# DirectUV: Image-Conditioned UV Texture Generation with Surface-Aware Positional Encoding

**Jiantao Lin**[1,*] **Yingjie Xu**[1,3,*] **Mingzhi Sheng**[1]
**Yangkai Wei**[3] **Hao Chen**[2] **Ying-Cong Chen**[1,2,†]

[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]The Hong Kong University of Science and Technology
[3]knowin.ai

[*]Equal contribution. [†]Corresponding author.

## Abstract

Generating high-quality UV textures for 3D meshes remains challenging. Multi-view projection pipelines suffer from occlusion and view inconsistency, and recent methods that generate textures directly in UV space still rely on auxiliary modules to supply 3D information, leaving the attention mechanism tied to UV-grid positions rather than to the underlying surface geometry. This mismatch limits coherence across seams and disconnected UV islands. We propose DirectUV, an image-conditioned UV texture diffusion framework that operates in the latent UV space of a pretrained image VAE, in which a Diffusion Transformer denoises the UV latent given a single input image and a coarse UV map. At its core, Surface-Aware Positional Encoding (SAPE) replaces the standard 2D-grid positional encoding with encodings derived from per-token 3D surface coordinates obtained via UV-to-surface correspondence. As positional encodings define the distance metric that attention operates on, SAPE enables tokens to attend to each other based on true surface proximity rather than UV-grid distance, restoring coherence across seams and disconnected islands. A multi-level extension further assigns different attention heads to progressively finer subdivisions of the same latent UV patch, allowing the model to reason about surface structure at multiple granularities. Experiments show that DirectUV produces sharper and more globally consistent textures than other baselines, with the largest improvements in occluded and view-unseen regions where projection-based methods leave gaps or stretched textures.

## 1 Introduction

Generating high-quality textures for 3D meshes requires reconciling two objectives: producing realistic and detailed appearance, and ensuring consistency across the entire 3D surface, free from seams, view-dependent artifacts, and misaligned details.

Existing approaches to generative mesh texturing largely fall into two paradigms. The first leverages powerful 2D diffusion priors to synthesize multi-view images, which are then projected and baked onto the mesh surface [1–8]. While effective at producing realistic local appearance, these methods suffer from multi-view inconsistency and error accumulation during projection, often resulting in seams, misalignment, and unstable textures. The second paradigm generates textures directly in UV space or mesh-native representations [9–11], avoiding projection artifacts by construction. While some of these methods incorporate 3D information through auxiliary modules such as point cloud attention layers [10] or cross-attention with surface coordinates [12], the core attention mechanism still operates on UV-space positions, leaving the model's notion of token proximity tied to the 2D UV layout rather than the underlying 3D surface geometry. As a result, texels that are adjacent on the

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

surface but separated across UV charts remain difficult to relate, limiting coherence around seams and across islands. Recent attempts to extend positional encoding itself with 3D awareness, including 3D-aware rotary embeddings for view-space generation [7, 13] and hierarchical multi-level encodings for general 3D fields [14], are either tied to the multi-view image grid or designed for tasks beyond UV texture synthesis, leaving open the question of how to install surface geometry into UV-space attention itself.

Unlike conditioning signals that the model may selectively attend to, positional encodings define the distance metric that attention is compelled to operate on, making them the natural place to embed surface geometry. Recent analysis of diffusion transformers shows that spatial coherence among patch tokens is primarily governed by positional encodings rather than by token-to-token content interactions [14], reinforcing positional encoding as the structural locus where spatial priors take effect. This observation suggests that, rather than relying on auxiliary modules to supply 3D information, surface geometry should be embedded directly into the attention mechanism. We introduce DirectUV, a diffusion-based framework that generates textures directly in latent UV space, conditioned on a single input image and a coarse UV map, while encoding 3D surface geometry into the positional encodings of the diffusion transformer.

DirectUV operates in the latent space of a pretrained image VAE, with the UV map as the sole generation target. A single input image provides appearance guidance, and a coarse UV map supplies surface-aligned layout cues. Both serve purely as conditioning signals rather than intermediate textures to be refined, preventing view-dependent artifacts from propagating into the final output. To bridge the gap between the 2D UV layout and the underlying 3D surface, we introduce Surface-Aware Positional Encoding (SAPE). Unlike existing methods that supply 3D information through auxiliary conditioning modules, SAPE embeds per-token 3D surface coordinates, obtained through UV-to-surface correspondence, directly into the positional encodings of the diffusion transformer. By redefining the positional encoding in this way, tokens attend to each other based on relative offsets between their ambient 3D coordinates, rather than receiving 3D geometry as an external signal that the model may or may not learn to follow. As a result, surface-adjacent texels can be directly related in attention even when they are disconnected in the UV layout, without requiring additional architectural components. We further extend SAPE to **Multi-Level SAPE**, which exploits the multi-head structure of the transformer by assigning different attention heads positional encodings derived from progressively finer subdivisions of the same latent UV patch. As a result, the model can represent both patch-level surface correspondence and finer sub-patch geometric variation within the existing multi-head attention framework, without introducing additional parameters or modules.

Our contributions are as follows:

- We formulate UV texture generation as a direct image-conditioned generation problem in latent UV space, where a single input image and a coarse UV map serve purely as conditioning signals, rather than generation targets to be projected or refined.
- We introduce Surface-Aware Positional Encoding (SAPE), which embeds per-token 3D surface coordinates into the positional encodings of the diffusion transformer, enabling attention to operate on relative offsets between ambient 3D coordinates rather than UV-grid distance. We further introduce Multi-Level SAPE, which leverages the multi-head structure through progressively finer subdivisions of the same latent UV patch, allowing different heads to capture surface geometry at multiple positional granularities.
- Experiments demonstrate that DirectUV produces sharper, more seam-consistent textures than both projection-based and direct-generation baselines, with the largest gains in occluded and view-unseen regions that are poorly covered by camera-based projection.

## 2 Related Work

### 2.1 Multi-View Projection-Based Texturing

A dominant line of work leverages the strong image prior of large pretrained 2D diffusion models [15, 16] to synthesize textured renderings from multiple viewpoints, typically conditioned on geometry cues such as depth or normals, and then consolidates them into a surface texture via projection, baking, and inpainting. Representative pipelines include [17–22]. This paradigm is appealing as it

reuses powerful 2D diffusion backbones with limited 3D supervision, while naturally supporting text prompts and reference images.

However, inconsistencies in texture detail and illumination remain difficult to fully eliminate, even with joint multi-view denoising. A finite set of viewpoints cannot cover occluded or grazing-angle regions, and the subsequent projection and baking must reconcile mismatched signals across views, leading to accumulated errors such as seam artifacts and local collapse. Recent work has begun incorporating 3D awareness directly into the attention of view-space diffusion models, such as the Paint module in Hunyuan3D-2.1 [13], which adopts the 3D-aware RoPE from RomanTex [7]. Nevertheless, these methods still rely on a projection-and-baking stage to map synthesized views onto the mesh surface.

### 2.2 Direct Texture Generation in UV Space

An alternative line of work avoids multi-view projection by generating textures directly in UV space or other mesh-native representations, eliminating projection artifacts by construction. Point-UV Diffusion [9] adopts a coarse-to-fine pipeline combining point cloud representations with UV-space diffusion to improve structural consistency. TEXGen [10] scales this direction with a large diffusion model that interleaves UV-space convolutions and point cloud attention, enabling feed-forward generation of high-resolution textures conditioned on single-view image and text. TexGarment [12] further incorporates global 3D structure via cross-attention between UV latents and point cloud features. More recently, UniTEX [11] retains a multi-view projection stage but performs completion in a learned 3D feature space using a triplane cube representation, querying unobserved regions and merging them with projected partial textures, this avoids UV chart discontinuities but treats projected pixels as ground truth to preserve rather than as conditioning to be regenerated.

