Native and Compact Structured Latents for 3D Generation
(Supplementary Material)

Contents

A. More Implementation Details **1**
- A.1. O-Voxel Conversion Algorithms . . . 1
  - A.1.1. Shape Conversion . . . 1
  - A.1.2. Material Conversion . . . 1
- A.2. Network Architectures . . . 1
- A.3. Training Details . . . 2

B. FlexGEMM: Our High-Performance Sparse Convolution Backend **3**

C. Data Preparation Details **4**

D. More Experiment Details **5**
- D.1. Evaluation Protocol . . . 5
  - D.1.1. Reconstruction Experiments . . . 5
  - D.1.2. Generation Experiments . . . 6
- D.2. User Study . . . 6

E. More Results **7**
- E.1. 3D Asset Reconstruction . . . 7
- E.2. Image to 3D Asset Generation . . . 7

F. Limitation Discussion and Future Work **8**

A. More Implementation Details

A.1. O-Voxel Conversion Algorithms

This section provides a detailed breakdown of the bidirectional conversion algorithms between standard 3D assets (meshes and PBR textures) and our O-Voxel representation. We present the process in four parts: converting a mesh to the O-Voxel shape representation, reconstructing a mesh from it, converting PBR textures to the O-Voxel material representation, and reconstructing textures from it.

A.1.1. Shape Conversion

The geometric component of O-Voxel is based on our *Flexible Dual Grid* formulation. Algo. 1 and 2 detail its conversion to and from a triangle mesh.

A.1.2. Material Conversion

The material component of O-Voxel stores PBR attributes in active voxels. The conversion to and from standard mesh textures is a direct sampling and interpolation process (Algo. 3 and 4).

Algorithm 1: Mesh-to-O-Voxel Conversion

Input: Input mesh $\mathcal{M}$, grid resolution N, weights $\lambda_{\text{bound}}, \lambda_{\text{reg}}$
Output: O-Voxel shape features $\boldsymbol{f}^{\text{shape}}$

▷1. Initialize a map to store data for each active voxel
voxel_data ← EmptyMap[$\boldsymbol{p} \to \{\text{QEF}, \bar{\boldsymbol{q}}, \boldsymbol{\delta}\}$]

▷2. Accumulate plane-distance QEFs from triangle intersections
for *each triangle* T *in* $\mathcal{M}$ **do**
  for *each voxel edge* e *intersected by* T **do**
    $\{\boldsymbol{q}, \boldsymbol{n}\} \leftarrow \text{GetIntersectionAndNormal}(T, e)$
    for *each neighboring voxel* V *of edge* e **do**
      $\boldsymbol{p} \leftarrow \text{GetCoordinate}(V)$
      Initialize voxel_data[$\boldsymbol{p}$] if not exists
      plane_qef ← BuildPlaneQEF($\boldsymbol{q}, \boldsymbol{n}$)
      voxel_data[$\boldsymbol{p}$].QEF.acc(plane_qef)
      voxel_data[$\boldsymbol{p}$].$\bar{\boldsymbol{q}}$.acc($\boldsymbol{q}$)
      voxel_data[$\boldsymbol{p}$].$\boldsymbol{\delta}$.update(e)

▷3. Accumulate boundary-distance QEFs from open mesh edges
for *each boundary edge* b *in* $\mathcal{M}$ **do**
  for *each voxel* V *intersected by* b **do**
    $\boldsymbol{p} \leftarrow \text{GetCoordinate}(V)$
    if $\boldsymbol{p}$ *in voxel_data* **then**
      $\{\boldsymbol{o}, \boldsymbol{d}\} \leftarrow \text{GetLineParameters}(b)$
      line_qef ← BuildLineQEF($\boldsymbol{o}, \boldsymbol{d}$)
      voxel_data[$\boldsymbol{p}$].QEF.acc($\lambda_{\text{bound}} \cdot$ line_qef)

▷4. Accumulate regularization-term QEFs
for *each* $\boldsymbol{p}$ *in voxel_data* **do**
  reg_qef ← BuildPointQEF(voxel_data[$\boldsymbol{p}$].$\bar{\boldsymbol{q}}$)
  voxel_data[$\boldsymbol{p}$].QEF.acc($\lambda_{\text{reg}} \cdot$ reg_qef)

▷5. Solve QEFs and finalize O-Voxel features
$\boldsymbol{f}^{\text{shape}} \leftarrow$ EmptyMap[]
for *each* $\boldsymbol{p}$*, data in voxel_data* **do**
  $\boldsymbol{v} \leftarrow$ SolveQEF(data.QEF)
  $\boldsymbol{\delta} \leftarrow$ data.$\boldsymbol{\delta}$
  $\gamma \leftarrow 0.5$
  $\boldsymbol{f}^{\text{shape}}[\boldsymbol{p}] \leftarrow \{\boldsymbol{v}, \boldsymbol{\delta}, \gamma\}$
return $\boldsymbol{f}^{\text{shape}}$

A.2. Network Architectures

Sparse Compression VAE The Sparse Compression VAE (SC-VAE) is a *fully sparse-convolutional network* designed to compress the O-Voxel representation into a compact latent space with a 16× spatial downsampling ratio. We employ a conventional U-Shaped VAE architecture, optimized with ConvNeXt-style [14] residual blocks and Residual AutoEncoding layers [1] for (down/up)sampling. The detailed architecture for the SC-VAE encoder is presented in Table 1. The decoder is constructed symmetrically using inverted block numbers and dimensions. The complete model comprises ∼800M parameters (354M for the encoder and 474M for the decoder). This configuration achieves near-lossless reconstruction fidelity while maintaining high computational efficiency.

