STRUCTURED VOXEL DIFFUSION FOR 3D GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent 3D generative models often rely on multiview intermediates or learned latent representations, separating the generated variables from the final geometry. We introduce GRAVER, an image-conditioned framework that combines compact latent modeling of coarse structure with direct generation of fine geometry in an explicit sparse representation. Direct high-resolution geometric generation is challenging because volumetric cost grows cubically while the surface signal is highly sparse. GRAVER addresses this challenge with a staged block-UDF formulation that predicts (i) active blocks on a global grid, (ii) local surface-support masks within those blocks, and (iii) fine unsigned distance values only near the predicted support. The first two stages organize sparse structure, whereas the final stage predicts explicit UDF values without a geometry autoencoder. This decomposition concentrates computation near surfaces, accommodates open and non-watertight geometry, and reconstructs meshes through sparse marching cubes. The same spatially indexed representation also enables native local editing by preserving selected support and UDF values, and supports a preliminary adaptation to blockwise PBR prediction. Experiments show competitive image–shape alignment, validate the support-conditioned fine-UDF design, and demonstrate strong preservation of fixed regions during editing.



Figure 1: GRAVER generates high-fidelity 3D geometry directly in a structured voxel representation.

1 INTRODUCTION

3D assets underpin applications in gaming, film, AR/VR, robotics, and industrial design, yet their creation and modification remain labor-intensive. A practical image-to-3D system should therefore support not only geometric synthesis but also subsequent manipulation: users need to inspect local structure, modify selected regions, and retain geometry that should remain unchanged. These requirements motivate a common question: can a generative model operate on a representation that is both geometrically expressive and directly editable in 3D?

Recent image-to-3D methods have made substantial progress through multiview generation and reconstruction (Liu et al., 2023; Shi et al., 2023; Liu et al., 2024; Hong et al., 2024), learned 3D

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

latent representations (Zhang et al., 2023; Zhao et al., 2023; Lan et al., 2024; Wu et al., 2024; Ren et al., 2024; Xiang et al., 2024), and joint modeling of geometry and appearance (Xiang et al., 2025; Zhang et al., 2024b; Team, 2025). These representations make high-dimensional generation tractable, but they introduce a separation between the variables being generated and the geometry ultimately manipulated. For latent-based pipelines, fine geometry must be recovered through a learned decoder; for view-based pipelines, it must be reconciled across projections. This raises two related challenges: preserving local detail through reconstruction and expressing spatially localized edits in the model's generative representation.

Generating directly in raw geometric space offers a complementary approach. When the generated variables are spatially indexed geometric quantities, supervision can act directly on the final field, and edits can be specified at explicit 3D locations. However, this alignment comes at a substantial computational cost: dense volumetric storage grows cubically with resolution, although the surface occupies only a small fraction of the domain. The representation must also accommodate geometry for which a consistent inside–outside assignment is inconvenient, including open and non-watertight surfaces. Unsigned distance fields (UDFs) provide a suitable geometric description without requiring such an assignment (Long et al., 2022; Guillard et al., 2021). The challenge is thus to retain direct access to fine geometry without paying the cost of dense high-resolution generation.

We introduce GRAVER, an image-conditioned framework that combines compact structural generation with *raw-space geometric synthesis*. Its central design separates *where to generate* from *what geometry to generate*. A coarse stage establishes active blocks, a support stage generates sparse latents and decodes them into local surface support, and a final stage directly models fine block-wise UDF values through support-conditioned flow matching. Crucially, learned compression is used for the coarse layout and surface support, while fine distance values remain the generative state of the final stage rather than the output of a geometry autoencoder. A clean-state prediction objective supervises these values directly, and support-restricted dynamics keep unsupported regions at a deterministic far-field value. The resulting field is converted into a mesh by sparse marching cubes.

The same representation also provides an interface for *native 3D editing*. Because block coordinates, support masks, and distance values have explicit spatial meaning, users can modify support, compose geometric regions, and retain selected UDF blocks while synthesizing edited content. Editing therefore acts on the geometric representation used for generation, rather than requiring an edit to be expressed solely through image projections or latent variables. This connection between synthesis and manipulation is central to our approach: explicit geometry is not merely the final output, but also the space in which fine generation and spatial control are defined.

Contributions.

1. **Raw-space fine geometry generation.** We introduce a support-conditioned flow model that directly generates explicit UDF values, placing fine geometric supervision in the generated field rather than behind a learned geometry decoder.
2. **Native 3D editing.** We expose a spatially addressable representation for support manipulation, geometric composition, and selective UDF preservation, connecting image-conditioned synthesis with explicit control over editable regions.
3. **Hierarchical support–geometry factorization.** We combine compact latent support generation with explicit block-UDF refinement, concentrating high-resolution prediction on active blocks and restricting stochastic field evolution to their near-surface support.

2 Related Work

2.1 Image-conditioned 3D generation

Image-conditioned 3D generation mainly follows three paradigms. View-mediated methods lift 2D priors through multiview synthesis or score distillation (Liu et al., 2023; Shi et al., 2023; Liu et al., 2024; Poole et al., 2022; Lin et al., 2023; Wang et al., 2023b); feed-forward reconstruction models regress triplanes or related intermediates from a single image (Hong et al., 2024; Wang et al., 2023a; Tochilkin et al., 2024; Tang et al., 2024; Xu et al., 2024; Huang et al., 2025); and latent-native methods generate in learned 3D latent spaces (Zhao et al., 2023; Lan et al., 2024; Wu et al., 2024;

2025; Ren et al., 2024; Xiang et al., 2024). Recent asset-level systems extend the latent-native line toward joint geometry-and-material generation (Xiang et al., 2025; Zhang et al., 2024b; Team, 2025; Chen et al., 2025c; Li et al., 2025a;b; Chen et al., 2025b). Across these families, scalability usually comes from view space, triplane-style intermediates, or learned 3D latents rather than from direct geometry generation.

This design improves practicality but weakens geometric supervision: view-based methods optimize image consistency, reconstruction models are constrained by the chosen intermediate, and latent-native methods inherit tokenizer or compression error. GRAVER instead factorizes generation along a geometric hierarchy—block coordinates, occupancy support, and local UDF detail. Its latent stages are tied to explicit structural targets, while the final stage operates directly on geometric values.

2.2 Explicit geometry and sparse structure

Explicit 3D generation spans sparse voxel hierarchies, mesh-centric models, and local surface or field representations. Prior work shows that hierarchical sparsity helps scale geometry generation, while mesh or remeshed-surface methods improve explicitness and surface quality from different directions (Ren et al., 2024; Nash et al., 2020; Siddiqui et al., 2023; Chen et al., 2024b; Weng et al., 2025; Chen et al., 2024a; Lin et al., 2025; Shen et al., 2021; 2023; Lyu et al., 2024; Gu et al., 2024; Kerbl et al., 2023; Zhang et al., 2024a; Yang et al., 2025; Yariv et al., 2024; Yu et al., 2024). SparseFlex and Sparc3D are especially close in spirit because they emphasize high-resolution geometry and highlight the limitations of watertight conversion or latent bottlenecks (He et al., 2025; Li et al., 2025c).

Our position is narrower: sparsity is not only an efficiency tool but also the supervision decomposition itself. GRAVER first predicts sparse support and then generates local blockwise UDFs, placing the high-resolution generative state in explicit geometric variables rather than in latent voxels or mesh-token sequences.

2.3 Implicit fields, topology, and direct diffusion in 3D

Occupancy Networks and DeepSDF established continuous fields as an alternative to dense voxels, and later work such as NeuS and VolSDF improved geometry-aware reconstruction (Mescheder et al., 2018; Park et al., 2019; Wang et al., 2021; Yariv et al., 2021). However, occupancy and signed distance fields still favor watertight surfaces. UDF-based methods instead show that unsigned distance better accommodates open surfaces and arbitrary topology (Long et al., 2022; Guillard et al., 2021).

For generation, many diffusion methods still move to auxiliary domains such as point clouds, SDF-oriented spaces, or wavelet representations (Nichol et al., 2022; Cheng et al., 2023; Chou et al., 2022; Zhou et al., 2024). Motivated by recent direct-generation results in native data spaces (Hoogeboom et al., 2023; 2024; Chen et al., 2025a; Yu et al., 2025; Li & He, 2025; Lu et al., 2026), GRAVER keeps the representation geometry-first and predicts local UDF values directly after sparse structure has been determined.

3 Method

3.1 Overview

As illustrated in Figure 2, GRAVER is an image-conditioned 3D generator that separates global localization, surface support, and fine geometry. Stage 1 generates active blocks on a global G^3 grid. Stage 2 generates sparse support latents and decodes them into an S^3 support mask within each active block. Stage 3 directly generates fine unsigned distance values on a local D^3 lattice, conditioned on the decoded support. The final mesh is extracted using sparse marching cubes (Lorensen & Cline, 1987). The key idea is to model surface support in a compact latent space while retaining explicit distance values for fine geometric synthesis.