Across these methods, 3D structure is introduced through auxiliary mechanisms rather than being integrated into UV-space modeling itself. Point-UV Diffusion, TEXGen, and TexGarment inject geometry via point cloud features or cross-attention, leaving the UV diffusion process unaware of mesh topology at its core, while UniTEX shifts completion into a 3D feature space instead of enriching UV-space modeling. In contrast, DirectUV remains entirely in UV space and embeds surface geometry directly into the positional encodings of the diffusion transformer, making the diffusion process itself surface-aware without relying on auxiliary modules or non-UV representations.

### 2.3 Positional Encoding in Diffusion Transformers

Diffusion Transformers (DiTs) [23] have emerged as a scalable backbone for high-quality image generation by operating on latent patch tokens. Within this architecture, positional encoding defines the spatial organization of tokens by shaping how attention relates different regions in latent space. Rotary positional encoding (RoPE) [16], widely adopted in modern generative models, injects positional information directly into query–key interactions and therefore determines the distance metric on which attention operates. Unlike conditioning signals that may be selectively attended to or ignored, positional encoding fundamentally governs token-to-token interactions throughout the denoising process, making it a natural place to encode spatial priors. Recent study [14] have further shown that the design of positional encoding significantly affects spatial coherence, controllability, and generalization behavior in diffusion transformers, rather than serving merely as an implementation detail.

Recent work has begun extending positional encoding with explicit 3D awareness. RomanTex [7] introduces 3D-aware rotary embeddings for multi-view image diffusion, and Hunyuan3D-2.1 [13] adopts this design in its Paint module to improve consistency during multi-view texture synthesis. However, these methods still operate on multi-view image grids, where surface-aware information is consumed during view-space denoising and only later projected back onto the mesh surface. As a result, coherence across UV seams and disconnected surface regions must still be recovered during the projection and baking stage. In contrast, DirectUV operates directly on UV-space latent tokens, allowing surface-aware attention to participate in the texture generation process itself. This enables seam-crossing surface relationships to be resolved during denoising rather than deferred to a subsequent projection step.

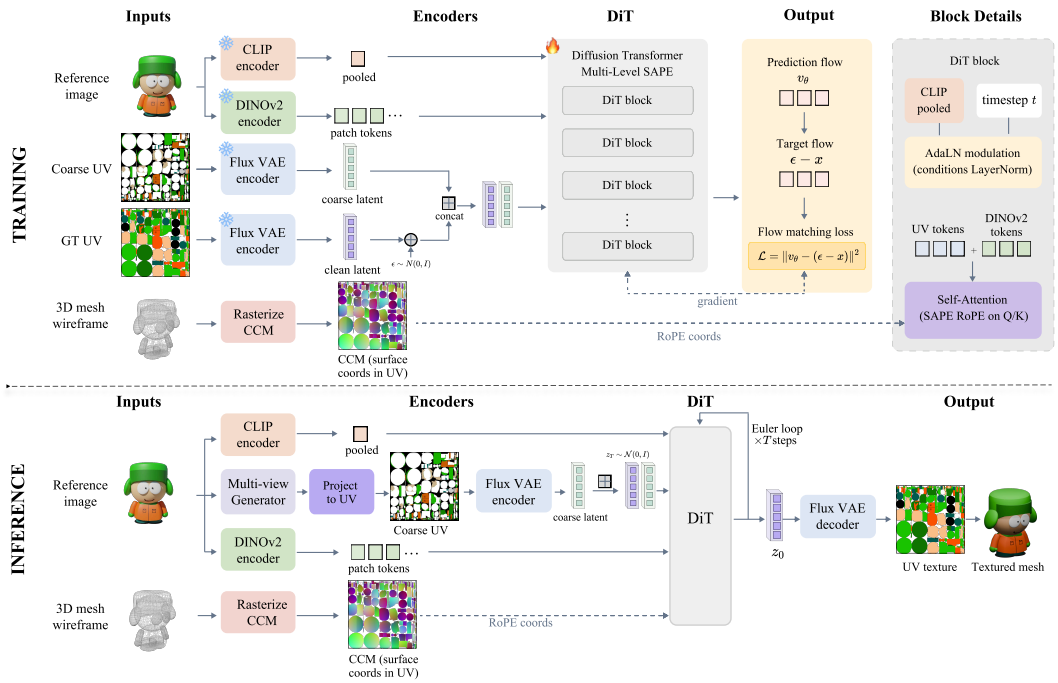


Figure 1: Overview of DirectUV. **Top (training).** The reference image is encoded by CLIP [24] into a pooled feature that drives AdaLN modulation, and by DINOv2 [25] into patch tokens that feed the joint-attention stream. A coarse UV map is encoded by the frozen Flux VAE and channel-concatenated with the noisy UV latent to form the DiT input. The CCM supplies per-head RoPE coordinates for Multi-Level SAPE. The model is trained with the rectified flow-matching objective. **Bottom (inference).** The reference image drives both an off-the-shelf multi-view generator, whose outputs are projected to a coarse UV map, and the CLIP/DINOv2 encoders. Under this conditioning, $N$ Euler steps denoise $\mathbf{z}_1$ to $\mathbf{z}_0$, which is decoded by the frozen Flux VAE and wrapped onto the mesh as the final texture.

# 3 Method

## 3.1 Overview

Since positional encoding governs the distance metric of attention, embedding surface coordinates into PE offers a direct path to surface-aware denoising without auxiliary 3D modules. DirectUV formulates image-conditioned UV texture generation as a diffusion process directly in UV space, with the UV map as the sole generation target. The pipeline operates in a latent UV space. Given an input image and a mesh with UV parameterization, the UV map is encoded into a compact latent representation by a pretrained image VAE. The same UV parameterization additionally yields a Canonical Coordinate Map (CCM), a UV-space image whose value at each pixel is the 3D surface coordinate of the corresponding mesh point. The CCM is precomputed once per mesh and provides the geometric input from which Surface-Aware Positional Encoding (SAPE) derives per-token surface coordinates. A diffusion transformer (DiT) then models the conditional generation process in this latent space, guided by the input image and a coarse UV map, with SAPE integrated into its attention layers to inject 3D surface geometry directly into token-to-token interactions. The predicted latent is finally decoded back to obtain the UV texture. An overview of the full training and inference pipeline is shown in Figure 1. In the following, we first introduce the latent UV diffusion formulation, then describe the conditioning strategy for incorporating image and UV cues, and finally present the design of SAPE for enabling surface-aware attention in UV-space generation.

## 3.2 Latent UV Diffusion Formulation

We model UV texture generation as a conditional diffusion process in a latent UV space. A UV map, defined as an image-like signal over a regular 2D grid, is encoded into a spatially downsampled latent representation $\mathbf{x} = \mathcal{E}(I_{\text{UV}})$ using a frozen Flux [15] VAE encoder, on which all denoising is

performed, and decoded back by $\mathcal{D}$ to obtain the final texture. This representation naturally aligns with pretrained 2D image priors, while the VAE reconstructs UV maps with negligible error, allowing the model to operate in a compact space without sacrificing fine-grained appearance.

With the UV map as the sole generation target, texture synthesis reduces to a single diffusion process, avoiding the inconsistencies introduced by projecting and blending intermediate views. The latent representation preserves the UV coordinate structure, providing a direct path for surface geometry to enter through positional encoding rather than through auxiliary modules or external 3D features. We instantiate $v_\theta$ using the Flux DiT architecture, and introduce the conditioning bundle $\mathbf{c}$ together with Surface-Aware Positional Encoding (SAPE) within its attention layers, whose details are presented in the following subsections.

### 3.3 Conditioning in UV-Space Generation

The conditioning bundle $\mathbf{c}$ consists of two complementary signals: a reference image providing appearance information and a coarse UV map encoding surface-aligned layout. These signals are injected into the DiT through different pathways depending on whether they share the UV coordinate system of the target texture.