Generative Models. Our generation framework consists of three Transformer-based models. These models adopt a standard encoder-only architecture, intentionally omitting complex designs such as token packing or skip connections to maintain a clean and scalable architecture (shown in Table 2). Conditional inputs are integrated using mechanisms tailored to the nature of the data:

- **Timestep Injection:** We use the *AdaLN-single* [2] scheme for the conditioning on diffusion timestep. This method modulates the activations within the network, allowing the model to effectively incorporate temporal information while drastically reducing required parameters compared to the AdaLN baseline.
- **Image Prompt Conditioning:** Image prompts are integrated via *cross-attention* layers. This enables the model to align its generative process with the semantic content of the visual conditioning signal.
- **Shape Conditioning:** For the material generation stage, shape information is provided as a condition by *concatenating* it channel-wise with the input tensor. This direct approach ensures that geometric constraints are explicitly

Algorithm 2: O-Voxel-to-Mesh Conversion

Input: O-Voxel shape features $\boldsymbol{f}^{\text{shape}}$
Output: Reconstructed mesh $\mathcal{M}'$

▷1. Create a mesh vertex for each dual vertex in the O-Voxel data
$V' \leftarrow$ EmptyList[]
vertex_indices $\leftarrow$ EmptyMap[$\boldsymbol{p} \rightarrow$ index]
for *each* $\boldsymbol{p}$*, data in* $\boldsymbol{f}^{\text{shape}}$ **do**
  V'.append(data.$\boldsymbol{v}$)
  vertex_indices[$\boldsymbol{p}$] $\leftarrow |V'| - 1$

▷2. Generate faces by connecting vertices across active edges
$F' \leftarrow$ EmptyList[]
for *each* $\boldsymbol{p}$*, data in* $\boldsymbol{f}^{\text{shape}}$ **do**
  ▷Iterate over the 3 predefined axes to avoid duplicate faces
  for *each axis* $a \in \{X, Y, Z\}$ **do**
    if *data.*$\boldsymbol{\delta}[a] == 1$ **then**
      quad_coords $\leftarrow$ GetQuadVoxel($\boldsymbol{p}$, a)
      ▷Ensure all four voxels are active
      if *all quad_coords exist in* $\boldsymbol{f}^{\text{shape}}$ **then**
        $i_0, i_1, i_2, i_3 \leftarrow$ vertex_indices[quad_coords]
        ▷Split the quadrilateral into two triangles
        $t_1, t_2 \leftarrow$ Split($\{i_0, i_1, i_2, i_3\}$, data.γ)
        F'.extend($[t_1, t_2]$)

▷3. Construct the final mesh from vertices and faces
$\mathcal{M}' \leftarrow$ Mesh(V', F')
return $\mathcal{M}'$

Algorithm 3: Texture-to-O-Voxel Conversion

Input: Input mesh $\mathcal{M}$ with PBR textures, O-Voxel shape features $\boldsymbol{f}^{\text{shape}}$
Output: O-Voxel material features $\boldsymbol{f}^{\text{mat}}$

▷1. Initialize an empty map for material features
$\boldsymbol{f}^{\text{mat}} \leftarrow$ EmptyMap[$\boldsymbol{p} \rightarrow \{\boldsymbol{c}, m, r, \alpha\}$]
▷2. For each active voxel, sample material attributes from the mesh
for *each voxel coordinate* $\boldsymbol{p}$ *in* $\boldsymbol{f}^{\text{shape}}$ **do**
  $\boldsymbol{p}_{\text{center}} \leftarrow$ GetVoxelCenter($\boldsymbol{p}$)
  intersecting_tris $\leftarrow$ FindIntersectingTriangles($\mathcal{M}$, $\boldsymbol{p}$)
  ▷Collect weighted samples from all intersecting triangles
  samples $\leftarrow$ EmptyList[]
  weights $\leftarrow$ EmptyList[]
  for *each triangle* T *in intersecting_tris* **do**
    $\boldsymbol{q} \leftarrow$ ProjectPointOntoTriangle($\boldsymbol{p}_{\text{center}}$, T)
    $d \leftarrow \|\boldsymbol{p}_{\text{center}}, \boldsymbol{q}\|_2$
    $w \leftarrow 1 - d$
    mip_level $\leftarrow$ GetMipLevel(T, voxel_size)
    $\boldsymbol{uv} \leftarrow$ GetUVCoordinates($\boldsymbol{q}$, T)
    pbr_sample $\leftarrow$ SampleTexture($\mathcal{M}$.textures, $\boldsymbol{uv}$, mip_level)
    samples.append(pbr_sample)
    weights.append(w)
  ▷Compute the final feature via weighted average
  $\boldsymbol{f}^{\text{mat}}[\boldsymbol{p}] \leftarrow$WeightedAverage(samples, weights)
return $\boldsymbol{f}^{\text{mat}}$

provided, helping to improve material-shape alignment. To enhance generalization across different input resolutions, we incorporate *Rotary Position Embedding (RoPE)* [19]. Furthermore, we employ a *QK-Norm* scheme [8] to stabilize the attention mechanism. This involves applying Root Mean Square Normalization (RMSNorm) [25] to the query and key tensors before the attention operation, which improves training stability.