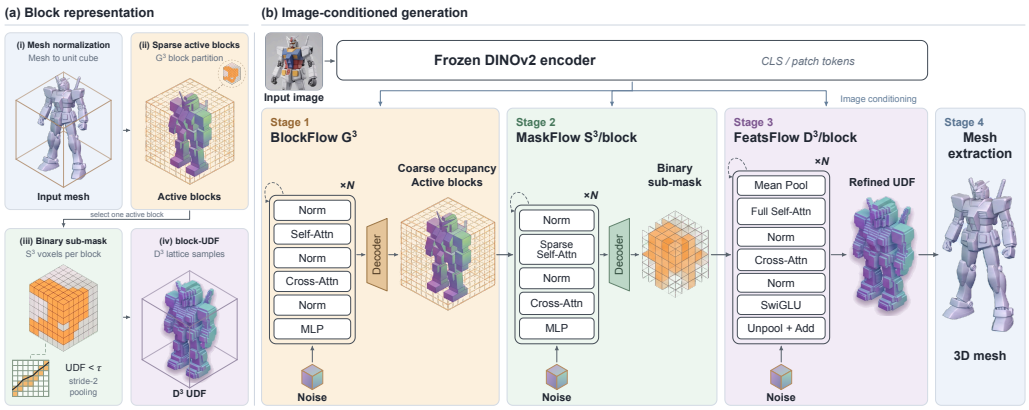

Figure 2: Overview of GRAVER and Block-UDF representation. **(a)** A mesh is decomposed into sparse blocks, each represented by a local D^3 UDF and an S^3 binary support mask. **(b)** Given an image, DINOv2 image tokens condition three stages that progressively predict the active-block layout, support masks, and fine UDF values. Sparse marching cubes then reconstructs the final 3D mesh.

3.2 Block-UDF Representation

We normalize each mesh into a canonical cube and partition the domain into G^3 blocks. Each active block $\mathbf{b}$ stores a local unsigned distance field (UDF) $\mathbf{u_b} \in [0,1]^{D^3}$. Adjacent blocks share boundary samples, giving an effective global resolution of $R = G(D-1)$. For a surface $\mathcal{M}$, normalized distances are defined as

$$u(\mathbf{x}) = \min\left(\frac{d(\mathbf{x},\mathcal{M})}{\tau}, 1\right), \qquad \tau = \frac{\kappa}{R}, \tag{1}$$

where κ controls the truncation width in grid intervals and 1 denotes the far field. We retain a sparse set of surface-relevant blocks $\mathcal{B}$ and summarize each local field with a binary support mask $\mathbf{m_b} \in \{0,1\}^{S^3}$:

$$m_{\mathbf{b},\mathbf{c}} = \mathbb{1}\left[\min_{\mathbf{j}\in\mathcal{V}_\mathbf{c}} u_{\mathbf{b},\mathbf{j}} < \tau_{\text{sup}}\right], \tag{2}$$

where $\mathcal{V}_\mathbf{c}$ contains the fine samples assigned to subcell $\mathbf{c}$. This representation provides three nested targets: block coordinates, surface support, and fine distances. We use $G = 64$, $S = 8$, and $D = 16$, yielding a 512^3 support grid and an effective UDF resolution of $R = 960$.

3.3 Progressive Geometry Generation

Stage 1: Global Block Layout. Following the sparse-structure VAE design of TRELLIS 2 (Xiang et al., 2025), Stage 1 models the global block layout in a compact latent space. An image-conditioned flow model generates a dense 16^3 latent with eight channels, which a frozen sparse-structure decoder maps to occupancy on the G^3 block grid. Thresholding the decoded occupancy yields the active block set $\mathcal{B}$ that indexes the subsequent stages. This stage establishes the coarse spatial structure, leaving surface support and fine distances to the following stages.

Stage 2: Latent Surface Support. Stage 2 models the distribution of local surface-support masks in a learned sparse latent space. A sparse occupancy VAE encodes the fine binary support into compact latent features. The encoder compresses the fine support through three spatial downsampling stages, producing a Gaussian posterior with a 32-dimensional latent at each retained coarse coordinate. The decoder reverses this hierarchy through learned subdivision: at each level, a parent feature predicts eight child-occupancy logits, and sparse upsampling instantiates the selected children. Three subdivision levels recover the support on the 512^3 grid.

The VAE is trained with

$$\mathcal{L}_{\text{VAE}} = \lambda_{\text{sub}}\mathcal{L}_{\text{sub}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}, \tag{3}$$

where $\mathcal{L}_{\text{sub}}$ supervises child occupancy at each level, $\mathcal{L}_{\text{rec}}$ supervises occupancy logits on the ground-truth support, and $\mathcal{L}_{\text{KL}}$ regularizes the posterior toward a standard Gaussian. Training uses ground-truth subdivision, whereas inference uses predicted subdivision.

After VAE training, we freeze the autoencoder and use channel-wise normalized posterior means as flow-matching targets. Let $\mathbf{z}_0$ denote these sparse latents. Following flow matching (Lipman et al., 2023), we construct

$$\mathbf{z}_s = (1-s)\mathbf{z}_0 + \big[\sigma_{\min} + (1-\sigma_{\min})s\big]\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{4}$$

where $s = 1$ corresponds to noise. A sparse transformer predicts the corresponding velocity:

$$\mathcal{L}_{\text{lat}} = \mathbb{E}_{s,\boldsymbol{\epsilon}}\left[\left\|\mathbf{v}_\theta(\mathbf{z}_s, s, \mathbf{c}; \mathcal{B}) - \big((1-\sigma_{\min})\boldsymbol{\epsilon} - \mathbf{z}_0\big)\right\|_2^2\right]. \tag{5}$$

Here $\mathbf{c}$ denotes DINOv2 image tokens (Oquab et al., 2024). The transformer uses sparse self-attention with rotary positional encoding and cross-attention to $\mathbf{c}$. At inference, sampled latents are denormalized and decoded into binary support masks $\hat{\mathbf{m}}_{\mathbf{b}}$.

Stage 3: Support-Conditioned Fine UDF. Stage 3 predicts explicit UDF values without a separate geometry decoder. Each block uses one token with a 64-dimensional bottleneck between its D^3 input and output fields. Pooled sparse attention in early layers aggregates coarse context, followed by windowed attention for local refinement.

The support serves two distinct purposes. A max-pooled 4^3 mask provides a compact conditioning signal, while the native 8^3 mask is upsampled to D^3 and dilated to define a binary gate $\mathbf{q}$. The gate restricts the stochastic field to a neighborhood of the support. Using $t = 0$ for noise and $t = 1$ for data, the corrupted field is

$$\mathbf{x}_t = \mathbf{q} \odot \big[t\mathbf{u} + (1-t)\boldsymbol{\epsilon}_u\big] + (1-\mathbf{q}) \odot \mathbf{1}, \qquad \boldsymbol{\epsilon}_u \sim \mathcal{N}(\mathbf{0}, \sigma_u^2\mathbf{I}). \tag{6}$$

Thus, unsupported samples remain at the deterministic far-field value throughout generation.

The network predicts the clean UDF $\hat{\mathbf{u}}$. We use a velocity-weighted squared residual,

$$e_{\mathbf{b},j} = q_{\mathbf{b},j}\left(\frac{\hat{u}_{\mathbf{b},j} - u_{\mathbf{b},j}}{\max(1-t, 0.05)}\right)^2, \tag{7}$$

and aggregate it over a narrow marching-cubes band, a wider near-surface band, and the remaining far field:

$$\mathcal{L}_{\text{udf}}^{(\mathbf{b})} = \lambda_{\text{mc}}\overline{e}_{\Omega_{\text{mc}}} + \lambda_{\text{near}}\overline{e}_{\Omega_{\text{near}}} + \lambda_{\text{far}}\overline{e}_{\Omega_{\text{far}}}. \tag{8}$$

Here $\overline{e}_\Omega$ denotes the mean residual over the corresponding distance band, with zero contribution from gated-out samples. The per-block objective is reweighted by a neighborhood-complexity factor and averaged per sample. An additional gradient-based regularizer supervises local field orientation.

Sampling and Mesh Extraction. Generation proceeds stage-wise. Stage 2 integrates the latent velocity field from $s = 1$ to $s = 0$ with Euler steps, followed by support decoding. Stage 3 integrates the clean-field prediction from $t = 0$ to $t = 1$:

$$\begin{aligned} \tilde{\mathbf{x}}_{t+\Delta t} &= \mathbf{x}_t + \Delta t\, \frac{\hat{\mathbf{u}}_\theta(\mathbf{x}_t, t, \mathbf{c}, \mathbf{m}) - \mathbf{x}_t}{\max(1-t, 0.05)}, \\ \mathbf{x}_{t+\Delta t} &= \mathbf{q} \odot \tilde{\mathbf{x}}_{t+\Delta t} + (1-\mathbf{q}) \odot \mathbf{1}. \end{aligned} \tag{9}$$

We use Heun correction for UDF sampling and take the clean prediction at the final step. Classifier-free guidance is applied to the image-conditioned predictions. Before meshing, predictions at shared block vertices are averaged. Sparse marching cubes extracts the positive UDF isosurface at $1/\kappa$, followed by mesh cleanup and normal-aware smoothing.

