-of-the-art baselines on the Objaverse and Toys4K datasets. The results are summarized in Tab. 3. Among the baselines, BPT [Weng et al. 2025] emerges as the most competitive one-stage autoregressive method, achieving the best Hausdorff scores among prior works. FastMesh [Kim et al. 2025], representing two-stage approaches, secures better Chamfer distances than BPT but suffers from higher Hausdorff distances. This discrepancy highlights the trade-off in existing methods: while FastMesh excels at average surface reconstruction, its deterministic connectivity prediction module often introduces severe redundant faces and geometric outliers, thereby degrading the worst-case error measured by HD.

In contrast, Nexus outperforms all baselines across both metrics on all datasets. By eliminating the autoregressive sorting bottleneck and employing diffusion for both geometry and topology, our method achieves superior global consistency while maintaining precise local details.

To provide a more comprehensive evaluation, we report additional mesh quality metrics in Tab. 4, including Edge Chamfer Distance (ECD), Normal Consistency (NC), average vertex count (#V), and average face count (#F). We compute ECD by only sampling points on edges. Here Normal Consistency is defined as the average dot product of neighboring face normals, within the range $[0, 2]$. Nexus achieves the lowest ECD (0.0233), indicating superior edge-level geometric accuracy. Our NC value (1.5441) is higher than baselines due to the absence of face normal prediction in our pipeline, which requires a post-processing orientation correction step. FastMesh only considers edge connectivity and thus produces numerous redundant faces.

4.2.2 Qualitative Comparison. We visually compare the generated meshes in Fig. 5, where the results highlight specific failure modes inherent to autoregressive architectures that Nexus successfully eliminates. FastMesh, representing two-stage models, relies on a

Table 5. **Robustness to input point cloud density.** Chamfer Distance remains stable across a $32\times$ density range (512–16,384 points). The model is trained with 8,192 points.

#Points	512	1024	2048	4096	8192	16384
CD ↓	0.0355	0.0258	0.0235	0.0235	0.0233	0.0231

Table 6. **Robustness to input noise.** We add Gaussian noise $\mathcal{N}(0, \sigma^2)$ to the input point cloud. Under typical real-scan noise ($\sigma=0.003$), CD degrades by less than 10%.

σ	0.000	0.001	0.003	0.010	0.030
CD ↓	0.0233	0.0250	0.0256	0.0309	0.0489

regression network for connectivity without generative ability. This limitation often manifests as topological defects in complex regions. For instance, it generates a broken neck for the violin (row 1) and loses structural integrity in the tractor wheel (row 3), suggesting a struggle to capture the distribution of valid topologies. Conversely, one-stage autoregressive methods like BPT are highly sensitive to sequence ordering, a weakness particularly evident in meshes with complex branching structures or enormous disconnected components. As seen in the tree model (row 4), BPT fails to generate leaves. In contrast, Nexus demonstrates superior robustness by treating both vertices and topology in a holistic manner learned by diffusion process. Our approach generates meshes with high structural integrity and fine detail, successfully eliminating both the typical sorting artifacts from autoregressive models and the connectivity errors of deterministic regressors.

4.2.3 Robustness Analysis for Point-Cloud-Conditioned Generation. We evaluate the robustness of Nexus to variations in input point cloud density and noise.

Input Point Cloud Density. Tab. 5 evaluates robustness to varying input point cloud density. Although trained with 8,192 points, Nexus maintains stable performance across a $32\times$ density range (512–16,384 points), with CD remaining approximately 0.023 from 2,048 points onward.

Input Noise. Tab. 6 evaluates robustness to Gaussian noise added to the input point cloud. Under typical real-scan noise levels ($\sigma=0.003$), CD degrades by less than 10% (0.0256 vs. 0.0233), confirming strong sensor-noise robustness.

4.2.4 User Study. To evaluate the perceptual quality of generated meshes, we conduct a rigorous blind pairwise user study and collect 1,221 comparison results from 3D practitioners in total. Participants are required to evaluate edge flow, structural integrity, and artifact reduction. As shown in Tab. 7, Nexus achieves the highest Elo rating (1440.4) and pair-wise preference rate (93.0%), significantly outperforming all baselines. This confirms that the topological quality advantage of our method is perceptually meaningful to professional users.