A.3. Training Details

Sparse Compression VAE. As described in the main paper, the SC-VAE is trained using a two-stage strategy. The first stage focuses on stabilizing the training process by employing a direct O-Voxel feature regression loss at a resolution of 256^3. In the second stage, resolution is increased to 512^3 while rendering-based perceptual loss is introduced to enhance visual quality, such as geometric sharpness and high-frequency material details, and to facilitate the model's adaptation to higher resolutions. This rendering loss is implemented as follows:

$$
\begin{aligned}
d_{\text{p}}(\boldsymbol{a}, \boldsymbol{b}) &= \|\boldsymbol{a} - \boldsymbol{b}\|_1 + 0.2 \cdot d_{\text{SSIM}} + 0.2 \cdot d_{\text{LPIPS}} \\
\mathcal{L}^{\text{shape}}_{\text{render}} &= \|\hat{m} - m\|_1 + 10 \cdot \|\hat{d} - d\|_1 + d_{\text{p}}(\hat{\boldsymbol{n}}, \boldsymbol{n}) \\
\mathcal{L}^{\text{mat}}_{\text{render}} &= d_{\text{p}}(\hat{\boldsymbol{c}}, \boldsymbol{c}) + d_{\text{p}}(\hat{\boldsymbol{mra}}, \boldsymbol{mra})
\end{aligned}
\tag{1}
$$

where $\mathcal{L}^{\text{shape}}_{\text{render}}$ and $\mathcal{L}^{\text{mat}}_{\text{render}}$ are the rendering losses for shape and material, respectively. The term $d_{\text{p}}(\cdot, \cdot)$ denotes a perceptual distance metric combining the L1 norm with SSIM and LPIPS losses. In these equations, variables with a hat ($\hat{\cdot}$) represent model predictions, while variables without are the ground-truth targets. Specifically, m is the silhouette mask, d is the depth map, $\boldsymbol{n}$ is the normal map, $\boldsymbol{c}$ is the base color, and $\boldsymbol{mra}$ corresponds to the metallic-

Algorithm 4: O-Voxel-to-Texture Conversion

Input: Reconstructed mesh $\mathcal{M}'$, O-Voxel material features $\boldsymbol{f}^{\text{mat}}$, mode $\in$ {'vertex', 'map'}
Output: Mesh $\mathcal{M}'$ with PBR materials applied

if *mode == 'vertex'* **then**
  ▷1. Generate vertex colors via trilinear interpolation
  vertex_materials ← EmptyList[]
  for *each vertex* $\boldsymbol{v}$ *in* $\mathcal{M}'$ **do**
    material ← TrilinearInterp($\boldsymbol{v}$, $\boldsymbol{f}^{\text{mat}}$)
    vertex_materials.append(material)
  ApplyVertexMaterials($\mathcal{M}'$, vertex_materials)
else if *mode == 'map'* **then**
  ▷2. Generate texture maps by filling interpolated values
  texture_maps ← Parameterize($\mathcal{M}'$)
  for *each texel* (u, v) *in texture_maps* **do**
    $\boldsymbol{q}$ ← GetSurfacePointFromUV($\mathcal{M}'$, u, v)
    material ← TrilinearInterp($\boldsymbol{q}$, $\boldsymbol{f}^{\text{mat}}$)
    texture_maps$[u, v]$ ← material
  ApplyTextureMaps($\mathcal{M}'$, texture_maps)
return $\mathcal{M}'$

Table 1. Architectural details of the SC-VAE encoder. The decoder follows a symmetrical design.

Stage (f_{down})	Block
1×	$\text{Linear}(6, 64)$ $\text{ResEnc}(64, 128)$
2×	$\begin{bmatrix}\text{SubMConv}(3, 128, 128)\\ \text{LayerNorm}\\ \text{Linear}(128, 512)\\ \text{SiLU}\\ \text{Linear}(512, 128)\end{bmatrix} \times 4$ $\text{ResEnc}(128, 256)$
4×	$\begin{bmatrix}\text{SubMConv}(3, 256, 256)\\ \text{LayerNorm}\\ \text{Linear}(256, 1024)\\ \text{SiLU}\\ \text{Linear}(1024, 256)\end{bmatrix} \times 8$ $\text{ResEnc}(256, 512)$
8×	$\begin{bmatrix}\text{SubMConv}(3, 512, 512)\\ \text{LayerNorm}\\ \text{Linear}(512, 2048)\\ \text{SiLU}\\ \text{Linear}(2048, 512)\end{bmatrix} \times 16$ $\text{ResEnc}(512, 1024)$
16×	$\begin{bmatrix}\text{SubMConv}(3, 1024, 1024)\\ \text{LayerNorm}\\ \text{Linear}(1024, 4096)\\ \text{SiLU}\\ \text{Linear}(4096, 1024)\end{bmatrix} \times 4$ $\text{Linear}(1024, 32 \times 2)$

roughness-alpha map.

For inputs with resolutions exceeding 512^3, we directly apply the pre-trained SC-VAE models without modification.

Table 2. Architectural Details for the Generative models.

Stage	Block
In_proj	$\text{Linear}(32(+32), 1536)$
Stem	$\begin{bmatrix}\text{AdaLN-single}\\ \text{SelfAttn}(12 \times 128)\\ \text{LayerNorm}\\ \text{CrossAttn}(12 \times 128)\\ \text{AdaLN-single}\\ \text{FFN}(1536, 8192)\end{bmatrix} \times 30$
Out_proj	LayerNorm $\text{Linear}(1536, 32)$

The fully sparse-convolutional design of the SC-VAE is inherently resolution-agnostic, a property that allows the models to generalize effectively to larger spatial resolutions without requiring fine-tuning.