3.4 Native Local 3D Editing

We perform local editing through support recomposition followed by selective UDF replacement. The block layout $\mathcal{B}$ remains fixed, while a binary mask $\mathbf{r}$ selects editable support cells. Conditioned on the edited image, Stage 2 samples and decodes a support proposal $\hat{\mathbf{m}}$. We preserve the original support elsewhere:

$$\mathbf{m}' = \mathbf{r} \odot \hat{\mathbf{m}} + (1-\mathbf{r}) \odot \mathbf{m}. \tag{10}$$

Table 1: Quantitative comparison on image-conditioned 3D generation (higher is better). CLIP-N, ULIP-2 (Xue et al., 2024), and Uni3D (Zhou et al., 2023) evaluate semantic alignment and are reported as mean ± standard deviation; Quality-LLM is the mean LLM rating.

Method	CLIP-N ↑	ULIP-2 ↑	Uni3D ↑	Quality-LLM ↑
TRELLIS	0.6730 ± 0.0690	$\mathbf{0.3556 \pm 0.0966}$	0.3495 ± 0.0860	*6.73*
Hunyuan3D 2.1	0.6661 ± 0.0709	0.3333 ± 0.0982	0.3382 ± 0.0759	*7.30*
UltraShape	0.6716 ± 0.0682	0.3400 ± 0.1000	0.3414 ± 0.0797	*7.32*
TRELLIS 2	$\mathbf{0.6867 \pm 0.0633}$	0.3493 ± 0.0901	$\underline{0.3538 \pm 0.0743}$	**7.85**
GRAVER (Ours)	$\underline{0.6826 \pm 0.0607}$	$\underline{0.3497 \pm 0.0813}$	$\mathbf{0.3568 \pm 0.0668}$	*7.45*

The selected region may include previously unsupported cells, allowing support to be added as well as removed. Stage 3 then samples a gate-clamped UDF proposal $\tilde{\mathbf{u}}$ conditioned on the edited image and $\mathbf{m}'$. Both the compact mask conditioning and the fine-grid gate are recomputed from $\mathbf{m}'$. Let $\mathbf{r}^{\uparrow}$ denote the corresponding fine-grid edit footprint, including gate dilation. After averaging shared block vertices, we compose

$$\mathbf{u}' = \mathbf{r}^{\uparrow} \odot \tilde{\mathbf{u}} + (1 - \mathbf{r}^{\uparrow}) \odot \mathbf{u}. \tag{11}$$

The proposal is retained only inside the editable footprint; all other UDF samples are copied exactly from the original field. Preservation is thus a hard constraint rather than a consistency objective. Support insertion, deletion, and replacement enable the corresponding edits within $\mathcal{B}$. This field-level guarantee applies before mesh cleanup and smoothing.

4 Experiments

We study three goals: (i) comparison with representative image-conditioned 3D generators, (ii) validation of the Stage 3 sparse design, and (iii) evaluation of native editing, together with preliminary appearance reuse. We refer to the stages as Stage 1 (block layout), Stage 2 (support mask), and Stage 3 (fine UDF).

4.1 Experimental Setup

Unless otherwise stated, all methods are evaluated under a shared image-conditioned protocol, and generated shapes are converted to meshes before geometry evaluation. We report an end-to-end comparison with representative baselines, controlled ablations of the Stage 3 fine-UDF predictor, and native-editing experiments. Implementation and training details are provided in the supplementary material.

4.2 Baselines

Our external comparison set covers representative image-to-3D systems that generate geometry through latent or implicit intermediate spaces, including TRELLIS, TRELLIS 2, Hunyuan3D 2.1, and UltraShape. These methods are the most relevant reference points because they target the same single-image 3D generation problem but do not predict an explicit sparse UDF hierarchy. Table 1 reports the resulting quantitative comparison under a shared evaluation protocol.

Beyond external baselines, we compare against internal variants of GRAVER. These include simplified pipelines that remove the explicit Stage 2 support-mask transfer used to gate Stage 3 supervision, as well as alternative Stage 3 training parameterizations such as v-prediction versus x-prediction. These controlled comparisons help separate the benefits of the explicit sparse hierarchy from the benefits of the optimization objective.

4.3 Quantitative Comparison

Our full end-to-end benchmark evaluates image–shape semantic alignment and perceptual quality under a shared protocol.

Table 1 shows that TRELLIS 2 leads CLIP-N and Quality-LLM, TRELLIS leads ULIP-2, and GRAVER leads Uni3D.



Figure 3: Qualitative comparison against baseline image-to-3D methods.

GRAVER ranks second on the other three metrics, demonstrating balanced performance. For the three embedding metrics, the gaps among leading methods are small relative to the reported standard deviations, so the ranking varies with the evaluation encoder and across samples. This supports using multiple complementary metrics rather than a single embedding space.

Figure 3 provides the qualitative baseline comparison used in our experiment section. It focuses on failure modes that are especially informative for explicit sparse generation: thin structures, disconnected components, block-boundary continuity, and high-curvature surface regions. These cases are where the coarse-to-fine decomposition is most likely to matter, because Stage 1 determines global support, Stage 2 determines where local geometry is allowed to exist, and Stage 3 determines the final surface detail.

4.4 Ablation Study

All ablations in this subsection are performed on the Stage 3 fine-UDF predictor (the "fine-feats" module in the implementation), since this final prediction stage is where most of the surface-quality gains are decided. Unless otherwise stated, we report a single metric, $\mathrm{MSE}_{\mathrm{mc}}$, which we use as a training-surface-accuracy loss. Concretely, after extracting the predicted surface and the ground-truth surface in the marching-cubes band, we measure the mean squared distance from predicted surface samples to their nearest ground-truth surface samples; lower is better. Intuitively, a lower $\mathrm{MSE}_{\mathrm{mc}}$ means that the reconstructed surface lies closer to the target geometry in the region that most directly

Table 2: Stage 3 ablations ($\text{MSE}_{\text{mc}} \downarrow$).

(a) Prediction / Loss

	x-Prediction	v-Prediction	ϵ-Prediction
x-loss	4.3×10^{-4}	0.957	2.112
v-loss	$\mathbf{2.7 \times 10^{-4}}$	**0.859**	**2.074**
ϵ-loss	3.7×10^{-4}	1.129	2.198

(b) Time Shift

μ	x-Prediction	v-Prediction	ϵ-Prediction
0.0	14.0×10^{-4}	1.21	4.68
-0.4	9.7×10^{-4}	1.09	4.18
-0.8	$\mathbf{3.2 \times 10^{-4}}$	**0.96**	**3.72**
-1.2	5.3×10^{-4}	1.03	3.89

(c) Mask Prediction

Setting	MSE_{mc}
no mask	8.3×10^{-4}
mask $(D/2)^3$	$\mathbf{2.2 \times 10^{-4}}$
mask $(D/4)^3$	2.7×10^{-4}
mask $(D/8)^3$	3.9×10^{-4}

(d) Bottleneck Dimension

Dimension	MSE_{mc}
none	12.1×10^{-4}
256	2.9×10^{-4}
128	2.6×10^{-4}
64	$\mathbf{2.5 \times 10^{-4}}$

determines the final mesh quality. We group the study into four parts: prediction/loss parameterization, time shift, mask resolution, and bottleneck dimension, summarized in Table 2.

The prediction/loss ablation supports the final clean-state parameterization: x-prediction with the velocity-weighted loss achieves the lowest error. The time-shift sweep consistently favors a moderately negative shift, with $\mu = -0.8$ performing best for all three prediction targets. The masking ablation is particularly important: transferring the Stage 2 support mask sharply reduces near-surface error, and the native $(D/2)^3 = 8^3$ setting gives the best result among the tested resolutions.

The bottleneck study suggests that lightweight compression is not only a memory-saving device but also an effective regularizer for the transformer backbone. All tested bottlenecks improve over the uncompressed input, with dimension 64 giving the lowest error. Together with the external comparison, these ablations connect the end-to-end results to the support transfer, objective, time shift, and bottleneck used by the final model.

4.5 Native Editing

Beyond generation, GRAVER's explicit block-UDF representation supports 3D inpainting and local editing by regenerating selected spatial regions while retaining support masks and UDF values elsewhere.

Local editing. We partition the sparse field into editable and fixed regions, regenerate only the editable region, and preserve the original field elsewhere. Figure 5A repairs defects around the throat and clothing by regenerating their support and fine UDF values. Figures 5B–C show larger add, remove, and replace operations. In each case, editing is performed in the native spatial representation, so satisfactory regions can be retained rather than regenerated with the entire asset.

Local preservation. Local editing is useful only if the unedited region remains stable. We therefore fix vertical regions covering 0%, 20%, 50%, or 100% of the asset height and regenerate the remainder. Table 3 reports differences over both the full asset and the fixed region for model-generated and artist-authored inputs. Preserving both the support mask and fine UDF values improves most fixed-region geometry and rendered-normal metrics over preserving the mask alone; the main exception is surface drift on artist-authored assets, where remeshing and smoothing can move the extracted surface despite field-level preservation. Larger fixed regions also generally increase full-asset similarity. Figure 4 visualizes the corresponding rendered differences.