**Image condition.** The reference image is incorporated through two streams aligned with how the DiT processes external inputs. A pooled CLIP feature captures global appearance attributes such as color, material tone, and identity, and is injected through the modulation pathway used for timestep and text embeddings in Flux, conditioning all tokens uniformly. In parallel, dense patch tokens from DINOv2 provide spatially localized details and are fed into the joint-attention stream, where they effectively replace text tokens in the original architecture. This enables UV latent tokens to attend to relevant image regions for fine-grained appearance cues.

**Coarse UV map.** The coarse UV map is obtained by projecting one or more input views into UV space and provides surface-aligned layout guidance. Since it shares the UV coordinate system with the target texture, it is injected through a position-aligned pathway. The same Flux VAE encodes it into a latent representation, which is concatenated with the noisy UV latent along the channel dimension before patchification. This per-position concatenation preserves one-to-one spatial correspondence with the latent being denoised, unlike the image condition, which relies on modulation and cross-attention due to the lack of coordinate alignment.

In contrast to projection-based pipelines that treat projected UV maps as intermediate results for refinement or inpainting, both signals act purely as conditioning. The UV texture is generated from scratch in latent space, allowing DirectUV to leverage multi-view cues without inheriting view-dependent inconsistencies.

### 3.4 Surface-Aware Positional Encoding (SAPE)

**Spatial Metric Mismatch.** In a diffusion transformer, positional encodings define the spatial metric over which attention relates tokens, fixing which regions are treated as neighbors during denoising. For UV texture generation this metric is critical: appearance should propagate according to proximity on the mesh surface rather than proximity on the flattened UV plane.

Standard DiTs use positional encodings on a regular 2D grid, so token relationships are measured in UV coordinates. This induces a mismatch between the parameterization and the underlying surface: texels that are immediate neighbors on the mesh can be split across a UV seam onto disconnected islands and end up arbitrarily far apart under the UV-grid metric (Figure 2, left), while texels nearby in the UV plane may correspond to distant surface regions. The 2D grid therefore supplies attention with an incorrect neighborhood prior, limiting coherence across seams and surface-disconnected UV regions.

**Surface-Aware RoPE.** We address this mismatch with **Surface-Aware Positional Encoding (SAPE)**, which replaces the 2D UV-grid positional encoding with positional codes derived from the CCM. Let $\mathbf{x}_i$ denote the $i$-th token of the latent $\mathbf{x}$ and $U_i$ the UV region it covers. We downsample the CCM to the resolution of the token grid so that $U_i$ maps to a single CCM pixel, whose value $\mathcal{S}_i \in \mathbb{R}^3$ gives the per-token surface coordinate. For any pair of tokens $\mathbf{x}_i, \mathbf{x}_j$ at head $h$, SAPE rotates their query and key as

$$\tilde{\mathbf{Q}}_i^{(h)} = \mathrm{RoPE}(\mathbf{Q}_i^{(h)}; \mathcal{S}_i), \qquad \tilde{\mathbf{K}}_j^{(h)} = \mathrm{RoPE}(\mathbf{K}_j^{(h)}; \mathcal{S}_j). \tag{1}$$

Following the standard axis-factorized form of RoPE, the head's feature dimension is split into three contiguous groups corresponding to the $(x, y, z)$ axes of $\mathcal{S}_i$, with each group rotated independently by its own coordinate. The dot product $\langle \tilde{\mathbf{Q}}_i^{(h)}, \tilde{\mathbf{K}}_j^{(h)} \rangle$ therefore depends only on the relative offset $\mathcal{S}_j - \mathcal{S}_i$ on the 3D surface rather than on the 2D UV grid. Texels nearby on the surface can therefore attend to each other directly, even when separated in UV space.

**Multi-Level SAPE.** A single per-token surface coordinate cannot simultaneously capture broad surface correspondence across UV islands and fine local structure around seams. We therefore extend SAPE to a multi-level form that reads the CCM at $L$ resolutions over the same latent patch. At level $l \in \{0, \dots, L-1\}$, the patch is subdivided into a $2^l \times 2^l$ layout, yielding $4^l$ sub-cell surface coordinates per token. We denote any one of them by $\mathcal{S}_i^{(l)}$, so the basic SAPE above corresponds to $l = 0$ with $\mathcal{S}_i = \mathcal{S}_i^{(0)}$. These per-level coordinates are then distributed across the head dimension of the diffusion transformer. Multi-head attention partitions the per-token feature into $H$ contiguous slices of size $d_{\text{head}}$, with each head rotating its own slice independently. Each head $h$ is assigned a level $l_h \in \{0, \dots, L-1\}$ and a specific sub-cell within that level. This replaces $\mathcal{S}_i$ in the rotation above with the head-specific coordinate $\mathcal{S}_i^{(l_h)}$, so the head's queries and keys are rotated using a single 3D surface coordinate at one granularity.

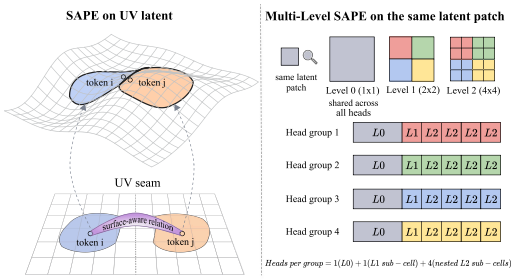


Figure 2: Mechanism of SAPE and Multi-Level SAPE. **Left.** Tokens $i$ and $j$ are adjacent on the mesh but separated onto different UV islands by a seam. SAPE measures their attention distance on the surface, allowing direct interaction across the seam. **Right.** Multi-Level SAPE reads the CCM at $L$ resolutions and assigns the resulting positional encodings to the $H$ heads, organized into $G$ groups (one cell per head, color denotes group).

The $H$ heads are organized into $G$ groups, each tied to one sub-cell of the level-1 layout. Within a group (for $L = 3$), one head reads the level-0 coordinate, one head reads the group's level-1 sub-cell, and the remaining four heads read the level-2 sub-cells nested inside it. The $G$ groups together partition the level-1 layout, so every sub-cell at any level is read by exactly one head across the head dimension. The specific values of $L$, $H$, $G$, and the per-level head budget are given in Section 4.2.

A token's feature vector therefore simultaneously carries surface information at all $L$ granularities, but on disjoint head slices: each head receives a single CCM coordinate rather than a stacked superposition of multiple levels. Since attention is computed per head and then mixed by the output projection, multi-level surface evidence is integrated downstream of attention, while inside attention each head reasons under a clean single-granularity positional metric. No additional latent maps, parameters, or auxiliary geometry branches are introduced. Figure 2 illustrates how SAPE establishes surface-aware interaction directly on the UV latent map and how each head within a group receives one specific CCM sub-cell.

### 3.5 Training and Inference

**Training objective.** We train $v_\theta$ under the rectified flow-matching formulation [26]. Given a clean UV latent $\mathbf{x}$, a noise sample $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and a timestep $t \in [0, 1]$, we form the noisy latent $\mathbf{z}_t = (1-t)\,\mathbf{x} + t\,\boldsymbol{\epsilon}$ and train the network to predict the corresponding flow direction $\boldsymbol{\epsilon} - \mathbf{x}$ via

$$\mathcal{L}_\theta = \mathbb{E}_{t,\mathbf{x},\boldsymbol{\epsilon},\mathbf{c}}\big[\, w(t)\, \|v_\theta(\mathbf{z}_t, t, \mathbf{c}) - (\boldsymbol{\epsilon} - \mathbf{x})\|_2^2 \,\big], \tag{2}$$

where $\mathbf{c}$ is the conditioning bundle defined in Section 3.3. We sample $t$ uniformly from $[0, 1]$ and use a uniform reweighting $w(t) \equiv 1$. Note that the CCM is not part of $\mathbf{c}$ and enters the model exclusively through SAPE.

**Inference.** During inference, given a reference image and a 3D mesh, the coarse UV map is produced from the reference image by an off-the-shelf multi-view generator followed by view-to-UV projection, and the CCM is rasterized once from the mesh. Starting from $\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we iterate the Euler update for $N$ denoising steps. The conditioning bundle $\mathbf{c}$ and the CCM are deterministic across steps

and reused without re-estimation, the final latent is decoded by the frozen Flux VAE to recover the UV texture.