Generative Models. We employ the *rectified flow* formulation [13] to train our generative models. This framework defines a forward process based on linear interpolation, $\boldsymbol{x}(t) = (1-t)\boldsymbol{x}_0 + t\boldsymbol{\epsilon}$, which constructs a straight path from a data sample $\boldsymbol{x}_0$ to a random noise sample $\boldsymbol{\epsilon}$, indexed by timestep $t \in [0, 1]$.

The corresponding reverse process is governed by a time-dependent vector field, $\boldsymbol{v}(\boldsymbol{x}, t) = \nabla_t \boldsymbol{x}$, which guides samples from the noise distribution back toward the data distribution. This vector field is approximated by a neural network, denoted $\boldsymbol{v}_\theta$, which is trained by minimizing the *Conditional Flow Matching (CFM)* objective [12]:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t, \boldsymbol{x}_0, \boldsymbol{\epsilon}} \|\boldsymbol{v}_\theta(\boldsymbol{x}(t), t) - (\boldsymbol{\epsilon} - \boldsymbol{x}_0)\|_2^2. \quad (2)$$

Following the approach of [23], we adopt an altered timestep sampling strategy, utilizing a $\text{logitNorm}(1, 1)$ distribution for better generation quality.

B. FlexGEMM: Our High-Performance Sparse Convolution Backend

The sparse convolutional networks in our model are accelerated by a *custom* high-performance backend developed for this work. This backend was engineered to address the performance and platform-dependency limitations of existing libraries, which are often tightly coupled to the NVIDIA CUDA ecosystem. By implementing our kernels in *Triton* [21], a high-level GPU programming language, we created a single, cross-platform codebase that delivers near-optimal performance on both NVIDIA and AMD hardware.

Our final, optimized implementation employs a *Masked Implicit GEMM* strategy [6]. This approach moves beyond naive explicit matrix multiplication by fusing the feature gathering (im2col) and the matrix multiplication (GEMM) steps into a single, highly-optimized kernel. This fusion minimizes global memory I/O by keeping intermediate data in fast on-chip memory. To further enhance performance in

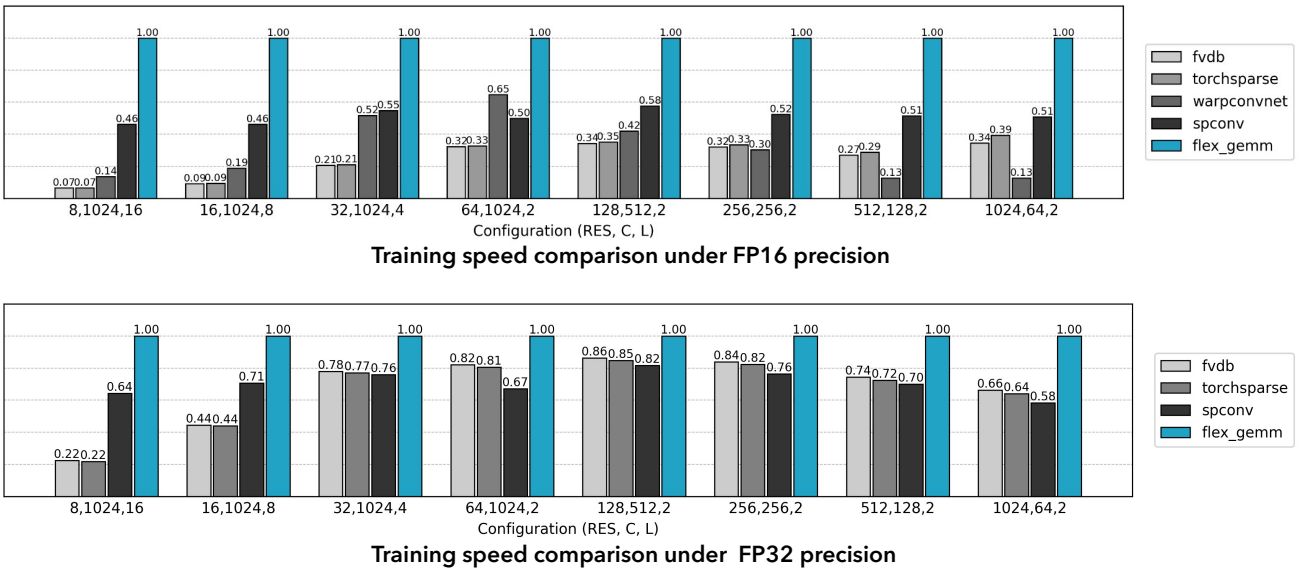

Figure 1. Speed test for FlexGEMM backend and baselines including Spconv [6], Torchsparse [20], fvdb [22], and WarpConvNet [3]

Table 3. Composition of the training set and evaluation set.

Source	Shape Available	Material Available
TexVerse [26]	503387	382996
ObjaverseXL (sketchfab) [7]	168307	141623
ObjaverseXL (github) [7]	293887	202188
ABO [4]	4485	4485
HSSD [11]	6670	6670
SC-VAE training set	473349	354966
All training set	976736	737962
Toys4k [18] (evaluation set)	3229	2282

sparse contexts, we introduce a masking mechanism that intelligently skips computation on empty neighbor slots. This is achieved by first reordering active voxels using *Gray code ordering*, a technique that groups voxels with similar neighborhood patterns together. This grouping significantly improves the SIMD efficiency of the GPU, reducing warp divergence and wasted computation. Finally, we incorporate a *Split-K* technique, which increases parallelism by dividing the accumulation dimension of the matrix multiplication into independent parallel tasks. This is particularly effective in common scenarios, such as those with a large number of channels or a small number of active voxels. The combination of these techniques results in a highly efficient backend, yielding up to a *2× speedup over widely-used sparse convolution libraries* in our benchmarking (see Fig. 1).