Small-scale PBR texturing We also adapt Stage 3 to predict blockwise PBR attributes while keeping the Stage 1 layout and Stage 2 mask fixed. The supplementary material describes this preliminary appearance-prediction experiment; it is not part of the geometry benchmark above.



Figure 4: LPIPS differences in regeneration under different fixed-region ratios.

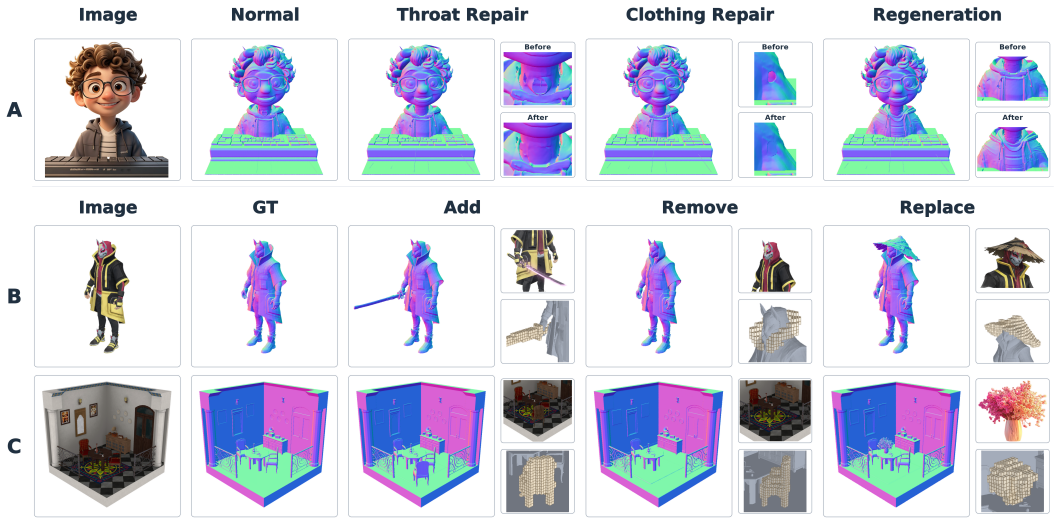

Figure 5: Native edits on 3D assets while retaining unedited geometry.

Table 3: Quantitative fixed-region preservation during native editing.

Setting	Method	Model-generated						Artist-authored					
		Geometry			Normal images			Geometry			Normal images		
		NC_{abs} (%) ↑	F-score (%) ↑	Drift (h) ↓	LPIPS ↓	SSIM ↑	CLIP-N ↑	NC_{abs} (%) ↑	F-score (%) ↑	Drift (h) ↓	LPIPS ↓	SSIM ↑	CLIP-N ↑
(a) All: full region													
Fix@0	Mask only	87.991	59.294	1.4443	0.1750	0.5907	0.9774	91.026	46.257	1.5795	0.2004	0.5509	0.9701
	Mask + Feats	87.996	59.315	1.4438	0.1751	0.5907	0.9778	91.027	46.284	1.5791	0.2006	0.5509	0.9703
Fix@20	Mask only	89.269	65.251	1.2462	0.1596	0.6368	0.9793	92.755	57.008	1.2508	0.1761	0.6028	0.9726
	Mask + Feats	89.586	66.108	1.2013	0.1578	0.6470	0.9797	92.960	65.026	1.2822	0.1740	0.6127	0.9729
Fix@50	Mask only	93.086	79.327	0.7762	0.1174	0.7275	0.9833	94.640	63.068	1.0975	0.1340	0.6742	0.9768
	Mask + Feats	94.163	81.681	0.6545	0.1093	0.7559	0.9837	95.137	75.852	1.1460	0.1269	0.6983	0.9774
Fix@100	Mask only	97.772	96.537	0.2111	0.0260	0.8976	0.9967	97.366	70.154	**0.9061**	0.0706	0.7856	0.9919
	Mask + Feats	**99.525**	**99.998**	**0.0172**	**0.0057**	**0.9544**	**0.9994**	**98.524**	**88.319**	0.9835	**0.0524**	**0.8379**	**0.9940**
(b) Fix: fixed region													
Fix@20	Mask only	97.427	96.289	0.2219	0.0244	0.8572	0.9868	97.887	68.832	0.9099	0.0458	0.7844	0.9679
	Mask + Feats	99.319	99.951	0.0216	0.0124	0.9238	0.9919	**98.859**	**90.723**	0.9775	**0.0373**	0.8283	0.9772
Fix@50	Mask only	97.709	96.285	0.2208	0.0271	0.8831	0.9907	97.712	69.673	0.9094	0.0606	0.7933	0.9836
	Mask + Feats	99.492	99.985	0.0173	0.0091	0.9447	0.9958	98.660	89.783	0.9801	0.0487	**0.8396**	0.9880
Fix@100	Mask only	97.772	96.537	0.2111	0.0260	0.8976	0.9967	97.366	70.154	**0.9061**	0.0706	0.7856	0.9919
	Mask + Feats	**99.525**	**99.998**	**0.0172**	**0.0057**	**0.9544**	**0.9994**	98.524	88.319	0.9835	0.0524	0.8379	**0.9940**

NC_{abs}: normal consistency; F-score ($\tau = h = 1/960$); Drift: surface distance/h. Fix@r: r% height fixed.

5 Conclusion

We presented GRAVER, an image-conditioned 3D generation framework that combines latent modeling of coarse structure with direct prediction of fine UDF values in an explicit sparse space. Its block-UDF hierarchy separates active-block localization, local surface support, and fine geometry, concentrating high-resolution prediction near the surface while accommodating open and non-watertight meshes. Experiments show competitive semantic alignment, validate the support-conditioned Stage 3 design, and demonstrate that native field composition can preserve fixed regions during local editing. The framework still relies on a hand-designed hierarchy and remains computationally demanding at high resolution. Moreover, the current evaluation focuses primarily on geometry; the PBR experiment is preliminary rather than a complete appearance-generation pipeline. Future work should improve efficiency, extend the representation to larger scenes, and develop joint geometry–material generation and broader editing evaluations.

AI USE STATEMENT

Generative AI tools were used only to assist with writing and language polishing, including improvements to grammar, clarity, and presentation. They were not used to develop the core research ideas or methods, conduct experiments, analyze results, or generate scientific claims. All AI-assisted text was reviewed and revised by the authors, who take full responsibility for the final content of this work.

REFERENCES

Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. Pixelflow: Pixel-space generative models with flow. *ArXiv*, abs/2504.07963, 2025a. URL https://api.semanticscholar.org/CorpusID:277667747.

Sijin Chen, Xin Chen, Anqi Pang, Xianfang Zeng, Wei Cheng, Yijun Fu, Fukun Yin, Yanru Wang, Zhibin Wang, Chi Zhang, Jingyi Yu, Gang Yu, Bin Fu, and Tao Chen. Meshxl: Neural coordinate field for generative 3d foundation models, 2024a. URL https://arxiv.org/abs/2405.20853.

Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, Guosheng Lin, and Chi Zhang. Meshanything: Artist-created mesh generation with autoregressive transformers, 2024b. URL https://arxiv.org/abs/2406.10163.

Yiwen Chen, Zhihao Li, Yikai Wang, Hu Zhang, Qin Li, Chi Zhang, and Guosheng Lin. Ultra3d: Efficient and high-fidelity 3d generation with part attention, 2025b. URL https://arxiv.org/abs/2507.17745.

Zhaoxi Chen, Jiaxiang Tang, Yuhao Dong, Ziang Cao, Fangzhou Hong, Yushi Lan, Tengfei Wang, Haozhe Xie, Tong Wu, Shunsuke Saito, Liang Pan, Dahua Lin, and Ziwei Liu. 3dtopia-xl: Scaling high-quality 3d asset generation via primitive diffusion, 2025c. URL https://arxiv.org/abs/2409.12957.

Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander Schwing, and Liangyan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation, 2023. URL https://arxiv.org/abs/2212.04493.

Gene Chou, Yuval Bahat, and Felix Heide. Diffusion-sdf: Conditional generative modeling of signed distance functions. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2262–2272, 2022. URL https://api.semanticscholar.org/CorpusID:254017862.

Chun Gu, Zeyu Yang, Zijie Pan, Xiatian Zhu, and Li Zhang. Tetrahedron splatting for 3d generation. In *Neural Information Processing Systems*, 2024. doi: 10.52202/079017-2548. URL https://mlanthology.org/neurips/2024/gu2024neurips-tetrahedron/.

Benoît Guillard, Federico Stella, and P. Fua. Meshudf: Fast and differentiable meshing of unsigned distance field networks. In *European Conference on Computer Vision*, 2021. URL https://api.semanticscholar.org/CorpusID:244714325.

Xianglong He, Zi-Xin Zou, Chia-Hao Chen, Yuanchen Guo, Ding Liang, Chun Yuan, Wanli Ouyang, Yan-Pei Cao, and Yangguang Li. Sparseflex: High-resolution and arbitrary-topology 3d shape modeling. *ArXiv*, abs/2503.21732, 2025.