# 4 Experiments

## 4.1 Dataset

We construct our training set from three large-scale 3D asset datasets: TexVerse [27], PartNext [28], and Objaverse [29]. After merging and filtering out meshes with excessive face counts, severely fragmented UV layouts, low-quality UV parameterizations, or invalid UV-space supervision, we retain approximately 100K textured meshes.

For each mesh we generate a UV mask, a ground-truth UV texture, and a CCM, all at $1024 \times 1024$ resolution. We also render four canonical RGB views at azimuth angles $\{0°, 90°, 180°, 270°\}$ with elevation $5°$. The front view serves as the reference image fed to DINOv2 and CLIP.

For latent normalization, we precompute the per-position per-channel mean and standard deviation of VAE-encoded UV latents over the training set, and apply them during training and inference.

## 4.2 Implementation Details

Our UV diffusion model is a Flux-style transformer with 6 MMDiT blocks and 12 single-stream blocks, using $H = 24$ attention heads with head dimension $d_{\text{head}} = 120$ and axes dimensions $(40, 40, 40)$. For Multi-Level SAPE, we use $L = 3$ subdivision levels and partition the $H$ heads into $G = 4$ groups of six. Within each group, one head reads the level-0 coordinate, one reads its level-1 sub-cell, and the remaining four read the four level-2 sub-cells nested inside, giving a 4:4:16 ratio across levels. Image conditioning uses DINOv2 ViT-L/14 with registers and CLIP ViT-B/32. For UV encoding and decoding, we reuse the frozen VAE from FLUX.1-dev [15], which has $8\times$ spatial downsampling and 64 latent channels. The noisy target UV latent and the VAE-encoded coarse UV are concatenated along the channel dimension, yielding 128 input channels and 64 output channels at the patchify layer.

The coarse UV map is constructed online by projecting rendered views into UV space using the known mesh geometry. At each step, we randomly select $0$, $1$, $2$, or $4$ views, where the zero-view case corresponds to a blank UV map. To simulate realistic view-level inconsistencies, some views are rendered with lighting while others use albedo colors, introducing illumination variation across the projected UV and making it an imperfect conditioning signal rather than a target to be directly copied.

We train on 8 NVIDIA A800 GPUs (Accelerate) with batch size 16 per GPU for 170K steps using 8-bit AdamW ($10^{-5}$ lr, $10^{-4}$ wd, betas $(0.9, 0.999)$, clip 1.0), bf16 and gradient checkpointing. At inference, we run 28 denoising steps with `FlowMatchEulerDiscreteScheduler`; coarse UV maps are obtained via Kiss3dGen [20] followed by projection to the mesh.

## 4.3 Evaluation

We evaluate on 200 objects randomly sampled from the Google Scanned Objects (GSO) [30] dataset, which is disjoint from our training data and thus serves as an out-of-distribution test set. For each object, we render a single front-facing view as the reference image input for all methods.

For quantitative evaluation, each textured mesh is rendered under albedo shading from a set of viewpoints providing full directional coverage, including top-down and bottom-up views. Renderings are compared against ground-truth albedo images, and we report PSNR and SSIM for per-pixel fidelity, along with FID [31] and KID [32] for distributional similarity.

## 4.4 Comparison with State-of-the-Art Methods

We compare our method with representative 3D texture generation baselines. TEXGen [10] generates mesh textures via a hybrid diffusion model that couples 2D UV-space convolutions with 3D point cloud attention, exchanging features between the two representations to achieve surface-aware texture synthesis. FlexPainter [6] unifies multiple control signals within a texture generation diffusion

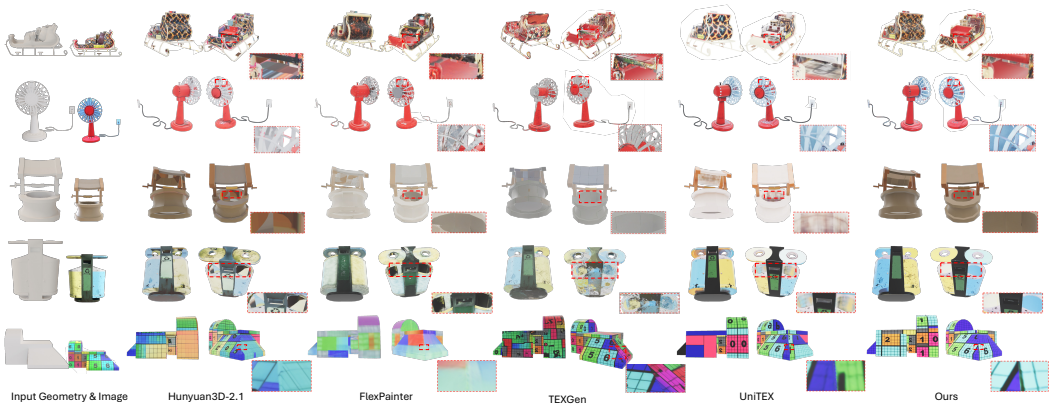


Figure 3: Qualitative comparison with state-of-the-art texture generation methods. **(i) Finer details:** our results preserve sharper high-frequency patterns, e.g., the line strokes in the number block. **(ii) Occluded regions:** for heavily occluded interiors such as the well and the trash bin, our method produces coherent and plausible content, while others yield blurry or incomplete textures. **(iii) Robustness to multi-view inconsistency:** by denoising directly in latent UV space, our method avoids being misled by conflicting views. In the fan case, UniTEX incorrectly transfers the back-cover texture onto the front blades, whereas ours preserves the correct appearance.

| Method | PSNR ↑ | SSIM ↑ | FID ↓ | KID ↓ |
|---|---|---|---|---|
| TEXGen [10] | 24.45 | 0.9417 | 44.09 | 34.32 |
| FlexPainter [6] | 23.21 | 0.9463 | 47.04 | 39.25 |
| Hunyuan3D-2.1 [13] | <u>24.74</u> | 0.9450 | <u>39.872</u> | <u>33.34</u> |
| UniTEX [11] | 24.23 | <u>0.9504</u> | 42.11 | 39.21 |
| Ours | **25.13** | **0.9527** | **39.17** | **30.03** |

Table 1: Quantitative comparison on the GSO dataset. Best results are shown in bold, and second-best results are underlined.

framework, supporting text, image, and multi-modal inputs. Hunyuan3D-2.1 [13] synthesizes multi-view images with 3D-aware RoPE in its Paint module and bakes the result into a UV texture. UniTEX [11] retains the multi-view projection stage and completes the resulting partial textured mesh through a Large Texturing Model that produces a triplane feature volume. All methods receive the same input meshes and reference images.

**Quantitative Results.** Table 1 reports the quantitative comparison. Our method achieves the best PSNR and SSIM among all compared methods, demonstrating superior reference fidelity and reconstruction accuracy. In addition, our method obtains competitive FID and KID scores, indicating that it preserves perceptual realism while maintaining stable texture consistency. Across all metrics, our method achieves the best overall balance between reconstruction fidelity and perceptual quality.

**Qualitative Results.** All qualitative renderings are produced under identical lighting conditions applied uniformly to all methods. We compare our method with the strongest baselines on a diverse set of meshes, as illustrated in Figure 3. Our results consistently exhibit finer texture details, as reflected in the sharp recovery of high-frequency patterns on the number block, where competing methods tend to blur or misalign thin structures. In heavily occluded regions, such as the inner walls of the well and the trash bin, our method produces coherent and semantically plausible content, while projection-based pipelines often yield blurry or fragmented artifacts due to limited visible evidence. The fan example further demonstrates robustness to multi-view inconsistency. UniTEX treats projected multi-view results as the target to be completed via inpainting, and thus transfers the back-cover texture onto the front blades when the views are inconsistent. In contrast, our method operates on the UV latent with conditioning signals, resolving such conflicts through global reasoning and preserving the correct appearance.