C. Data Preparation Details

The data preparation pipeline is largely based on the setup proposed in TRELLIS [23]. We begin by curating a collection of 3D assets but exclude the 3D-FUTURE [10] dataset due to its lack of Physically-Based Rendering (PBR) materials. The remaining assets form the basis for training our SC-VAEs.

All assets in this curated collection are used to extract geometric data for training the shape SC-VAE. For the material SC-VAE, a more specific filtering process is required. We employ a custom Blender [5] script to parse materials from the raw assets and retain only those that utilize a standard metallic-roughness PBR workflow. This filtering process yields a subset of approximately 350,000 assets suitable for training the material VAE.

To train the generative models, we further augment the dataset with TexVerse [26] to increase the diversity of high-quality PBR materials. As a final quality control step, we filter the assets based on an aesthetic score. For simplicity, we leverage the thumbnail images provided on the Sketchfab [17] platform to estimate this score. Objects with an estimated aesthetic score below 4.5 are excluded from the training set. Detailed statistics of the final dataset are provided in Table 3.

To generate the image prompts required for training our image-conditioned model, we render a diverse set of views for each 3D asset using Blender. We apply a series of augmentations during this rendering process to ensure the model is robust against common ambiguities found in real-world inputs. Key augmentations include:

- **Field of View (FoV):** The camera's Field of View (FoV) is randomly sampled between $10°$ and $70°$. This augmentation is designed to make the model robust to variations in camera intrinsics, which are often unknown in practice.
- **Lighting Conditions:** The lighting environment is randomized by randomly placing and adjusting the intensity of light sources. This improves the model's ability to predict intrinsic PBR attributes accurately, disentangling them from environmental illumination.

D. More Experiment Details

D.1. Evaluation Protocol

In the main paper, we present quantitative comparisons and ablation studies using a series of numerical metrics. We provide the detailed protocols for their calculation below.

D.1.1. Reconstruction Experiments

Test set. To ensure a robust evaluation of reconstruction quality, we prepared two distinct test sets.

- *Toys4k-PBR.* Our first test set is derived from the Toys4k dataset. For a rigorous metric, we filtered the raw assets to include only those containing all three standard PBR maps (base color, metallic, and roughness). This process resulted in a refined test set of 473 instances.
- *Sketchfab Featured.* Recognizing that the assets in Toys4k are relatively simple, we curated a second, more challenging test set from high-quality, recent assets on Sketchfab. Specifically, we selected models from the "Staff Picks" category, which features professionally curated content. We then applied a filter to retain only assets that utilize the metallic-roughness PBR workflow and were uploaded within the last two years. This process yielded a high-quality test set comprising 90 instances, designed to evaluate performance on complex, professional-grade assets.

Geometry Accuracy. To assess the overall geometric fidelity, we use *Mesh Distance* and the corresponding *F-score*. Unlike Chamfer Distance, which is sensitive to point cloud density, Mesh Distance provides a more stable measure of the discrepancy between two triangle meshes. This makes it particularly suitable for evaluating reconstruction accuracy across all surfaces, including those that are fully enclosed. For this evaluation, we sample 1 million points from the surface of each mesh. For the F-score calculation, we use a distance threshold of $\tau = 1 \times 10^{-8}$.

For evaluating the accuracy of visible surfaces, we compute *Chamfer Distance (CD)* and the corresponding *F-score*. The evaluation is performed on point clouds generated by sampling the outer shell of the meshes. Specifically, we render depth maps for each mesh from 100 uniformly sampled camera views. These depth maps are then unprojected to create a dense 3D point cloud, from which we randomly sample 1 million points. For the F-score calculation, we use a distance threshold of $\tau = 1 \times 10^{-6}$.

To evaluate the quality of fine surface details, we compute *PSNR* and *LPIPS* on rendered normal maps. For this, we render images from four fixed camera positions for all assets. The camera is placed on a sphere of radius 10 with a fixed pitch angle of 30° and a narrow Field of View (FoV) of 6°. The four views correspond to yaw angles of $30^\circ, 120^\circ, 210^\circ$, and 300°.

Prior to any metric calculation, all ground-truth and predicted meshes are normalized to fit within a unit cube. The definitions for the geometric metrics are as follows:

- *Mesh Distance (MD).* MD is calculated as the bidirectional point-to-mesh surface distance, averaged over a dense sampling of points from both meshes. Given two meshes S_X and S_Y, with sampled points P_X and P_Y, MD is defined as:

$$\begin{aligned} \text{MD}(S_X, S_Y) = &\frac{1}{2|P_X|} \sum_{\boldsymbol{x} \in P_X} \min_{\boldsymbol{y} \in S_Y} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \\ &+ \frac{1}{2|P_Y|} \sum_{\boldsymbol{y} \in P_Y} \min_{\boldsymbol{x} \in S_X} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2. \end{aligned} \tag{3}$$

- *Chamfer Distance (CD).* Given two point clouds, X and Y, the Chamfer Distance is defined as:

$$\begin{aligned} \text{CD}(X, Y) = &\frac{1}{2|X|} \sum_{\boldsymbol{x} \in X} \min_{\boldsymbol{y} \in Y} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \\ &+ \frac{1}{2|Y|} \sum_{\boldsymbol{y} \in Y} \min_{\boldsymbol{x} \in X} \|\boldsymbol{y} - \boldsymbol{x}\|_2^2. \end{aligned} \tag{4}$$