4.3 Image-Conditioned Mesh Generation

We qualitatively evaluate the image conditional generation ability of our method in Fig. 6, Fig. 7, and Fig. 13.

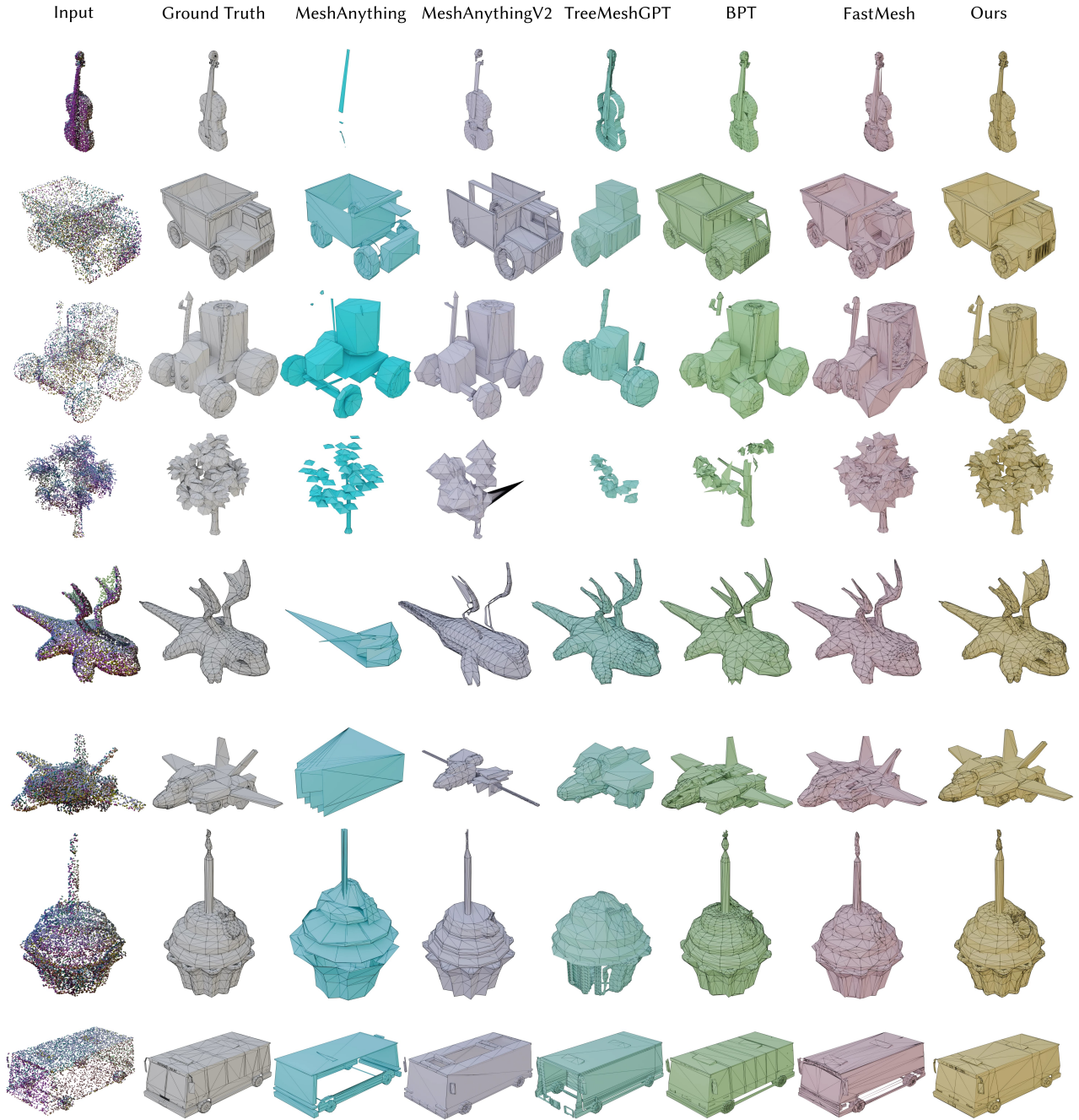


Fig. 5. **Qualitative comparison of mesh reconstruction quality on the Toys4K dataset.** The leftmost column shows the input point cloud.