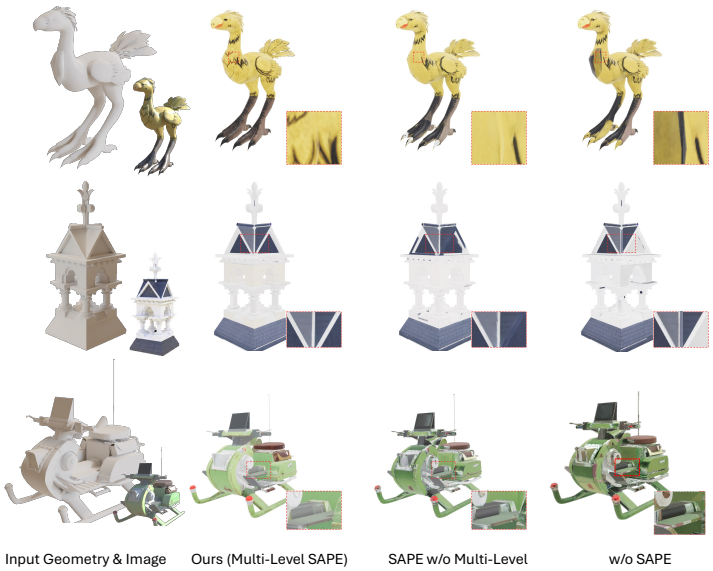


Figure 4: Qualitative ablation of Surface-Aware Positional Encoding. **Without SAPE**, texels that are adjacent on the 3D surface but split into different UV islands cannot interact properly, leading to **severe appearance divergence** in some regions, as seen in the chocobo case where the black texture from the legs is incorrectly generated on the neck. **Removing only the multi-level design** preserves cross-island coherence but loses spatial precision, yielding **blurry textures**.

### 4.5 Ablation Study

**Effect of Surface-Aware Positional Encoding.** To isolate the contribution of SAPE, we replace it with the standard 2D UV-grid RoPE used by Flux while keeping the rest of the model unchanged, yielding a pure latent UV-space DiT with no surface coordinate information in its positional encoding. As shown in Table 2 and Figure 4, this variant drops across all metrics, with the most visible degradation on regions that surface-disconnected islands. Attention can only relate tokens by UV-plane proximity, so mesh-adjacent texels split across charts cannot interact directly, producing seam mismatches and inconsistent appearance between disconnected UV islands of the same surface region.

| Variant | PSNR ↑ | SSIM ↑ | FID ↓ | KID ↓ |
|---|---|---|---|---|
| w/o SAPE | 23.82 | 0.9376 | 58.42 | 72.36 |
| SAPE w/o Multi-Level | <u>24.36</u> | <u>0.9462</u> | <u>40.86</u> | <u>36.74</u> |
| Ours (Multi-Level SAPE) | **25.13** | **0.9527** | **39.17** | **30.03** |

Table 2: Ablation of SAPE and its multi-level design on the GSO dataset.

**Effect of Multi-Level Design.** Holding SAPE in place, we ablate only its multi-level structure by tying all $H$ heads to a single level-0 coordinate $\mathcal{S}_i = \mathcal{S}_i^{(0)}$. As illustrated in Table 2 and Figure 4, this single-level variant retains seam coherence but produces blurry textures. With every head reading the same coordinate per token, attention cannot distinguish sub-token positions inside a latent patch, so within-patch detail collapses to a near-uniform appearance after decoding. Multi-level coordinates expose finer sub-patch granularities to different heads, breaking this ambiguity and restoring high-frequency detail.

## 5 Conclusion

We presented DirectUV, an image-conditioned UV texture diffusion framework that generates textures directly in latent UV space, using projected multi-view results only as conditioning signals. The key idea is Surface-Aware Positional Encoding (SAPE), which embeds surface geometry into positional encodings so that attention operates on true surface proximity, enabling coherent synthesis across seams without auxiliary 3D modules. A multi-level design further allows reasoning at multiple spatial scales. DirectUV outperforms other baselines, especially in occluded and view-unseen regions. A remaining limitation is the reliance on UV parameterization quality, which future work may address.

## Acknowledgments and Disclosure of Funding

This research received no specific grant from any funding agency. Computational resources were provided by the Envision Lab, led by Prof. Ying-Cong Chen, at The Hong Kong University of Science and Technology (Guangzhou). The authors declare no competing interests.

## References

[1] R. Chen, Y. Chen, N. Jiao, and K. Jia, "Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 22 246–22 256.

[2] E. Richardson, G. Metzer, Y. Alaluf, R. Giryes, and D. Cohen-Or, "Texture: Text-guided texturing of 3d shapes," in *ACM SIGGRAPH 2023 conference proceedings*, 2023, pp. 1–11.

[3] D. Z. Chen, Y. Siddiqui, H.-Y. Lee, S. Tulyakov, and M. Nießner, "Text2tex: Text-driven texture synthesis via diffusion models," *arXiv preprint arXiv:2303.11396*, 2023.

[4] X. Zeng, X. Chen, Z. Qi, W. Liu, Z. Zhao, Z. Wang, B. Fu, Y. Liu, and G. Yu, "Paint3d: Paint anything 3d with lighting-less texture diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 4252–4262.

[5] W. Cheng, J. Mu, X. Zeng, X. Chen, A. Pang, C. Zhang, Z. Wang, B. Fu, G. Yu, Z. Liu *et al.*, "Mvpaint: Synchronized multi-view diffusion for painting anything 3d," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 585–594.

[6] D. Yan, L. Wu, J. Lin, L. Wang, T. Xu, Z. Chen, Z. Yang, L. Xu, S. Zhang, and Y. Chen, "Flexpainter: Flexible and multi-view consistent texture generation," *arXiv preprint arXiv:2506.02620*, 2025.

[7] Y. Feng, M. Yang, S. Yang, S. Zhang, J. Yu, Z. Zhao, Y. Liu, J. Jiang, and C. Guo, "Romantex: Decoupling 3d-aware rotary positional embedded multi-attention network for texture synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025, pp. 17 203–17 213.

[8] Y. Georgiou, M. Loizou, M. Averkiou, and E. Kalogerakis, "Im2surftex: Surface texture generation via neural backprojection of multi-view images," *arXiv preprint arXiv:2502.14006*, 2025.

[9] X. Yu, P. Dai, W. Li, L. Ma, Z. Liu, and X. Qi, "Texture generation on 3d meshes with point-uv diffusion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4206–4216.

[10] X. Yu, Z. Yuan, Y.-C. Guo, Y.-T. Liu, J. Liu, Y. Li, Y.-P. Cao, D. Liang, and X. Qi, "Texgen: a generative diffusion model for mesh textures," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 6, pp. 1–14, 2024.

[11] Y. Liang, K. Luo, X. Chen, R. Chen, H. Yan, W. Li, J. Liu, and P. Tan, "Unitex: Universal high fidelity generative texturing for 3d shapes," *arXiv preprint arXiv:2505.23253*, 2025.

[12] J. Liu, J. Wu, X. Gao, J. Hu, B. Xiong, X. Liu, C. Zhao, H. Pei, H. Feng, Y. Li *et al.*, "Texgarment: Consistent garment uv texture generation via efficient 3d structure-guided diffusion transformer," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 26 566–26 575.

[13] T. H. Team, "Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material," 2025.

[14] Y. Bai, H. Li, and Q. Huang, "Positional encoding field," *arXiv preprint arXiv:2510.20385*, 2025.

[15] B. F. Labs, "Flux," https://github.com/black-forest-labs/flux, 2024.

[16] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," *Neurocomputing*, vol. 568, p. 127063, 2024.

[17] R. Liu, R. Wu, B. V. Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick, "Zero-1-to-3: Zero-shot one image to 3d object," 2023.