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 50678–50702, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/dcad3425f5c8c36b5b3885c091bf1257-Paper-Conference.pdf.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. *ArXiv*, abs/2301.11093, 2023. URL https://api.semanticscholar.org/CorpusID:256274516.

Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion (sid2): 1.5 fid on imagenet512 with pixel-space diffusion. *ArXiv*, abs/2410.19324, 2024. URL https://api.semanticscholar.org/CorpusID:273638639.

Zixuan Huang, Mark Boss, Aaryaman Vasishta, James M. Rehg, and Varun Jampani. Spar3d: Stable point-aware reconstruction of 3d objects from single images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16860–16870, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Huang_SPAR3D_Stable_Point-Aware_Reconstruction_of_3D_Objects_from_Single_Images_CVPR_2025_paper.html.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. doi: 10.1145/3592433. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

Yushi Lan, Fangzhou Hong, Shuai Yang, Shangchen Zhou, Xuyi Meng, Bo Dai, Xingang Pan, and Chen Change Loy. Ln3diff: Scalable latent neural fields diffusion for speedy 3d generation. In *European Conference on Computer Vision*, 2024. URL https://arxiv.org/abs/2403.12019.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025. URL https://arxiv.org/abs/2511.13720.

Weiyu Li, Jiarui Liu, Hongyu Yan, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long. Craftsman3d: High-fidelity mesh generation with 3d native generation and interactive geometry refiner, 2025a. URL https://arxiv.org/abs/2405.14979.

Weiyu Li, Xuanyang Zhang, Zheng Sun, Di Qi, Hao Li, Wei Cheng, Weiwei Cai, Shihao Wu, Jiarui Liu, Zihao Wang, Xiao Chen, Feipeng Tian, Jianxiong Pan, Zeming Li, Gang Yu, Xiangyu Zhang, Daxin Jiang, and Ping Tan. Step1x-3d: Towards high-fidelity and controllable generation of textured 3d assets, 2025b. URL https://arxiv.org/abs/2505.07747.

Zhihao Li, Yufei Wang, Heliang Zheng, Yihao Luo, and Bihan Wen. Sparc3d: Sparse representation and construction for high-resolution 3d shapes modeling. *ArXiv*, abs/2505.14521, 2025c.

Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation, 2023. URL https://arxiv.org/abs/2211.10440.

Yuchen Lin, Chenguo Lin, Panwang Pan, Honglei Yan, Yiqiang Feng, Yadong Mu, and Katerina Fragkiadaki. Partcrafter: Structured 3d mesh generation via compositional latent diffusion transformers. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=0e1KaHtgvU.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9298–9309, 2023.

Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 27676–27697, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/753d9584b57ba01a10482f1ea7734a89-Paper-Conference.pdf.

Xiaoxiao Long, Chu-Hsing Lin, Lingjie Liu, Yuan Liu, Peng Wang, Christian Theobalt, Taku Komura, and Wenping Wang. Neuraludf: Learning unsigned distance fields for multi-view reconstruction of surfaces with arbitrary topologies. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20834–20843, 2022. URL https://api.semanticscholar.org/CorpusID:254018110.

William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 163–169, 1987. doi: 10.1145/37402.37422. URL https://dl.acm.org/doi/10.1145/37402.37422.

Yiyang Lu, Susie Lu, Qiao Sun, Hanhong Zhao, Zhicheng Jiang, Xianbang Wang, Tianhong Li, Zheng Geng, and Kaiming He. One-step latent-free image generation with pixel mean flows. *ArXiv*, abs/2601.22158, 2026. URL https://api.semanticscholar.org/CorpusID:285139904.

Zhaoyang Lyu, Ben Fei, Jinyi Wang, Xudong Xu, Ya Zhang, Weidong Yang, and Bo Dai. Getmesh: A controllable model for high-quality mesh generation and manipulation, 2024. URL https://arxiv.org/abs/2403.11990.

Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4455–4465, 2018.

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. Polygen: An autoregressive generative model of 3d meshes, 2020. URL https://arxiv.org/abs/2002.10880.

Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts, 2022. URL https://arxiv.org/abs/2212.08751.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. URL https://arxiv.org/abs/2304.07193.

Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 165–174, 2019. URL https://api.semanticscholar.org/CorpusID:58007025.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022. URL https://arxiv.org/abs/2209.14988.

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

Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: A hybrid representation for high-resolution 3d shape synthesis. In *Neural Information Processing Systems*, 2021. URL https://mlanthology.org/neurips/2021/shen2021neurips-deep/.

Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization, 2023. URL https://arxiv.org/abs/2308.05371.

Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv:2308.16512*, 2023.

Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers, 2023. URL https://arxiv.org/abs/2311.15475.

Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation, 2024. URL https://arxiv.org/abs/2402.05054.

Tencent Hunyuan3D Team. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material, 2025.

Dmitry Tochilkin, David Pankratz, Zexiang Liu, Zixuan Huang, Adam Letts, Yangguang Li, Ding Liang, Christian Laforte, Varun Jampani, and Yan-Pei Cao. Triposr: Fast 3d object reconstruction from a single image, 2024. URL https://arxiv.org/abs/2403.02151.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 27171–27183. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/e41e164f7485ec4a28741a2d0ea41c74-Paper.pdf.

Peng Wang, Hao Tan, Sai Bi, Yinghao Xu, Fujun Luan, Kalyan Sunkavalli, Wenping Wang, Zexiang Xu, and Kai Zhang. Pf-lrm: Pose-free large reconstruction model for joint pose and shape prediction, 2023a. URL https://arxiv.org/abs/2311.12024.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation, 2023b. URL https://arxiv.org/abs/2305.16213.

Haohan Weng, Yikai Wang, Tong Zhang, C. L. Philip Chen, and Jun Zhu. Pivotmesh: Generic 3d mesh generation via pivot vertices guidance. In *International Conference on Learning Representations*, 2025. URL https://mlanthology.org/iclr/2025/weng2025iclr-pivotmesh/.

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Jingxi Xu, Philip Torr, Xun Cao, and Yao Yao. Direct3d: Scalable image-to-3d generation via 3d latent diffusion transformer. *Advances in Neural Information Processing Systems*, 37:121859–121881, 2024.

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

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21469–21480, 2024.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hong Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3d generation. *ArXiv*, abs/2512.14692, 2025.

Yinghao Xu, Zifan Shi, Yifan Wang, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation, 2024. URL https://arxiv.org/abs/2403.14621.

Le Xue, Ning Yu, Shu Zhang, Artemis Panagopoulou, Junnan Li, Roberto Mart'ın-Mart'ın, 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*, pp. 27091–27101, 2024.

Dingdong Yang, Yizhi Wang, Konrad Schindler, Ali Mahdavi Amiri, and Hao Zhang. Gala: Geometry-aware local adaptive grids for detailed 3d generation. In *International Conference on Learning Representations*, 2025. URL https://mlanthology.org/iclr/2025/yang2025iclr-gala/.

Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:235605960.

Lior Yariv, Omri Puny, Natalia Neverova, Oran Gafni, and Yaron Lipman. Mosaic-sdf for 3d generative models, 2024. URL https://arxiv.org/abs/2312.09222.

Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqi Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation. *ArXiv*, abs/2511.20645, 2025. URL https://api.semanticscholar.org/CorpusID:283251385.

Zhengming Yu, Zhiyang Dou, Xiaoxiao Long, Cheng Lin, Zekun Li, Yuan Liu, Norman Müller, Taku Komura, Marc Habermann, Christian Theobalt, Xin Li, and Wenping Wang. Surf-d: Generating high-quality surfaces of arbitrary topologies using diffusion models, 2024. URL https://arxiv.org/abs/2311.17050.

Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Trans. Graph.*, 42(4), jul 2023. ISSN 0730-0301. doi: 10.1145/3592442. URL https://doi.org/10.1145/3592442.

Bowen Zhang, Yiji Cheng, Jiaolong Yang, Chunyu Wang, Feng Zhao, Yansong Tang, Dong Chen, and Baining Guo. Gaussiancube: A structured and explicit radiance representation for 3d generative modeling, 2024a. URL https://arxiv.org/abs/2403.19655.

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

Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, BIN FU, Tao Chen, Gang YU, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xmxgMij3LY.

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.

Junsheng Zhou, Weiqi Zhang, Baorui Ma, Kanle Shi, Yu-Shen Liu, and Zhizhong Han. Udiff: Generating conditional unsigned distance fields with optimal wavelet diffusion. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21496–21506, 2024. URL https://api.semanticscholar.org/CorpusID:269033187.

SUPPLEMENTARY MATERIAL

A NETWORK ARCHITECTURES

Table 4. Architecture summary of the three generative stages used in GRAVER.