Comparison with EdgeRunner. As shown in Fig. 6, our method significantly outperforms EdgeRunner in both geometric fidelity and topological quality. While EdgeRunner tends to produce over-smoothed, rough hulls with irregular triangulation, our approach recovers precise high-frequency details with clean element layouts. For instance, we accurately reconstruct intricate structures, such as the defined facial features of the Buddha (column 5) where the

baseline yields featureless approximations. Notably, our method achieves better topological completeness. For the anime character (column 6), our method can generate a mesh with detailed hair, whereas EdgeRunner results in disconnected limbs and missing thin structures.

Comparison with TRELIS. We compare our image-conditioned generation against TRELIS [Xiang et al. 2025], a state-of-the-art

Table 7. **User study results.** We conduct a blind pairwise study with 1,221 comparisons from 3D practitioners, evaluating edge flow, structural integrity, and artifact reduction.

Method	Elo Rating $\uparrow$	Preference (%) $\uparrow$
Ours	1440.4	93.0
FastMesh	1090.4	62.2
BPT	1075.3	60.9
TreeMeshGPT	928.2	41.2
MeshAnythingV2	806.6	28.9
MeshAnything	659.1	14.9



Fig. 6. **Qualitative comparison with EdgeRunner on image-conditioned generation.** Results show improved geometric quality and surface detail compared to Edgerunner across diverse object categories.

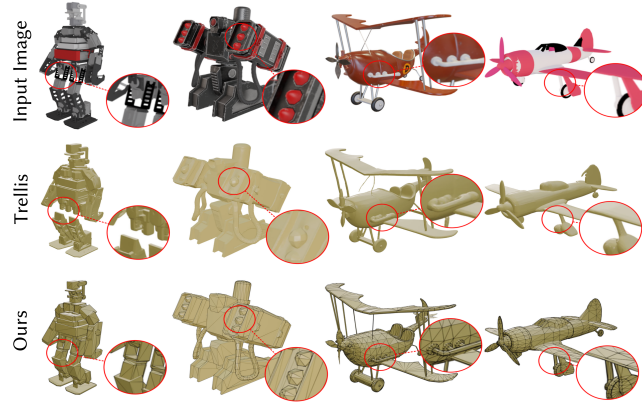


Fig. 7. **Qualitative comparison with TRELIS on image-conditioned generation.** Nexus generates native meshes with explicit topology and sharper geometric details, while TRELIS produces smooth isosurfaces via marching cubes.

method based on structured latent representations and isosurface extraction. As shown in Tab. 8, Nexus outperforms TRELIS on both standard geometric metrics (CD: 0.0681 vs. 0.0802; HD: 0.2309 vs. 0.2503) and 3D-aware semantic similarity metrics (ULIP-Sim [Xue et al. 2023], UNI3D-Sim [Zhou et al. 2023]). Specifically, we utilize

Table 8. **Image-conditioned generation: comparison with TRELIS.** We report standard geometric metrics (CD, HD) alongside 3D-aware semantic similarity (ULIP-Sim, UNI3D-Sim).

Method	CD $\downarrow$	HD $\downarrow$	ULIP-Sim $\uparrow$	UNI3D-Sim $\uparrow$
TRELIS [Xiang et al. 2025]	0.0802	0.2503	0.1317	0.2510
Ours	0.0681	0.2309	0.1438	0.2738

Table 9. **Quantitative comparison of face and edge metrics under different design choices.** The upper section shows results at 8k iterations. “Euclidean (L_2)” and “Spherical” replace the spacetime interval with the corresponding metric. The last row shows our final autoencoder at 50k iterations.

Experiment	Face			Edge		
	Recall $\uparrow$	Precision $\uparrow$	F1 Score $\uparrow$	Recall $\uparrow$	Precision $\uparrow$	F1 Score $\uparrow$
Ours	0.99995	0.99976	0.99985	0.99996	0.99991	0.99994
Minkowski loss	0.99985	0.99476	0.99729	0.99998	0.99996	0.99997
Euclidean (L_2)	0.93370	0.79619	0.85948	0.97632	0.88190	0.92671
Spherical (cos)	0.90626	0.50889	0.65178	0.99976	0.57707	0.73176
w/o balance loss	0.99995	0.99956	0.99976	0.99998	0.99990	0.99994
w/o enc attn	0.99986	0.99965	0.99976	0.99998	0.99991	0.99995
w/o enc GCN	0.98206	0.94899	0.96522	0.89112	0.73120	0.80327
w/o split feature	0.99960	0.96150	0.98016	0.98796	0.97315	0.98069
Ours (50k)	0.99998	0.99991	0.99994	1.00000	1.00000	1.00000

ULIP-Sim and Uni3D-Sim to measure the cosine similarity between the generated mesh and the input image within pre-trained 3D-image joint embedding spaces. Following the evaluation protocol of recent high-fidelity 3D generation works like Lattice [Lai et al. 2025b], these metrics are employed to capture high-level semantic fidelity that goes beyond pure geometry.