[18] Z. Huang, Y.-C. Guo, H. Wang, R. Yi, L. Ma, Y.-P. Cao, and L. Sheng, "Mv-adapter: Multi-view consistent image generation made easy," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025, pp. 16 377–16 387.

[19] X. Long, Y.-C. Guo, C. Lin, Y. Liu, Z. Dou, L. Liu, Y. Ma, S.-H. Zhang, M. Habermann, C. Theobalt *et al.*, "Wonder3d: Single image to 3d using cross-domain diffusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 9970–9980.

[20] J. Lin, X. Yang, M. Chen, Y. Xu, D. Yan, L. Wu, X. Xu, L. Xu, S. Zhang, and Y.-C. Chen, "Kiss3dgen: Repurposing image diffusion models for 3d asset generation," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5870–5880.

[21] X. Yang, J. Lin, Y. Xu, H. Li, and Y. Chen, "Advancing high-fidelity 3d and texture generation with 2.5 d latents," *arXiv preprint arXiv:2505.21050*, 2025.

[22] Y. Liu, M. Xie, H. Liu, and T.-T. Wong, "Text-guided texturing by synchronized multi-view diffusion," *arXiv preprint arXiv:2311.12891*, 2023.

[23] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 4195–4205.

[24] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *ICML*, 2021.

[25] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, R. Howes, P.-Y. Huang, H. Xu, V. Sharma, S.-W. Li, W. Galuba, M. Rabbat, M. Assran, N. Ballas, G. Synnaeve, I. Misra, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski, "Dinov2: Learning robust visual features without supervision," 2023.

[26] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," *arXiv preprint arXiv:2209.03003*, 2022.

[27] Y. Zhang, L. Zhang, R. Ma, and N. Cao, "Texverse: A universe of 3d objects with high-resolution textures," *arXiv preprint arXiv:2508.10868*, 2025.

[28] P. Wang, Y. He, X. Lv, Y. Zhou, L. Xu, J. Yu, and J. Gu, "Partnext: A next-generation dataset for fine-grained and hierarchical 3d part understanding," *arXiv preprint arXiv:2510.20155*, 2025.

[29] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi, "Objaverse: A universe of annotated 3d objects," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 13 142–13 153.

[30] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke, "Google scanned objects: A high-quality dataset of 3d scanned household items," in *2022 International Conference on Robotics and Automation (ICRA)*. Ieee, 2022, pp. 2553–2560.

[31] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *Advances in neural information processing systems*, vol. 30, 2017.

[32] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying mmd gans," *arXiv preprint arXiv:1801.01401*, 2018.

# Appendix

## A Coarse UV as Soft Conditioning

The coarse UV map assembled from multi-view projections provides a surface-aligned layout cue, but it can carry inaccurate or misleading color in regions where the projected views do not faithfully represent the true surface appearance. DirectUV treats this map as a soft conditioning signal and does not directly copy its values into the final texture; instead, the denoising process can deviate from the coarse UV when the model's learned prior suggests a more coherent appearance.

Figure 5 illustrates this on a keyboard example. The coarse UV renders the underside of the chassis as white, yet the reference image and surrounding surface clearly indicate a continuous dark metallic casing. The final texture correctly assigns the metal material to the bottom face, overriding the white coarse UV through global reasoning over the full surface. This behavior confirms that the coarse UV guides rather than constrains the generation.

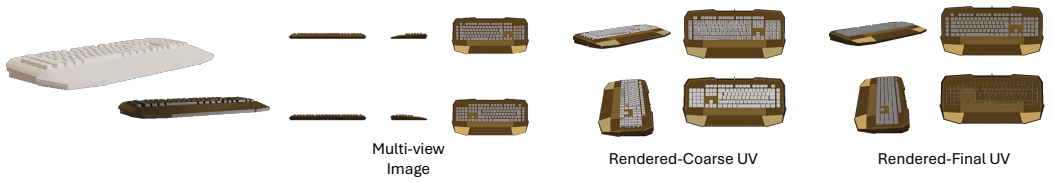


Figure 5: The keyboard underside appears white in the coarse UV (center) despite being covered by the bottom-view projection, but is correctly recovered as dark metallic casing in the final texture (right), consistent with the reference image. DirectUV's learned prior overrides the inaccurate coarse UV values rather than copying them.

We quantify this behavior in Table 3. We use the same coarse UV map directly as the mesh texture, render it from the evaluation cameras, and compare it against DirectUV *only* at pixels corresponding to projection-covered texels. Restricting the comparison in this way excludes any gain that would come from merely filling empty texels, so the remaining improvement isolates how DirectUV treats projected content that already exists. DirectUV improves all three metrics, showing that it corrects the projected values rather than copying them.

Table 3: DirectUV improvement over rendering the coarse UV map directly as the mesh texture, evaluated only at coarse-covered pixels.

| Region | $\Delta$PSNR $\uparrow$ | $\Delta$SSIM $\uparrow$ | LPIPS reduction $\uparrow$ |
|---|---|---|---|
| Coarse-covered | +1.11 | +0.0301 | +0.03 |

## B Robustness to Coarse UV Completeness

During training, the number of views used to construct the coarse UV is randomly sampled from $\{0, 1, 2, 4\}$ at each step, exposing the model to a wide range of coarse UV completeness and preventing it from over-relying on any single input quality level. Figure 6 shows that this strategy yields a model robust to incomplete coarse UV inputs at inference time. Even in the zero-view setting where the coarse UV is entirely blank, DirectUV produces plausible textures by relying on the reference image and geometry-conditioned denoising alone, confirming that the coarse UV is treated as a soft conditioning signal rather than a hard initialization.

Table 4 reports the corresponding quantitative behavior, varying only the number of projected views while keeping all other inputs fixed. The first projected view provides most of the PSNR/SSIM improvement, after which these metrics largely saturate, while FID and KID continue to improve

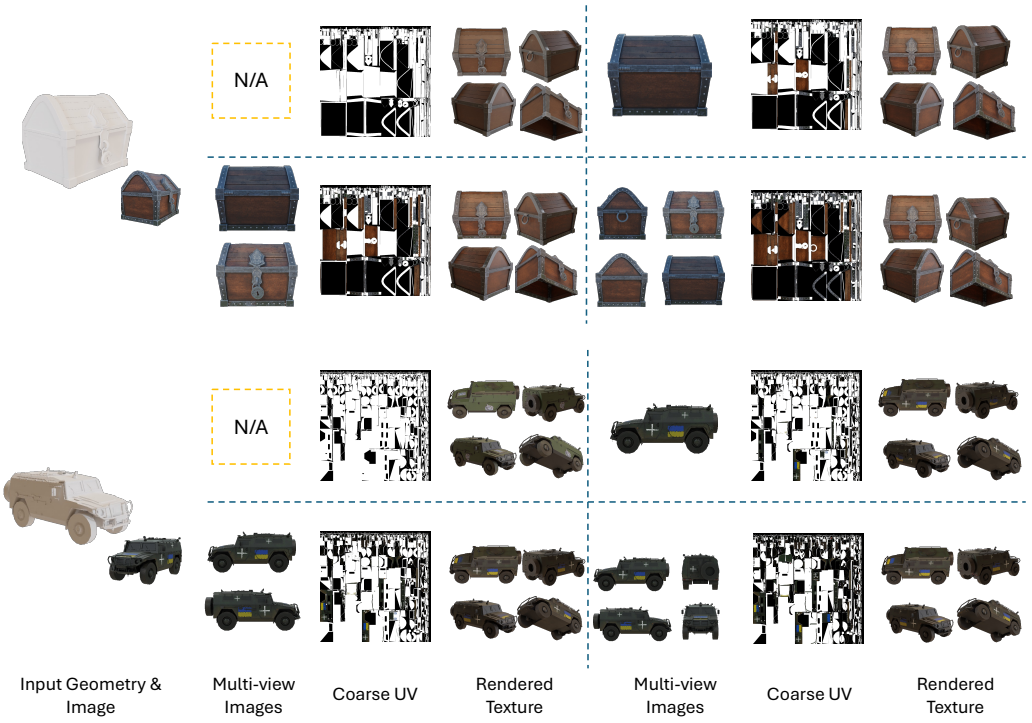


Figure 6: Texture generation results under varying coarse UV completeness at inference time (0, 1, 2, and 4 projected views). The reference image and mesh are identical across all cases.

with additional views. This indicates that a single view supplies most of the pixel-level alignment, whereas further views mainly enrich reference-specific detail and distributional quality.