Stage 1: Block-Layout Latent Flow	Stage 2: Mask-Latent Flow	Stage 3: Fine-UDF Flow
Representation. Dense 8-channel sparse-structure VAE latents on a 16^3 grid **VAE.** Pretrained TRELLIS 2 sparse-structure (SS) VAE encoder/decoder; the decoder maps generated latents to 64^3 sparse block coordinates **Input.** Noisy structure latents, 8 channels **Condition.** DINOv2 ViT-L/14-reg image features, 1024 channels **Backbone.** TrellisSparseStructureFlowModel 30 transformer blocks, hidden dim. 1536, 12 heads **Attention.** RoPE; image cross-attention; QK RMSNorm for self- and cross-attention **MLP.** Ratio $= 5.3334$; shared modulation **Prediction.** Latent velocity on the 16^3 grid **Output.** VAE-decoded occupancy on the 64^3 block grid	**Representation.** 32-channel sparse occupancy-VAE latents on a 64^3 grid **VAE.** SparseTrellis2OccupancyVAE; channels $[64, 128, 256, 512]$, latent dim. 32 **VAE blocks.** Sparse ConvNeXt blocks with sparse down/up ResBlocks **Input.** Noisy sparse occupancy latents and their coordinates **Condition.** DINOv2 ViT-L/14 with registers, 1024 channels **Backbone.** ElasticSparseOccLatentTrellis2FlowModel 30 blocks, hidden dim. 1024, 16 heads **Attention.** RoPE; QK RMSNorm for self- and cross-attention; shared modulation **MLP.** Ratio $= 5.3334$ **Prediction.** Sparse latent velocity, 32 channels **Output.** Three-level VAE decoding produces the $S{=}8$ support mask (512^3 globally)	**Representation.** Explicit UDF values on a local $D^3{=}16^3$ lattice **Input.** Gate-clamped noisy UDF values on the active sparse block set **Support.** Native 8^3 mask for gating; max-pooled 4^3 mask for conditioning **Block input.** Bottleneck dim. 64 **Condition.** DINOv2 ViT-L/14 with 32 learned context registers, 1024 channels **Backbone.** SparseFlowModel; 8 pooled layers followed by local sparse layers (20 total) **Attention.** Windowed sparse attention, window size 32; RoPE; QK RMSNorm for self- and cross-attention **Hidden dim.** 1024, 16 heads; MLP ratio $= 4$ **Prediction.** Clean UDF values, 16^3 per active block **Output.** Explicit UDF field without a geometry decoder

The first two stages use VAEs to define compact latent targets for flow matching and freeze them before training their flow models. In Stage 1, the TRELLIS 2 sparse-structure VAE encodes the block layout into a dense 8-channel latent on a 16^3 grid. The image-conditioned flow model predicts its latent velocity, and the frozen decoder produces occupancy on the 64^3 block grid. In Stage 2, a separate sparse occupancy VAE encodes the $S{=}8$ support masks into 32-channel sparse latents. The mask-latent flow model generates these latents on the active block set, and the three-level decoder reconstructs the global 512^3 support. Stage 3 does not use a geometry VAE: it directly predicts clean 16^3 UDF values for each active block.

Stage 2's latent flow model uses a sparse, elastic transformer. Its input consists of sparse latent features and their associated coordinates; RoPE supplies positional information within the transformer. Stage 3 uses sparse pooling for global image-conditioned reasoning and windowed sparse attention for local UDF refinement. Its native 8^3 support mask is upsampled and dilated to gate the stochastic field, while a max-pooled 4^3 version conditions the transformer.

B DATA PREPARATION DETAILS

Our dataset is constructed by combining several large-scale 3D repositories, including TexVerse, ObjaverseXL (from both Sketchfab and GitHub), ABO, and HSSD. As shown in Table 5, the resulting training set comprises a diverse collection of 3D shapes with substantial variability in geometry and category coverage.

Overall, the training set contains 679,174 shapes. For evaluation, we use the Toys4k dataset, which includes 3,229 high-quality 3D assets. This split ensures no overlap between training and evaluation data, allowing for a reliable assessment of the model's generalization ability. Following the large-scale 3D asset preparation protocols in recent 3D generation works, we build a fully automated pipeline to convert raw 3D assets into paired image conditions and sparse geometry supervision. The pipeline consists of four stages: metadata construction, conditional view rendering, image-feature encoding, and sparse block-UDF extraction.

B.1 ASSET INDEXING AND METADATA CONSTRUCTION

We begin from a collection of raw 3D assets in common formats, including `.glb`, `.gltf`, `.fbx`, `.blend`, and `.obj`. For each asset, we compute a SHA256 hash over the original file content and

Table 5: Dataset composition.

Source	Shape Available
TexVerse	268135
ObjaverseXL (Sketchfab)	168307
ObjaverseXL (GitHub)	231577
ABO	4485
HSSD	6670
Training set total	**679174**
Toys4k (evaluation)	3229

use it as the unique instance identifier. The metadata table stores this identifier together with the processing status fields needed by later stages.

This hash-based indexing provides a stable mapping between the processed supervision and the original assets, independent of naming conventions. During incremental preprocessing, existing entries are preserved and newly discovered assets are appended. When preprocessing is distributed across multiple jobs, per-job metadata shards are merged by grouping entries with the same SHA256 identifier and keeping the latest valid processing status, yielding a single canonical metadata table.

B.2 Conditional Rendering

For each indexed asset, we render a set of image conditions using Blender. The rendering script imports the original asset, removes hidden objects, converts all visible geometry to meshes, triangulates the geometry, and normalizes the scene into a unit cube centered at the origin. The resulting normalized geometry is then passed to the later geometry-encoding stage.

By default, we render $V = 8$ RGBA views per asset at 1024×1024 resolution. To make the image-conditioned model robust to natural object photographs, camera poses are sampled with a bias toward product-like front-facing views while retaining viewpoint diversity. Specifically, for each asset we deterministically seed the camera sampler using the SHA256 hash. For 70% of the views, the yaw angle is sampled from a truncated Gaussian centered at the front direction, clipped to $\pm 60^\circ$, and the pitch angle is sampled around 37.5° and clipped to $[20^\circ, 50^\circ]$. The remaining views are sampled on the upper hemisphere using a low-discrepancy Hammersley sequence. The camera field of view is randomly sampled in the range $[30^\circ, 50^\circ]$ by sampling the corresponding camera radius.

For each asset, we keep the rendered RGBA views together with the corresponding camera intrinsics/extrinsics and the normalized mesh geometry. An instance is marked as successfully rendered only if the required number of views and camera metadata are both present. This strict validation avoids stale entries when some renders are missing or incomplete, and failed or timed-out Blender jobs can be resumed safely.

B.3 Image Condition Feature Encoding

After rendering, we optionally pre-compute image condition features for all assets with successful conditional renders. For each rendered RGBA image, we follow the same preprocessing used during online training: we crop the object using the non-empty alpha bounding box, enlarge the crop by a factor of 1.2, resize it to 518×518, and premultiply the RGB channels by the alpha mask. The resulting image is normalized with ImageNet statistics and passed through a pretrained DINOv2 ViT-L/14-reg encoder.

For each view, we store the normalized patch-token feature

$$\mathbf{c}_v \in \mathbb{R}^{1369 \times 1024},$$

corresponding to the 37×37 patch grid produced from a 518×518 input image. The metadata table is updated with a boolean flag indicating whether DINOv2 condition features have been successfully encoded. These offline features reduce data-loading overhead during training and ensure that training and evaluation use identical image preprocessing.

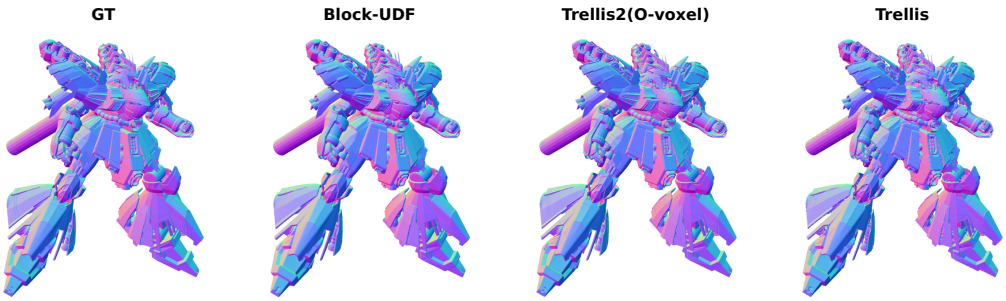

Figure 6: Representation comparison among ground-truth meshes, our Block-UDF, TRELLIS.2 occupancy voxels, and TRELLIS outputs.

B.4 Sparse Block-UDF Geometry Encoding

To obtain geometry supervision, we convert each normalized `mesh.ply` into a sparse block-based unsigned distance representation. The representation is defined on a coarse 64^3 grid. We first detect active blocks by querying the unsigned distance from each block center to the mesh surface. A block is considered active if its center distance is below a conservative threshold proportional to the block diagonal:

$$d(\mathbf{x}) \leq 1.1 \cdot \frac{\sqrt{3}}{2N}, \quad N = 64.$$

This identifies blocks that potentially intersect or lie near the object surface. For each active block, we sample a local unsigned distance field on a dense 16^3 lattice. Adjacent blocks share boundary samples by using 15 inner intervals per block, giving an effective global sampling resolution of

$$R = 64 \times 15 = 960.$$

Let $\Delta = 1/R$ be the global voxel size. We truncate the unsigned distance to five global voxels and normalize it to $[0, 1]$:

$$u(\mathbf{x}) = \text{clip}\left(\frac{d(\mathbf{x})}{5\Delta}, 0, 1\right).$$

Only surface blocks whose minimum normalized UDF value is below the surface threshold are retained. In our implementation, the marching-cubes isovalue is set to 0.2, corresponding to one global voxel in normalized UDF space, and the surface retention threshold is set to 0.4.