Fig. 7 provides a qualitative comparison. While TRELIS produces smooth surfaces via marching cubes, it tends to over-smooth fine geometric details. In contrast, Nexus preserves sharp features and generates artist-friendly mesh topology with explicit and clean edge flow.

Uncurated Generation Gallery. To demonstrate the consistency of our method beyond cherry-picked examples, Fig. 13 shows 50 randomly selected outputs from our model conditioned on images. No manual selection or filtering was applied, emphasizing the robustness of our holistic generation paradigm across diverse categories and geometric complexities.

4.4 Ablation Studies

To evaluate the effectiveness of some key designs, we conduct a series of ablation experiments focusing on the topology autoencoder and the proposed spacetime indicator. The quantitative results are summarized in Tab. 9.

Topology Indicator (Minkowski loss). To evaluate the effectiveness of our proposed **2nd Order Spacetime Interval** defined in Eq. 5, we investigate an alternative formulation for face topology derived from the pseudo-Euclidean inner product. While our default **2nd Order Spacetime Interval** calculates the difference between temporal and spatial squared areas independently, the alternative indicator, which we call Minkowski Interval, couples these components within a pseudo-Euclidean inner product. Let $\mathbf{u} = \mathbf{z}_u - \mathbf{z}_w$ and $\mathbf{v} = \mathbf{z}_v - \mathbf{z}_w$ be the relative vertex embeddings. The pseudo-Euclidean inner product $[\cdot, \cdot]$ is defined as:

$$[\mathbf{u}, \mathbf{v}] = \langle \mathbf{s}_u, \mathbf{s}_v \rangle - \langle \mathbf{t}_u, \mathbf{t}_v \rangle, \quad (8)$$

where $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean inner product. The face topology indicator $d_{u,v,w}^{\text{Mink}}$ is formulated as the determinant:

$$d_{u,v,w}^{\text{Mink}} = [\mathbf{u}, \mathbf{u}][\mathbf{v}, \mathbf{v}] - [\mathbf{u}, \mathbf{v}]^2. \quad (9)$$

Conceptually, this variant treats the latent space as a unified Minkowski manifold. However, as shown in Tab. 9, we observe that the baseline outperforms the Minkowski variant. Its performance on faces is significantly inferior to the original version. We hypothesize that by decoupling the temporal and spatial geometric determinants, the model gains better numerical stability during the optimization process, as it prevents the gradients of the temporal and spatial components from interfering within the quadratic expansion of the determinant.

Distance Metric (Euclidean / Spherical). To validate the necessity of the spacetime formulation, we replace the indefinite metric with standard Euclidean (L_2) and Spherical (cosine similarity) metrics. For Euclidean distance, edge indicator compares $\|\mathbf{z}_u - \mathbf{z}_v\|^2$ with a learned threshold and face indicator compares $A_{u,v,w}^2$ with another threshold. For spherical distance, edge indicator uses $\cos \theta_{u,v} = \frac{\mathbf{z}_u \cdot \mathbf{z}_v}{\|\mathbf{z}_u\| \|\mathbf{z}_v\|}$ while face indicators uses $\frac{\cos \theta_{u,v} + \cos \theta_{u,w} + \cos \theta_{v,w}}{3}$. As shown in Tab. 9, both alternatives cause massive topological degradation. The Euclidean variant achieves only 0.859 Face F1 (vs. 0.9999 for ours), while the Spherical variant drops further to 0.652. This confirms that the indefinite metric signature is essential: the subtractive temporal component provides the expressive slack needed to represent complex, non-planar connectivity patterns that positive-definite metrics cannot capture.

Loss Balancing Strategy (w/o balance loss). We employ a simple loss balancing strategy described in Eq.7 due to the extreme sparsity of edges and faces in candidate vertex pairs and triplets. Excluding this strategy (w/o balance loss) results in a decrease in the face F1 score (0.99976 vs. 0.99985), validating its necessity for achieving better performance during training.