Table 4: Effect of the number of projected views used to construct the coarse UV.

| Projected views | PSNR ↑ | SSIM ↑ | FID ↓ | KID ↓ |
|---|---|---|---|---|
| 0 | 23.07 | 0.9116 | 70.92 | 86.97 |
| 1 | 25.08 | 0.9521 | 50.43 | 57.09 |
| 2 | 25.11 | 0.9523 | 42.17 | 40.28 |
| 4 | **25.13** | **0.9527** | **39.17** | **30.03** |

## C Positional Injection versus Auxiliary Geometry Conditioning

SAPE uses surface coordinates in the positional mechanism rather than in a conditioning branch. To determine whether its gains arise from this placement or simply from access to the CCM, we train an identical-backbone variant that concatenates the normalized CCM with the latent spatial channels while retaining standard 2D UV RoPE. All other components, training data, and conditioning signals remain fixed.

Table 5 shows that channel concatenation provides only modest gains. Although the backbone can use the auxiliary geometry, it must implicitly infer the coordinate relationships among UV tokens. Using the same coordinates to define positional relations in self-attention yields substantially larger gains across all metrics at negligible computational cost. The comparison between Single-Level and Multi-Level SAPE further isolates the benefit of sub-patch surface sampling.

Table 5: Controlled comparison of geometry-injection mechanisms under an identical backbone, training set, and conditioning pipeline. Only the way the CCM enters the model differs.

| Variant | Geometry injection | PSNR ↑ | SSIM ↑ | FID ↓ | KID ↓ |
|---|---|---|---|---|---|
| w/o SAPE (2D RoPE) | None | 23.82 | 0.9376 | 58.42 | 72.36 |
| CCM concatenation + 2D RoPE | Latent-channel concatenation | 23.97 | 0.9395 | 50.62 | 66.68 |
| Single-Level SAPE | CCM coordinates in RoPE | <u>24.36</u> | <u>0.9462</u> | <u>40.86</u> | <u>36.74</u> |
| Multi-Level SAPE | Multi-level CCM coordinates in RoPE | **25.13** | **0.9527** | **39.17** | **30.03** |

# D Seam Consistency

UV seams separate texels that are adjacent on the mesh surface. We therefore evaluate texture quality specifically near seams. We identify mesh edges whose incident faces are continuous in 3D but map to different UV islands, then sample matched strips just inside both islands using identical surface interpolation parameters. The strips are joined along their virtual surface adjacency and packed into fixed-format per-object atlases for comparison with the corresponding ground truth.

Alongside PSNR, SSIM, FID, and KID on these atlases, we report a *GT-referenced seam discrepancy*. For matched texels one texel inside each island, we compute the mean absolute cross-seam RGB jump in the prediction and ground truth, then average the absolute difference between the two jumps over all seam samples. With RGB normalized to $[0, 1]$, the metric also lies in $[0, 1]$, and lower is better. Referencing the ground truth avoids penalizing genuine texture discontinuities or rewarding over-smoothed predictions that remove all cross-seam contrast.

Table 6 shows that channel concatenation modestly improves over 2D RoPE: access to 3D coordinates helps, but does not overcome the positional discontinuity between UV islands. Single-Level SAPE substantially improves every seam metric by using these coordinates in attention, and Multi-Level SAPE improves further, indicating that sub-patch surface locations are particularly useful near seam boundaries.

Because seam neighborhoods are thin, high-frequency strips that directly expose cross-island discontinuities, their absolute scores are much lower than global metrics and should be interpreted comparatively. The low 2D RoPE baseline reflects the failure mode SAPE is designed to address.

Table 6: Texture quality on matched UV-seam regions. All variants share the same backbone, training set, and conditioning.

| Variant | Seam PSNR ↑ | Seam SSIM ↑ | Seam FID ↓ | Seam KID ↓ | GT-referenced discrepancy ↓ |
|---|---|---|---|---|---|
| w/o SAPE (2D RoPE) | 11.30 | 0.2448 | 109.92 | 225.11 | 0.18 |
| CCM concatenation + 2D RoPE | 12.11 | 0.3412 | 91.43 | 183.24 | 0.14 |
| Single-Level SAPE | <u>16.70</u> | <u>0.6555</u> | <u>60.59</u> | <u>71.60</u> | <u>0.10</u> |
| Multi-Level SAPE | **21.35** | **0.8790** | **48.50** | **43.28** | **0.09** |

# E Visible and View-Unseen Regions

The coarse UV provides texture evidence only for surfaces observed by the projection cameras. Using their union coverage mask, we label a surface texel as *visible* if it receives evidence from at least one projection and as *view-unseen* otherwise. View-unseen regions include interior cavities and undersides that are difficult to capture from standard exterior viewpoints. We report mask-aware PSNR, SSIM, and LPIPS, omitting FID and KID because they measure image-set distributions rather than errors within a spatial mask.

Table 7 reports gains over the 2D UV RoPE baseline using identical regional masks. Channel concatenation yields only small improvements in either regime, whereas both SAPE variants perform substantially better. Their largest gains occur on view-unseen surfaces, where the model has no projected texture evidence and must rely on reference features, surface coordinates, and the learned prior.

These comparisons keep the coarse UV and all other conditioning fixed. Because SAPE's largest gains occur where the coarse UV provides no evidence, they cannot be explained by that condition; its separate contribution is measured in Table 4.

Table 7: Visible and view-unseen region breakdown, reported as gains over the 2D UV RoPE baseline under identical regional masks.

| Variant | Region | ΔPSNR ↑ | ΔSSIM ↑ | LPIPS reduction ↑ |
|---|---|---|---|---|
| CCM concatenation + 2D RoPE | Visible | +0.08 | +0.0014 | +0.01 |
| CCM concatenation + 2D RoPE | Unseen | +0.17 | +0.0151 | +0.01 |
| Single-Level SAPE | Visible | +0.14 | +0.0268 | +0.03 |
| Single-Level SAPE | Unseen | +0.85 | +0.0297 | +0.04 |
| Multi-Level SAPE | Visible | **+1.14** | **+0.0578** | **+0.05** |
| Multi-Level SAPE | Unseen | **+1.39** | **+0.0948** | **+0.05** |

## F Application to Generated Meshes

To evaluate DirectUV in the context of current 3D content creation workflows, we apply it to meshes produced by recent image-to-3D shape generators like Tripo and Rodin, forming a complete image-to-textured-mesh pipeline. Given a reference image, the shape generator produces a mesh, and DirectUV textures it conditioned on the same image. Figure 7 shows results across diverse objects, demonstrating that DirectUV handles the varying topologies and UV layouts produced by generative shape models and integrates naturally into an end-to-end production setting. These meshes use the UV parameterizations automatically exported by their platforms, without re-unwrapping or manual cleanup such as island merging, seam repair, or fragment deletion.