- *F-score.* The F-score evaluates shape correspondence by combining precision and recall, calculated based on a distance threshold τ. Given a ground-truth shape S_{gt} and a predicted shape S_{pred}, we sample point sets P_{gt} from S_{gt} and P_{pred} from S_{pred}. Precision and Recall are then defined as:

$$\begin{aligned} \text{Prec}(\tau) &= \frac{1}{|P_{pred}|} \sum_{p \in P_{pred}} \mathbb{I}\left(d^2(p, S_{gt}) < \tau\right), \\ \text{Rec}(\tau) &= \frac{1}{|P_{gt}|} \sum_{p \in P_{gt}} \mathbb{I}\left(d^2(p, S_{pred}) < \tau\right), \end{aligned} \tag{5}$$

where $\mathbb{I}(\cdot)$ is the indicator function and $d(p, S)$ is the minimum Euclidean distance from a point p to the shape S. The F-score is the harmonic mean of these values:

$$\mathbf{F}\text{-score}(\tau) = \frac{2 \cdot \text{Prec}(\tau) \cdot \text{Rec}(\tau)}{\text{Prec}(\tau) + \text{Rec}(\tau)}. \tag{6}$$

The distance function $d(p, S)$ is defined differently depending on the context, described below.

  - For CD F-score: The shapes S_{gt} and S_{pred} are treated as discrete point clouds. The distance $d(p, S)$ is the Euclidean distance from point p to the nearest point within the point cloud S.
  - For MD F-score: The shapes S_{gt} and S_{pred} are treated as continuous triangle meshes. The distance $d(p, S)$ is the Euclidean distance from point p to the closest point on the surface of the mesh S.

Appearance Fidelity. To assess the quality of the reconstructed materials, we evaluate both the raw PBR attribute maps and the final shaded images. For both the ground-truth and the reconstructed 3D assets, we render two sets of images using the nvdiffrec renderer [15]. This rendering is performed using the same fixed-camera setup as the normal map evaluation, capturing four distinct views. The PSNR and LPIPS metrics are then calculated by comparing the rendered outputs from the reconstructed asset against those from the ground truth. The final reported scores are the average values across these four views.

D.1.2. Generation Experiments

Test Set. For quantitative evaluation of our image-to-3D generation capabilities, we conduct experiments on a challenging test set of 100 image prompts generated by the NanoBanana text-to-image model [9]. This dataset was specifically chosen for its diversity and complexity. It features prompts that describe objects with intricate geometries, varied and dramatic lighting conditions, and a wide range of realistic materials, including metal, leather, rust, and translucent substances such as glass.

Evaluation Metrics. We employ a suite of metrics targeting different aspects of the output. The visual and semantic alignment between the input image prompt and rendered images of the asset is measured using the *CLIP score* [16]. To evaluate how well the 3D geometry and appearance properties match the image prompt, we use the multimodal foundation models *ULIP-2* [24] and *Uni3D* [27]. Details of the metrics are listed below:

- *CLIP Score.* The CLIP score measures the semantic similarity between two images. In our evaluation, we render the generated 3D asset from 4 predefined viewpoints (same yaw, pitch setup as previous metrics). We then compute the average cosine similarity between the CLIP embedding of the input image prompt and the rendered images (or normal map). A higher CLIP score indicates a better semantic alignment between the conditional input and the appearance (or geometry) of the generated asset.
- *ULIP-2 and Uni3D Scores.* ULIP-2 and Uni3D are models designed to understand and align 3D content with text/image. To prepare the input for these models, we first convert our generated mesh into a colored point cloud. Specifically, we uniformly sample 10,000 points from the surface of the mesh with Farthest Point Sampling. The color for each point is determined by querying its corresponding RGB value from the asset's base color map. This colored point cloud is then fed into the ULIP-2 and Uni3D models to compute a similarity score against the image prompt. These scores provide a quantitative measure of how well the generated assets align with the condition from a native 3D perspective.