For each retained block i, we store its integer coordinate

$$\mathbf{p}_i \in \{0, \ldots, 63\}^3$$

and its local UDF feature

$$\mathbf{u}_i \in [0, 1]^{16^3}.$$

In addition, we derive a per-block binary support mask. The 16^3 UDF samples are partitioned into 8^3 sub-cells, and a sub-cell is marked active if any contained UDF sample lies below the surface threshold. This gives

$$\mathbf{m}_i \in \{0, 1\}^{8^3},$$

which is decoded by Stage 2 and subsequently used by Stage 3 for both conditioning and fine-grid gating.

The final block representation of an asset is

$$\mathcal{B} = \{(\mathbf{p}_i, \mathbf{u}_i, \mathbf{m}_i)\}_{i=1}^{L},$$

where L is the number of retained surface blocks. Coordinates are stored as 32-bit integers, local UDF features are stored in float16 to reduce disk usage, and support masks are stored as float32 binary arrays. The metadata table is updated with the number of retained blocks and a success/failure status for each asset.

For efficiency, UDF queries are accelerated with a GPU BVH when available. If the input mesh contains more than one million faces, we simplify it for BVH construction to avoid excessive memory usage and stack overflows during distance queries. The processing script supports rank/world-size sharding and can therefore be launched as independent jobs over large datasets.

B.5 Quality Control and Resumability

Each stage is designed to be resumable. Rendering is skipped only when all expected images and camera metadata are present; feature encoding is skipped when all view features already exist; and geometry encoding is skipped when the compressed block file contains all required arrays. Optionally, assets can be filtered by aesthetic score when such annotations are available. This filtering removes visually poor or overly simplistic assets and improves the geometric and visual diversity of the final training set.

Overall, the resulting dataset provides, for each valid 3D asset, a stable identifier, the original asset path, multi-view image prompts, DINOv2 image features, a normalized mesh, and a sparse high-resolution block-UDF representation. This format follows the spirit of recent sparse structured 3D latent pipelines while adapting the preprocessing to our block-based geometry representation.

C Training Details

We use DINOv2 image tokens for all stages, but the generated variables and prediction parameterizations differ. Stages 1 and 2 generate VAE latents with velocity prediction, whereas Stage 3 predicts clean UDF values directly.

Stage 1: block-layout latent flow. The pretrained sparse-structure VAE is frozen. Let $\mathbf{a}_0 \in \mathbb{R}^{16^3 \times 8}$ denote its channel-wise normalized posterior mean for the block layout. We sample $s \sim \mathcal{U}(0,1)$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and construct

$$\mathbf{a}_s = (1-s)\mathbf{a}_0 + \big[\sigma_{\min} + (1-\sigma_{\min})s\big]\boldsymbol{\epsilon}. \tag{12}$$

The structure model predicts the latent velocity $(1-\sigma_{\min})\boldsymbol{\epsilon} - \mathbf{a}_0$:

$$\mathcal{L}_{\text{stage1}} = \mathbb{E}_{s,\boldsymbol{\epsilon}}\left[\left\|\mathbf{v}_\theta^{\text{blk}}(\mathbf{a}_s, s, \mathbf{c}) - \big((1-\sigma_{\min})\boldsymbol{\epsilon} - \mathbf{a}_0\big)\right\|_2^2\right]. \tag{13}$$

At inference, we integrate from $s=1$ to $s=0$, denormalize the sampled latent, and decode it with the frozen VAE. Thresholding the decoded 64^3 occupancy gives the active block set $\mathcal{B}$.

Stage 2: mask-latent flow. Stage 2 first trains a sparse occupancy VAE for the native $S^3 = 8^3$ support masks. The encoder uses three spatial downsampling stages and produces a 32-dimensional Gaussian latent at each retained coarse coordinate. Its objective is

$$\mathcal{L}_{\text{VAE}} = \lambda_{\text{sub}}\mathcal{L}_{\text{sub}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}, \tag{14}$$

where $\mathcal{L}_{\text{sub}}$ supervises the eight child occupancies at each subdivision level, $\mathcal{L}_{\text{rec}}$ supervises the final support logits, and $\mathcal{L}_{\text{KL}}$ regularizes the posterior. Ground-truth subdivision is used for VAE training; predicted subdivision is used at inference.

After training, the occupancy VAE is frozen. Let $\mathbf{z}_0$ be the channel-wise normalized posterior mean on the active block set. Using the same noise convention as Stage 1,

$$\mathbf{z}_s = (1-s)\mathbf{z}_0 + \big[\sigma_{\min} + (1-\sigma_{\min})s\big]\boldsymbol{\epsilon}, \tag{15}$$

and the sparse transformer is trained with

$$\mathcal{L}_{\text{stage2}} = \mathbb{E}_{s,\boldsymbol{\epsilon}}\left[\left\|\mathbf{v}_\theta^{\text{mask}}(\mathbf{z}_s, s, \mathbf{c}; \mathcal{B}) - \big((1-\sigma_{\min})\boldsymbol{\epsilon} - \mathbf{z}_0\big)\right\|_2^2\right]. \tag{16}$$

Euler integration from $s=1$ to $s=0$ yields a sparse latent sample, which is denormalized and hierarchically decoded into the 8^3 support mask for each active block.

Stage 3: support-conditioned clean-UDF prediction. For each active block, Stage 3 directly predicts $\mathbf{u}_\mathbf{b} \in [0,1]^{D^3}$ with $D=16$. The decoded 8^3 support mask has two roles. A 2^3 max-pooling operation gives the compact 4^3 conditioning mask

$$\mathbf{m}_\mathbf{b}^{\text{cond}} = \text{MaxPool}_{2^3}(\mathbf{m}_\mathbf{b}), \tag{17}$$

while nearest-neighbor upsampling followed by dilation gives the fine-grid gate

$$\mathbf{q_b} = \mathrm{Dilate}\left(\mathrm{Up}_{16^3}(\mathbf{m_b})\right) \in \{0,1\}^{16^3}. \tag{18}$$

The dilation retains a small neighborhood around the decoded support, covering thin structures, high-curvature regions, and shared block boundaries.

Using $t = 0$ for noise and $t = 1$ for data, we construct

$$\mathbf{x}_{t,\mathbf{b}} = \mathbf{q_b} \odot [t\mathbf{u_b} + (1-t)\boldsymbol{\epsilon}_{u,\mathbf{b}}] + (1-\mathbf{q_b}) \odot \mathbf{1}, \qquad \boldsymbol{\epsilon}_{u,\mathbf{b}} \sim \mathcal{N}(\mathbf{0}, \sigma_u^2 \mathbf{I}). \tag{19}$$

Thus, unsupported samples remain fixed at the normalized far-field value 1. The network predicts the clean UDF $\hat{\mathbf{u}}_\mathbf{b}$ rather than velocity. Its voxelwise residual is

$$e_{\mathbf{b},j} = q_{\mathbf{b},j} \left(\frac{\hat{u}_{\mathbf{b},j} - u_{\mathbf{b},j}}{\max(1-t, 0.05)} \right)^2. \tag{20}$$

We use the final timestep-shift setting $\mu = -0.8$. We aggregate this residual over the marching-cubes, near-surface, and far-field bands:

$$\mathcal{L}_{\mathrm{udf}}^{(\mathbf{b})} = \lambda_{\mathrm{mc}} \overline{e}_{\Omega_{\mathrm{mc}}} + \lambda_{\mathrm{near}} \overline{e}_{\Omega_{\mathrm{near}}} + \lambda_{\mathrm{far}} \overline{e}_{\Omega_{\mathrm{far}}}, \tag{21}$$

where gated-out samples contribute zero. The final objective is

$$\mathcal{L}_{\mathrm{stage3}} = \lambda_{\mathrm{flow}}\, \mathbb{E}_{\mathbf{b}}[w_{\mathbf{b}}^{\mathrm{comp}} \mathcal{L}_{\mathrm{udf}}^{(\mathbf{b})}] + \lambda_{\mathrm{normal}} \mathcal{L}_{\mathrm{normal}}, \tag{22}$$

where $w_{\mathbf{b}}^{\mathrm{comp}}$ upweights locally complex geometry and $\mathcal{L}_{\mathrm{normal}}$ supervises UDF gradients in the near-surface band.

At inference, Stage 3 integrates the clean prediction from $t = 0$ to $t = 1$ with Heun correction, reapplies the gate after every update, and uses the final clean prediction. Shared block vertices are averaged before sparse marching cubes at isovalue $1/\kappa$, followed by mesh cleanup and normal-aware smoothing. Classifier-free guidance is applied to the image-conditioned predictions.