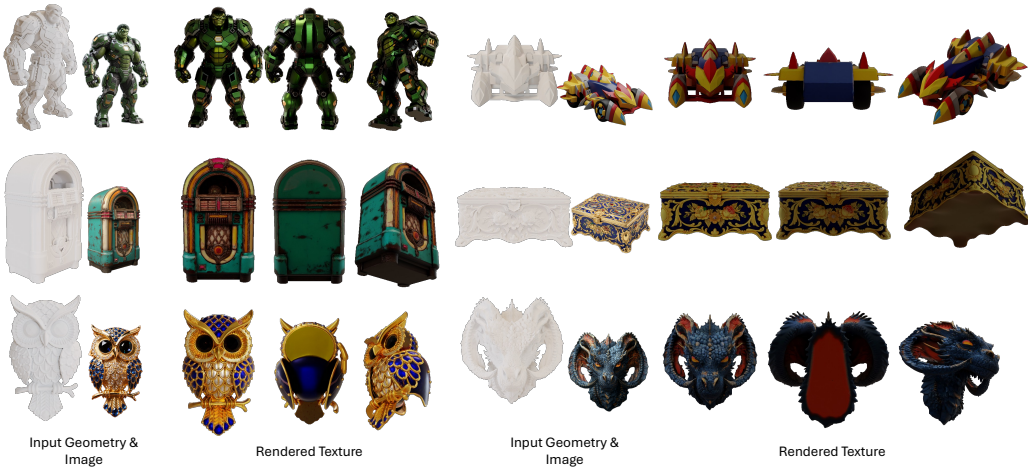


Figure 7: Texture generation on meshes produced by image-to-3D shape generators, demonstrating DirectUV in a complete image-to-textured-mesh pipeline. Each block shows the input reference image alongside the generated texture rendered from multiple viewpoints.

## G Robustness to UV Fragmentation

Island count measures how extensively a surface is partitioned into disconnected UV charts. Because GSO mainly contains meshes with relatively few islands, we also evaluate held-out TexVerse objects, extending the tested range beyond 50,000 islands.

Table 8 reports quality by island count. For GSO groups above 10 islands, PSNR remains between 25.73 and 28.27 dB; across held-out TexVerse groups spanning 10k to more than 50k islands, it remains between 26.00 and 28.57 dB. PSNR, SSIM, and LPIPS show no consistent degradation as island count increases, supporting robustness to chart-count fragmentation over the tested range. Section H presents a complementary boundary case with poor local chart geometry, including tiny slivers, corner patches, and inefficient packing. Thus, DirectUV tolerates high chart counts in the evaluated layouts, although severely ill-conditioned local parameterization can still reduce fine-detail quality.

FID and KID are set-level metrics sensitive to subgroup size and composition, so we do not compare them across groups. The GSO bin above 1,000 islands contains only six objects (48 renders), for which FID worsens while KID improves. We therefore assess the fragmentation trend using the per-image metrics.

Table 8: Texture quality as a function of UV atlas fragmentation, measured by island count. The held-out TexVerse subsets extend the tested range well beyond what GSO covers.

| Dataset | Island count | Objects | PSNR ↑ | SSIM ↑ | LPIPS ↓ | FID ↓ | KID $\times 10^4$ ↓ |
|---|---|---|---|---|---|---|---|
| GSO | 1–10 | 77 | 23.19 | 0.9411 | 0.07 | 59.76 | 50.66 |
| | 11–50 | 48 | 26.31 | 0.9597 | 0.04 | 82.36 | 28.21 |
| | 51–100 | 23 | 25.73 | 0.9562 | 0.04 | 91.75 | 31.97 |
| | 101–1000 | 46 | 26.47 | 0.9616 | 0.04 | 73.14 | 44.36 |
| | >1000 | 6 | 28.27 | 0.9637 | 0.03 | 153.78 | 8.97 |
| TexVerse (held out from training) | 10k–20k | 220 | 27.09 | 0.9675 | 0.04 | 59.46 | 19.25 |
| | 20k–50k | 134 | 26.00 | 0.9673 | 0.04 | 84.08 | 25.43 |
| | >50k | 64 | 28.57 | 0.9706 | 0.04 | 102.28 | 55.69 |

## H Failure Cases

DirectUV is sensitive to the quality of the underlying UV parameterization. On meshes with highly fragmented UV layouts—composed of many small slivers and corner patches—the per-token surface coordinate $\mathcal{S}_i$ aggregates information over relatively large surface regions for each fragment, resulting in a coarse CCM signal. In such cases, the multi-level subdivision is unable to fully recover fine-grained local detail. Figure 8 shows representative examples where the rendered textures exhibit blurriness and color leakage in these regions.

These failure modes arise from the UV parameterization and mesh quality, rather than from the SAPE design itself. We view this limitation as an opportunity for future work, such as UV-aware preprocessing or jointly learned tokenization strategies tailored to UV-space data.

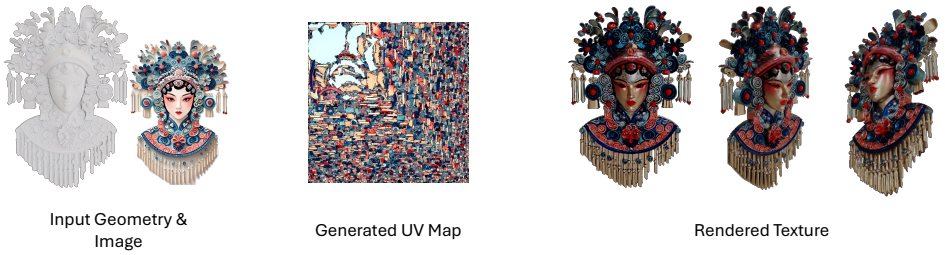


Figure 8: Failure cases on heavily fragmented UV layouts. The UV map (top row) is split into a large number of small islands and slivers with substantial unused border space, leading to blurred or color-leaked regions in the wrapped texture (bottom row).

## I Inference Runtime and Memory

We measure 28-step denoising at batch size 1 on a single A100-80GB GPU, using the same backbone across variants to isolate positional-encoding cost. Table 9 reports latency and peak GPU memory. Single-Level SAPE has no measurable overhead over standard 2D UV RoPE, while Multi-Level SAPE adds only a small latency increase and no additional memory cost. SAPE has no learnable parameters and only changes the coordinates passed to the existing rotary embedding.

Texture resolution determines the number of UV tokens and dominates the cost, whereas mesh resolution mainly affects the one-time rasterization used to construct the CCM and coarse UV. The reported numbers cover denoising only; the multi-view generation stage is an off-the-shelf component shared with projection-based pipelines.

Table 9: Denoising time and peak GPU memory under an identical backbone. Each cell reports latency / peak memory.

| Resolution | 2D UV RoPE | Single-Level SAPE | Multi-Level SAPE |
|---|---|---|---|
| $512^2$ | 13.06 s / 15.20 GiB | 13.05 s / 15.20 GiB | 13.38 s / 15.20 GiB |
| $1024^2$ | 35.99 s / 18.04 GiB | 35.98 s / 18.04 GiB | 36.40 s / 18.04 GiB |

# J More Results

We present additional texture generation results in Figure 9, covering a diverse range of object categories and mesh complexities.




Figure 9: Additional texture generation results.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately state the three contributions (latent UV diffusion, SAPE, multi-level SAPE), which are directly supported by Sections 3 and 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in the conclusion and in the appendix, covering sensitivity to UV parameterization quality and failure cases on fragmented UV layouts.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper proposes a method without formal theorems or proofs; contributions are validated empirically.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4.2 provides full architecture and training details, including model configuration, optimizer settings, and inference procedure.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code and pretrained model weights are being prepared for public release at `https://ltt-o.github.io/DirectUV.github.io/`

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Training hyperparameters, data construction, and evaluation protocol are fully described in Sections 4.2 and 4.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Error bars are not reported; evaluation metrics are computed over 200 randomly sampled GSO objects, which provides sufficient coverage to support the reported comparisons.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4.2 reports the GPU type (8× A800), batch size, and step count.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research involves 3D texture generation and raises no ethical concerns.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This work focuses on 3D texture generation for creative applications and does not have direct negative societal impacts beyond those common to generative models in general.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The proposed model generates object textures and poses no significant risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and pretrained models used are properly cited; licenses were verified to permit research use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new datasets or code assets are released with this submission.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human subjects; evaluation relies solely on automated metrics.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects and requires no IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used as part of the core method; any LLM assistance was limited to writing and editing.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.