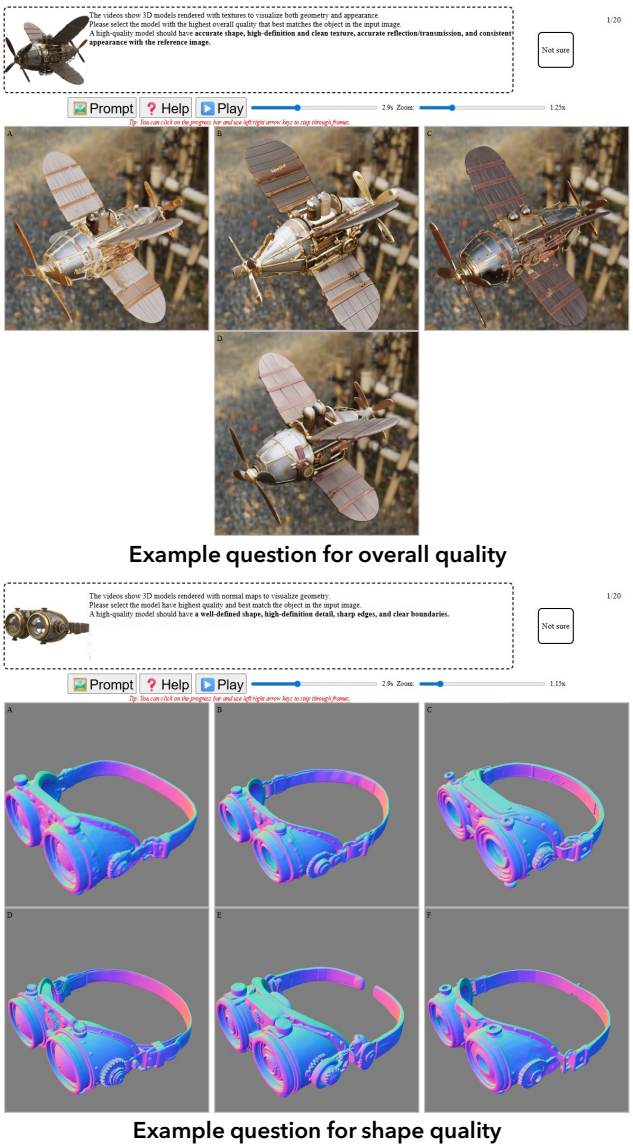

Figure 2. The interface for our user study. Participants were presented with two types of questions: one for evaluating the *overall quality* of fully rendered assets (top) and another for assessing the *shape quality* using normal map visualizations (bottom). The interface provided interactive controls for a thorough inspection.

D.2. User Study

While quantitative metrics provide objective measurements of fidelity, they often fail to capture the nuanced perceptual qualities that define a high-quality 3D asset, such as aesthetic appeal and fine-detail plausibility. To provide a comprehensive evaluation that aligns with human perception, we conducted a rigorous user study to compare our method against others.

Study Design and Interface. Our study was designed to assess two critical aspects of 3D asset quality: *overall qual-*

Table 4. Detailed statistics of the user study.

Method	Overall		Shape	
	Selections ↑	Percentage ↑	Selections ↑	Percentage ↑
Not Sure	4	2.0%	3	1.4%
TRELLIS	13	6.4%	6	2.8%
Hi3DGen	–	–	14	6.6%
Direct3D-S2	–	–	26	12.2%
Step1X-3D	24	11.8%	1	0.5%
Hunyuan3D 2.1	27	13.3%	16	7.5%
Ours	**135**	**66.5%**	**147**	**69.0%**
Total	202	100%	213	100%

ity (combining geometry and appearance) and *shape quality* (isolating geometric fidelity). Participants were presented with a series of choice questions through a custom web interface, as shown in Figure 2.

For each question, participants were shown a reference image and a set of turntable video renderings of the 3D models generated by different methods. The interface provided interactive controls, allowing users to play, pause, scrub through the animation timeline, and zoom in to inspect details closely. This ensured that participants could perform a thorough comparison. The positions of the generated models were randomized for each question to prevent positional bias.

The study consisted of two distinct types of questions:

- **Overall Quality Evaluation:** In this task, participants were shown fully textured and rendered 3D models. They were instructed to select the model with the "highest overall quality that best matches the object in the input image." The evaluation criteria emphasized a holistic assessment, including accurate shape, high-definition textures, realistic material properties (reflection and transmission), and consistent appearance with the reference.
- **Shape Quality Evaluation:** To specifically evaluate geometric accuracy without the confounding influence of materials, this task presented the models rendered with only a normal map. Participants were asked to select the model with the best shape, focusing on criteria such as "a well-defined shape, high-definition detail, sharp edges, and clear boundaries."

Detailed Analysis. We recruited about 40 participants in the evaluation. For each question, the model selected by a participant was recorded as a "win" over the other options presented. We aggregated these results from all participants and computed a global preference rate for each method. This percentage provides a clear ranking of perceptual quality. Detailed statistics of the user study are shown in Table 4.

E. More Results

E.1. 3D Asset Reconstruction

Additional Reconstruction Results. We present additional reconstruction results of our SC-VAE in Figure 3. The figure showcases the model's ability to achieve high-fidelity reconstruction across a diverse range of 3D assets. Our method successfully captures hard-surface mechanical objects (a combat mech), intricate thin structures (a shopping cart, a ferris wheel), open surfaces (a plant), words (a fridge), and complex material properties (a crystal). Despite the highly compact nature of the learned latent space, the model faithfully recovers both complex geometries, visualized via normal maps, and detailed PBR materials, shown in the final renders.

Additional Qualitative Comparisons. Figure 4 provides an extended qualitative comparison of shape reconstruction fidelity against several state-of-the-art methods. The comparison includes normal map renderings, magnified insets to highlight fine details, and corresponding error maps that visualize the deviation from the ground truth. Across all examples, our method consistently demonstrates a superior ability to preserve high-frequency geometric details. For instance, our model more accurately reconstructs the intricate chainmail links of the helmet and the sharp ornamental patterns on the decorative vessel, where other methods often produce overly smooth or blurry surfaces. Notably, as demonstrated in the final column, our method also excels at recovering enclosed internal structures, which pose a significant challenge for many surface reconstruction techniques. This high fidelity is further corroborated by the error maps, which show visibly lower reconstruction errors for our method across all examples when compared to the baselines.

E.2. Image to 3D Asset Generation

Additional Generation Results. We present additional qualitative results from our image-to-3D generation method in Figure 5. The figure demonstrates the model's versatility and robustness across a wide range of categories, including organic structures (a garden trellis with ivy), complex hard-surface machinery (a sci-fi pod, a bulldozer), and detailed characters (a dwarf blacksmith, a soldier). For each generated asset, we display the final physically-based render, the corresponding normal map to illustrate geometric detail, and a breakdown of the constituent PBR attribute maps along with relighting results. This comprehensive visualization highlights our method's ability to jointly generate not only high-fidelity geometry but also plausible PBR materials that respond correctly to novel lighting conditions.