Graph Convolution (w/o enc GCN). Removing the graph convolution layers from the encoder results in the most drastic performance decline, with the edge F1 score falling to 0.88448. This demonstrates that injecting the connectivity with graph operators is paramount. Without these layers, the model fails to effectively aggregate information from immediate neighbors, which is indispensable for understanding the initial topological structure of the mesh.

Full Attention (w/o enc attn). Restricting the encoder to only graph-based operations without global full-attention (transformer blocks) also leads to a performance decrease. This confirms that relying solely on local topological relationships is insufficient because graph convolutions cannot propagate messages between disjoint components (unconnected parts of the mesh). The global attention is therefore necessary to bridge this gap, enabling information exchange across the entire geometry regardless of connectivity.

Feature Decoupling Strategy (w/o feature split). In our default configuration, the per-vertex embedding $\mathcal{Z}$ is split into two disjoint subsets of channels: $\mathcal{Z}_{\text{edge}}$ for computing 1st-order intervals (edges) and $\mathcal{Z}_{\text{face}}$ for 2nd-order intervals (faces). In the w/o split variant, we employ a unified embedding to represent both edge and face

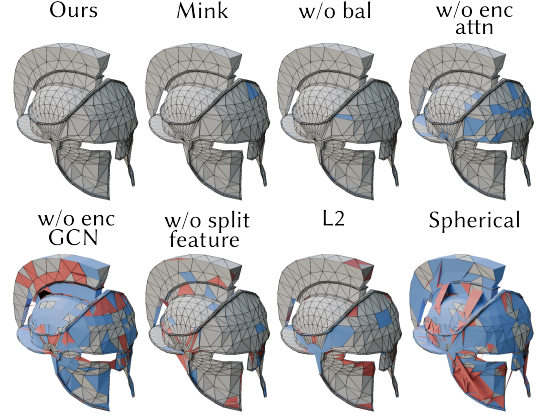


Fig. 8. **Qualitative ablation comparison with zoom-in details.** Each column shows a different ablation variant. Zoom-in regions highlight topological defects (holes, floating faces, broken edges) not fully captured by F1 scores alone.

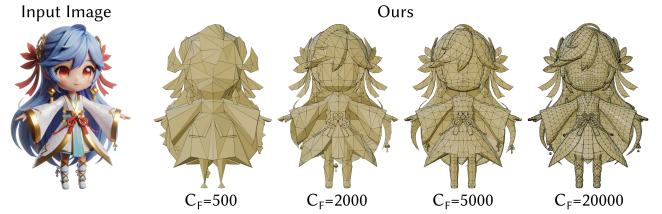


Fig. 9. **Level-of-Detail control.** Given the same input, Nexus generates meshes at different face budgets (500–20,000), preserving global shape while varying local detail.

topology simultaneously. As shown in Tab. 9, we observe a significant performance degradation, with the face F1 score dropping from 0.99985 to 0.98016. This drop suggests that forcing a single feature space to satisfy the geometric constraints of both pairwise (edge) and triplet (face) relationships leads to severe optimization difficulty. Decoupling these features allows the model to learn specialized representations for different topologies.

Fig. 8 provides qualitative comparisons for key ablation variants. The Euclidean (L_2) variant produces severe topological artifacts with many broken edges and missing faces. The Spherical variant shows even worse degradation, with large portions of connectivity lost. Removing graph convolutions (w/o enc GCN) leads to chaotic topology. In contrast, our full model achieves the best reconstruction

4.5 Applications

Level-of-Detail Control. By incorporating face-count conditioning in the first stage, Nexus can generate meshes at varying detail levels from the same input. Fig. 9 demonstrates generation at target budgets ranging from 500 to 20,000 faces, preserving overall shape while progressively adding geometric detail.

Mesh Editing and Texturing. Fig. 10 demonstrates that Nexus-generated meshes integrate seamlessly into standard graphics pipelines, supporting texture generation and interactive local mesh editing.