D More Experiments

Material extraction and texture inference with Stage 3. Beyond geometry generation, the Stage 3 model can be reused as a local appearance-aware predictor once geometry has been fixed. In our preliminary PBR experiments, we keep the sparse block layout and the Stage 2 support mask unchanged, and replace the scalar UDF target with per-voxel material attributes.

The training data are constructed directly from a textured mesh. We first detect active blocks using the same geometric UDF pipeline as in geometry generation,

$$\mathcal{B}_{\mathrm{surf}} = \{\mathbf{b} \mid \min(\mathbf{u_b}) < \tau_{\mathrm{surf}}\}. \tag{23}$$

For each retained block, we evaluate a dense near-surface indicator on the fine lattice,

$$\mathbf{r_b}(\mathbf{v}) = \mathbf{1}\left[\mathbf{u_b}(\mathbf{v}) < \tau_{\mathrm{surf}}\right], \qquad \mathbf{v} \in \{1, \ldots, D^3\}, \tag{24}$$

and only sample material values on voxels inside this narrow band. For a voxel center $\mathbf{x}_{\mathbf{b},\mathbf{v}}$, we query the closest surface point on the original textured mesh, interpolate its UV coordinate barycentrically, and bilinearly sample the corresponding material textures. This produces a dense PBR target

$$\mathbf{p_b}(\mathbf{v}) = \begin{cases} \Pi(\mathbf{x}_{\mathbf{b},\mathbf{v}}), & \mathbf{r_b}(\mathbf{v}) = 1, \\ \mathbf{p}_{\mathrm{def}}, & \mathbf{r_b}(\mathbf{v}) = 0, \end{cases} \tag{25}$$

where $\Pi(\cdot)$ denotes closest-surface texture/material sampling and $\mathbf{p}_{\mathrm{def}}$ is the default far-field PBR value. In our implementation, the stored channels follow the order base color (3), metallic (1), roughness (1), and alpha (1).

To remain compatible with the Stage 3 sparse conditioning path, we also derive a coarse support mask by pooling the dense UDF field, while the dense PBR tensor is quantized and stored blockwise. Training then reuses the same sparse Stage 3 backbone as geometry refinement, except that the regression target is now the PBR tensor rather than the UDF.

Algorithm 1 Mesh-to-block PBR target construction

Require: Textured mesh M; optional shape blocks $f_{\text{shape}} = \{C, U\}$
Ensure: PBR blocks $f_{\text{pbr}} = \{C, m_{\text{pbr}}, P_{\text{pbr}}\}$
$d_{\text{pbr}} \leftarrow [0.5, 0.5, 0.5, 0.0, 0.5, 1.0]$
build a BVH over the mesh triangles of M
if f_{shape} is not provided **then**
    detect active blocks C on the 64^3 grid and compute block UDFs U on the 16^3 lattice
end if
for all active blocks $b \in C$ **do**
    initialize dense local PBR tensor $D_b \leftarrow d_{\text{pbr}}$ on 16^3
    define the near-surface mask $M_b \leftarrow \mathbf{1}[U[b] < \tau_{\text{surf}}]$
    for all voxels s with $M_b[s] = 1$ **do**
        $\mathbf{x} \leftarrow \text{BlockLocalToWorld}(b, s)$
        $(f, \beta) \leftarrow \text{BVHNearestFace}(M, \mathbf{x})$
        $\mathbf{u} \leftarrow \text{InterpolateUV}(f, \beta)$ and query the corresponding material
        $D_b[s] \leftarrow [c_r, c_g, c_b, m, r, a]$
    end for
    for all coarse cells q on the 8^3 lattice **do**
        $\Omega_q \leftarrow$ the corresponding $2 \times 2 \times 2$ voxels in D_b
        if $\exists s \in \Omega_q$ with $M_b[s] = 1$ **then**
            $P_{\text{pbr}}[b, q] \leftarrow \text{Average}\{D_b[s] \mid s \in \Omega_q,\ M_b[s] = 1\}$
            $m_{\text{pbr}}[b, q] \leftarrow 1$
        else
            $P_{\text{pbr}}[b, q] \leftarrow d_{\text{pbr}}$ and $m_{\text{pbr}}[b, q] \leftarrow 0$
        end if
    end for
end for
return $\{C, \text{Flatten}(m_{\text{pbr}}),\ \text{Flatten}(\text{Quantize255}(P_{\text{pbr}}))\}$

Algorithm 1 summarizes the mesh-to-block PBR target construction used in these supplementary experiments.

Figure 7 shows representative examples. In each pair, the left object is the extracted geometry without appearance, while the right object uses material channels predicted by the reused Stage 3 model. Although this experiment is still preliminary and does not yet constitute a full end-to-end texturing pipeline, it demonstrates that the fine-stage representation already captures sufficiently rich local context to support texture and material inference in addition to geometry refinement.

Additional qualitative mesh cases. To further illustrate the diversity and geometric quality of meshes generated by GRAVER, we include additional qualitative cases in Figure 9. These examples provide a broader visual sample across different object categories and local geometric structures.



Figure 7: Supplementary qualitative PBR cases for material extraction and texture inference with the reused Stage 3 model. In each composite, textured predictions are paired with their corresponding geometry-only counterparts. Part I.



Figure 8: Supplementary qualitative PBR cases for material extraction and texture inference with the reused Stage 3 model. Part II.

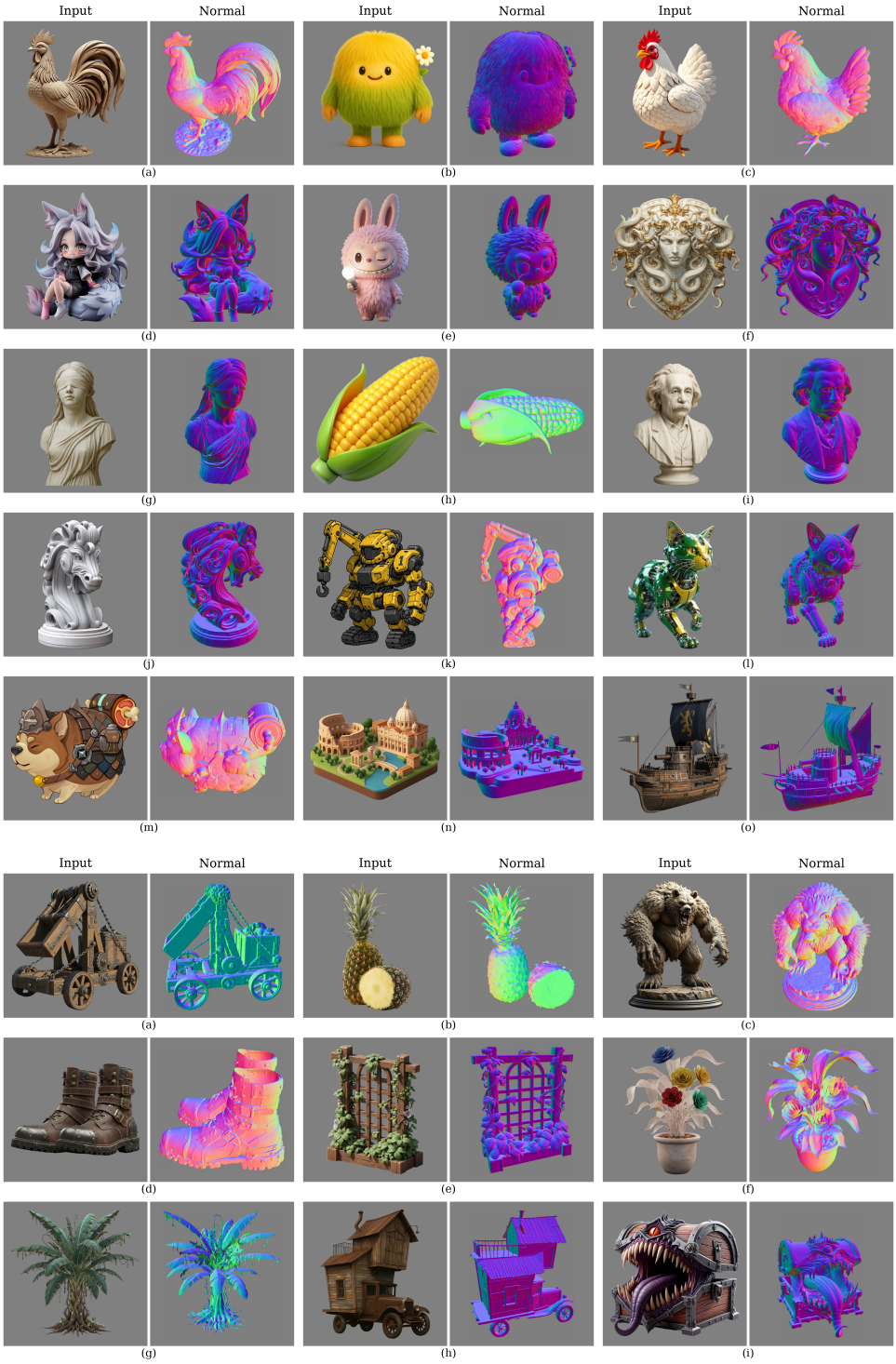

Figure 9: Additional supplementary qualitative mesh cases produced by GRAVER. Part I.