Additional Qualitative Comparisons. In Figure 6, we provide further qualitative comparisons for the image-to-3D generation task against several recent state-of-the-art methods. A primary advantage of our method is its ability to generate high-quality PBR materials, a capability not present in several baselines such as Step1X-3D, TRELLIS, Direct3D-S2, and Hi3DGen. When comparing geometric fidelity via

the normal maps, our results consistently exhibit sharper and more coherent details. For example, our method more accurately captures the fine mechanical joints of the crab and the face of the character, whereas competing methods often produce results that are overly smoothed or contain noticeable artifacts. Furthermore, for methods that do produce PBR materials (Hunyuan3D 2.1), our approach generates textures that are visually more plausible and better aligned with the input prompts.

F. Limitation Discussion and Future Work

Despite the promising results, our method has several limitations that open avenues for future research.

First, similar to other voxel-based methods, O-Voxel's representation power is bounded by its spatial resolution. For detailed geometric features smaller than the voxel size, the Flexible Dual Grid formulation could produce aliasing artifacts. For example, when two parallel surfaces that are very close to each other intersect the same voxel, the QEF solver, by design, will place the dual vertex at a position that minimizes the error to both surfaces, often resulting in a vertex located between them rather than accurately on one. Similarly, the volumetric material attributes in such a voxel will be an average of the properties of both surfaces, leading to blurred appearance.

Second, we observe that the reconstructed and generated results sometimes contain small holes, though they can mostly be rectified with standard mesh post-processing techniques (e.g., hole filling). We attribute this issue to challenges in the sparse nature of our decoder, where ensuring a perfectly closed, manifold surface from the high-resolution sparse structure predicted by our decoder can be difficult. Improving the inherent stability of decoding process is an important area for improvement.

Finally, our O-Voxel is currently focused on geometry and material and it does not explicitly encode higher-level structural or semantic information. A significant direction for future research is to extend our representation to incorporate part-level segmentation and a graph-based topological structure. Such a structured representation would unlock an even wider range of downstream applications.

Figure 3. Reconstruction results of our method. Despite highly compact, it achieves high-fidelity recovery of complex shapes and materials.

Figure 4. Qualitative comparison of shape reconstruction fidelity. Error maps are shown on the bottom right.

Figure 5. More image-to-3D generation results of our method. PBR attributes and relightings are shown below. (***Best viewed with zoom***)

Figure 6. More comparisons of image-to-3D generation results. Rendering results and PBR attributes (if applicable) are shown below. (***Best viewed with zoom***)

References

[1] Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. *arXiv preprint arXiv:2410.10733*, 2024. 1

[2] Junsong Chen, YU Jincheng, GE Chongjian, Lewei Yao, Enze Xie, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. In *ICLR*, 2024. 1

[3] Chris Choy and NVIDIA Research. Warpconvnet: High-performance 3d deep learning library. https://github.com/NVlabs/warpconvnet, 2025. 4

[4] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21126–21136, 2022. 4

[5] Blender Online Community. *Blender — a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 4

[6] Spconv Contributors. Spconv: Spatially sparse convolution library. https://github.com/traveller59/spconv, 2022. 3, 4

[7] 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. *Advances in Neural Information Processing Systems*, 36, 2024. 4

[8] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024. 2

[9] Alisa Fortin, Guillaume Vernade, Kat Kampf, and Ammaar Reshi. Introducing gemini 2.5 flash image: Our state-of-the-art image model. https://developers.googleblog.com/en/introducing-gemini-2-5-flash-image/, 2025. Google Developer Blog. 6

[10] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision*, pages 1–25, 2021. 4

[11] Mukul Khanna*, Yongsen Mao*, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X. Chang, and Manolis Savva. Habitat Synthetic Scenes Dataset (HSSD-200): An Analysis of 3D Scene Scale and Realism Tradeoffs for ObjectGoal Navigation. *arXiv preprint*, 2023. 4

[12] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *ICLR*, 2023. 3

[13] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023. 3

[14] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022. 1

[15] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8280–8290, 2022. 6

[16] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 6

[17] Sketchfab. Sketchfab - the best 3d viewer on the web. https://sketchfab.com/, 2025. 4

[18] Stefan Stojanov, Anh Thai, and James M Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1798–1808, 2021. 4

[19] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. 2

[20] Haotian Tang, Shang Yang, Zhijian Liu, Ke Hong, Zhongming Yu, Xiuyu Li, Guohao Dai, Yu Wang, and Song Han. Torchsparse++: Efficient training and inference framework for sparse convolution on gpus. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2023. 4

[21] Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19, 2019. 3

[22] Francis Williams, Jiahui Huang, Jonathan Swartz, Gergely Klar, Vijay Thakkar, Matthew Cong, Xuanchi Ren, Ruilong Li, Clement Fuji-Tsang, Sanja Fidler, et al. fvdb: A deep-learning framework for sparse, large scale, and high performance spatial intelligence. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024. 4

[23] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21469–21480, 2025. 3, 4

[24] Le Xue, Ning Yu, Shu Zhang, Artemis Panagopoulou, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, et al. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27091–27101, 2024. 6

[25] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[26] Yibo Zhang, Li Zhang, Rui Ma, and Nan Cao. Texverse: A universe of 3d objects with high-resolution textures. *arXiv preprint arXiv:2508.10868*, 2025. 4

[27] Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3d: Exploring unified 3d representation at scale. *arXiv preprint arXiv:2310.06773*, 2023. 6