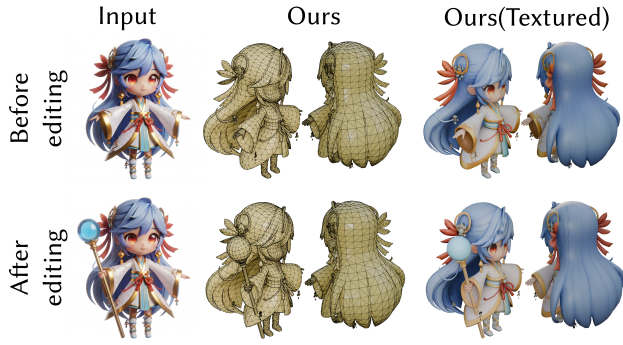


Fig. 10. **Downstream applications.** Nexus meshes support texture synthesis and interactive mesh editing.



Fig. 11. **Scene-level mesh generation.** By scaling to 20,000 faces, Nexus generates complex scenes composed of multiple objects.

Scene-level Generation. By scaling Nexus to 20,000 faces, we find that it can generate complex scene-level meshes composed of multiple objects (Fig. 11). Our topology diffusion uses approximately $|\mathcal{F}|/2$ tokens compared to $\sim 9|\mathcal{F}|$ tokens for vanilla autoregressive approaches, making large-scale generation feasible with inference completing in approximately 60 seconds on a single 24GB GPU.

5 Conclusion

In this paper, we presented Nexus, a novel sort-free approach for holistic mesh generation that fundamentally departs from the prevailing sequential serialization paradigm. Unlike autoregressive approaches that are plagued by permutation sensitivity and error accumulation, our method treats mesh generation as an order-agnostic, holistic process. We achieved this through several core innovations: a hierarchical octree diffusion model that synthesizes vertices in a coarse-to-fine manner, and the novel Spacetime Interval for topology representation. Based on the representation, we managed to encode discrete topology into continuous, diffusable vertex embeddings that can be easily modeled by standard diffusion models.



Fig. 12. **Failure cases.** From left to right: (a) **Ambiguity of geometry and texture:** missing small-scale components (e.g., a cup on the table); (b) **Missing faces:** large missing surfaces on building walls; (c) **Resolution-related artifacts:** distortions on fine structures at depth $D=9$ due to discrete coordinate snapping; (d) **Normal inconsistencies:** conflicting face orientations before the application of our orientation correction module.

Extensive experiments demonstrate that Nexus significantly outperforms state-of-the-art autoregressive and two-stage baselines. A blind user study with 1,221 pairwise comparisons from 3D practitioners confirms strong perceptual preference for our results. Furthermore, our method demonstrates robust performance under varying input noise and point cloud density, and scales to meshes with up to 20,000 faces. By decoupling geometry and topology into efficient, parallelizable diffusion processes, we not only improve generation quality but also offer a new perspective on structured mesh generation. Our work suggests that complex, non-Euclidean structures can be effectively modeled through continuous diffusion, paving the way for more robust and scalable 3D generative models.

Limitations. Although our approach can generate meshes in a holistic manner, the first stage can still cost around 1 minute to generate vertices in 512 resolution as it involves multiple diffusion inferences. We believe it can be accelerated by integrating few-step training or distillation techniques. In terms of the topology generation, our approach ignores the face normals so an extra orientation correction stage is required to make them consistent. We believe that a learning-based or rendering-based solution can be good enough. Also, despite of the robustness with in-the-wild inputs, it can still fails in some cases (Fig. 12), such as ambiguous about geometry and texture, missing faces and grid artifacts with limited resolution.

6 Acknowledgments

This work was supported in part by the International (Hong Kong, Macao, and Taiwan) Collaborative R&D Project, Beijing Major Science and Technology Project under Contract No. Z251100007125016. We would like to sincerely thank Yumeng Li and Peng-Shuai Wang for their valuable suggestions and insightful discussions.

References

- Antonio Alliegro, Yawar Siddiqui, Tatiana Tommasi, and Matthias Nießner. 2023. PolyDiff: Generating 3D Polygonal Meshes with Diffusion Models. arXiv:2312.11417 [cs.CV] <https://arxiv.org/abs/2312.11417>
- Sijin Chen, Xin Chen, Anqi Pang, Xianfang Zeng, Wei Cheng, Yijun Fu, Fukun Yin, Billzb Wang, Jingyi Yu, Gang Yu, et al. 2024a. Meshxl: Neural coordinate field for



Fig. 13. **Uncurated image-to-mesh generation gallery.** 50 randomly selected outputs with no manual filtering, demonstrating the consistency and robustness of Nexus across diverse categories.

generative 3d foundation models. *Advances in Neural Information Processing Systems* 37 (2024), 97141–97166.

Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, et al. 2024b. Meshanything: Artist-created mesh generation with autoregressive transformers. *arXiv preprint arXiv:2406.10163* (2024).

Yiyen Chen, Yikai Wang, Yihao Luo, Zhengyi Wang, Zilong Chen, Jun Zhu, Chi Zhang, and Guosheng Lin. 2025. Meshanything v2: Artist-created mesh generation with adjacent mesh tokenization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 13922–13931.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli

VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. 2023. Objaverse-XL: A Universe of 10M+ 3D Objects. *arXiv:2307.05663* [cs.CV] <https://arxiv.org/abs/2307.05663>

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. 2022. Objaverse: A Universe of Annotated 3D Objects. *arXiv:2212.08051* [cs.CV] <https://arxiv.org/abs/2212.08051>

Kangle Deng, Hsueh-Ti Derek Liu, Yiheng Zhu, Xiaoxia Sun, Chong Shang, Kiran Bhat, Deva Ramanan, Jun-Yan Zhu, Maneesh Agrawala, and Tinghui Zhou. 2025. Efficient Autoregressive Shape Generation via Octree-Based Adaptive Tokenization. *arXiv:2504.02817* [cs.CV] <https://arxiv.org/abs/2504.02817>

- Jingyu Guo, Sensen Gao, Jia-Wang Bian, Wanhu Sun, Heliang Zheng, Rongfei Jia, and Mingming Gong. 2025. Hyper3D: Efficient 3D Representation via Hybrid Triplane and Octree Feature for Enhanced 3D Shape Variational Auto-Encoders. *arXiv:2503.10403* [cs.CV] <https://arxiv.org/abs/2503.10403>
- William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. *CoRR* abs/1706.02216 (2017). *arXiv:1706.02216* <http://arxiv.org/abs/1706.02216>
- Zekun Hao, David W Romero, Tsung-Yi Lin, and Ming-Yu Liu. 2024. Meshtron: High-fidelity, artist-like 3d mesh generation at scale. *arXiv preprint arXiv:2412.09548* (2024).
- Xianglong He, Junyi Chen, Di Huang, Zexiang Liu, Xiaoshui Huang, Wanli Ouyang, Chun Yuan, and Yangguang Li. 2025. Meshcraft: Exploring efficient and controllable mesh generation with flow-based dits. *arXiv preprint arXiv:2503.23022* (2025).
- Jeonghwan Kim, Yushi Lan, Armando Fortes, Yongwei Chen, and Xingang Pan. 2025. FastMesh: Efficient Artistic Mesh Generation via Component Decoupling. *arXiv:2508.19188* <https://arxiv.org/abs/2508.19188>
- Diederik P Kingma and Max Welling. 2022. Auto-Encoding Variational Bayes. *arXiv:1312.6114* [stat.ML] <https://arxiv.org/abs/1312.6114>
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Zejiang Lai, Yunfei Zhao, Zibo Zhao, Haolin Liu, Qingxiang Lin, Jingwei Huang, Chunchao Guo, and Xiangyu Yue. 2025a. LATTICE: Democratize High-Fidelity 3D Generation at Scale. *arXiv preprint arXiv:2512.03052* (2025).
- Zejiang Lai, Yunfei Zhao, Zibo Zhao, Haolin Liu, Qingxiang Lin, Jingwei Huang, Chunchao Guo, and Xiangyu Yue. 2025b. LATTICE: Democratize High-Fidelity 3D Generation at Scale. *arXiv:2512.03052* [cs.GR] <https://arxiv.org/abs/2512.03052>
- Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, et al. 2025. Tripog: High-fidelity 3d shape synthesis using large-scale rectified flow models. *arXiv preprint arXiv:2502.06608* (2025).
- Stefan Lionar, Jiabin Liang, and Gim Hee Lee. 2025. Treemeshgpt: Artistic mesh generation with autoregressive tree sequencing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 26608–26617.
- Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. 2023. Flow Matching for Generative Modeling. In *ICLR*. OpenReview.net.
- Jiuming Liu, Ruiji Yu, Yian Wang, Yu Zheng, Tianchen Deng, Weicai Ye, and Hesheng Wang. 2024. Point Mamba: A Novel Point Cloud Backbone Based on State Space Model with Octree-Based Ordering Strategy. *arXiv:2403.06467* [cs.CV]
- Xingchao Liu, Chengyue Gong, and Qiang Liu. 2023. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *ICLR*. OpenReview.net.
- Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. 2022. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. *arXiv:2206.00927* [cs.LG] <https://arxiv.org/abs/2206.00927>
- Donald Meagher. 1982. Geometric modeling using octree encoding. *Computer graphics and image processing* 19, 2 (1982), 129–147.
- Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. 2020. Polygen: An autoregressive generative model of 3d meshes. In *International conference on machine learning*. PMLR, 7220–7229.
- William Peebles and Saining Xie. 2023. Scalable Diffusion Models with Transformers. *arXiv:2212.09748* [cs.CV] <https://arxiv.org/abs/2212.09748>
- Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. 2024. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *arXiv preprint arXiv:2403.17898* (2024).
- Tianchang Shen, Zhaoshuo Li, Marc Law, Matan Atzmon, Sanja Fidler, James Lucas, Jun Gao, and Nicholas Sharp. 2024. Spacemesh: A continuous representation for learning manifold surface meshes. In *SIGGRAPH Asia 2024 Conference Papers*. 1–11.
- Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. 2024. Meshgpt: Generating triangle meshes with decoder-only transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 19615–19625.
- Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. 2025. DINOv3. *arXiv:2508.10104* [cs.CV] <https://arxiv.org/abs/2508.10104>
- Stefan Stojanov, Anh Thai, and James M. Rehg. 2021. Using Shape to Categorize: Low-Shot Learning with an Explicit Shape Bias. *arXiv:2101.07296* [cs.CV] <https://arxiv.org/abs/2101.07296>
- Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2023. RoFormer: Enhanced Transformer with Rotary Position Embedding. *arXiv:2104.09864* [cs.CL] <https://arxiv.org/abs/2104.09864>
- Jiaxiang Tang, Zhaoshuo Li, Zekun Hao, Xian Liu, Gang Zeng, Ming-Yu Liu, and Qinsheng Zhang. 2024. Edgerunner: Auto-regressive auto-encoder for artistic mesh generation. *arXiv preprint arXiv:2409.18114* (2024).
- Peng-Shuai Wang. 2023. OctFormer: Octree-based Transformers for 3D Point Clouds. *ACM Transactions on Graphics (SIGGRAPH)* 42, 4 (2023).
- Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. 2017. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 72.
- Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. 2018. Adaptive O-CNN: a patch-based deep representation of 3D shapes. In *SIGGRAPH Asia 2018 Technical Papers*. ACM, 217.
- Si-Tong Wei, Rui-Huan Wang, Chuan-Zhi Zhou, Baoquan Chen, and Peng-Shuai Wang. 2025. Octgpt: Octree-based multiscale autoregressive models for 3d shape generation. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*. 1–11.
- Haohan Weng, Zibo Zhao, Biwen Lei, Xianghui Yang, Jian Liu, Zejiang Lai, Zhuo Chen, Yuhong Liu, Jie Jiang, Chunchao Guo, et al. 2025. Scaling mesh generation via compressive tokenization. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 11093–11103.
- Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. 2025. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 21469–21480.
- Bojun Xiong, Si-Tong Wei, Xin-Yang Zheng, Yan-Pei Cao, Zhouhui Lian, and Peng-Shuai Wang. 2025. OctFusion: Octree-based Diffusion Models for 3D Shape Generation. *Computer Graphics Forum* (2025). doi:10.1111/cgf.70198
- Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. 2023. ULIP: Learning a Unified Representation of Language, Images, and Point Clouds for 3D Understanding. *arXiv:2212.05171* [cs.CV] <https://arxiv.org/abs/2212.05171>
- Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 2023. 3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models. *arXiv:2301.11445* [cs.CV] <https://arxiv.org/abs/2301.11445>
- Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. 2024. Clay: A controllable large-scale generative model for creating high-quality 3d assets. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–20.
- Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. 2024. SGLang: Efficient Execution of Structured Language Model Programs. *arXiv:2312.07104* [cs.AI] <https://arxiv.org/abs/2312.07104>
- Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. 2023. Uni3D: Exploring Unified 3D Representation at Scale. *arXiv:2310.06773* [cs.CV] <https://arxiv.org/abs/2310.06773>