INK3D: SCULPTING 3D ASSETS WITH EXTREMELY COMPLEX TEXTURES VIA VIDEO GENERATIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent 3D generative models can synthesize high-quality geometry but often struggle to reproduce intricate textures from reference images, largely due to the scarcity of large-scale 3D training data with rich surface appearance. In contrast, visual generative models are trained on datasets several orders of magnitude larger and excel at modeling complex visual patterns. Motivated by this gap, we introduce Ink3D, a framework that bridges 3D generation with large-scale video generative models to synthesize *extremely complex textures*. Ink3D first reconstructs a white-mesh geometry using an off-the-shelf 3D generation model. It then employs OrbitPainter, a conditional video generative model, to produce dense orbit-scan videos capturing object appearance across viewpoints. To convert these views into coherent textures, we introduce TextureOptimizer, a neural baking module that integrates dense multi-view observations while mitigating geometry inconsistencies arising from video generation. By decoupling geometry and texture synthesis and leveraging large-scale pretrained video priors, Ink3D enables significantly richer and more faithful texture generation than prior approaches.



Figure 1: *Ink3D* introduces geometry-aware video generation into the 3D generation pipeline to synthesize 3D assets with extremely complex textures such as intricate clothing patterns and high-frequency decorative details.

1 INTRODUCTION

Recent 3D generative models (Xiang et al., 2025b;a; Zhao et al., 2025; Hunyuan3D et al., 2025) commonly use variational autoencoders to encode assets into compact latent spaces, where diffusion models learn their distribution. This paradigm has improved geometric fidelity, producing shapes with finer structural details (Chen et al., 2025; Wu et al., 2025b; He et al., 2025).

However, despite impressive geometric quality, existing 3D generation models often struggle to synthesize *extremely complex textures*. Typical failure cases include text-rich surfaces (e.g., logos, signage, printed words), intricate clothing patterns (e.g., plaid fabrics, embroidery, layered motifs),

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.

or high-frequency decorative details. The underlying reason is straightforward: high-quality 3D training data are scarce and expensive to construct. In contrast, large-scale image and video data on the Internet exceed 3D datasets by several orders of magnitude, often reaching billions of samples. Consequently, image and video generative models (Wan et al., 2025; Wu et al., 2025a; Cai et al., 2025) pretrained on such massive corpora demonstrate superior visual modeling capability and remarkable generalization, especially for fine-grained appearance and texture-rich content.

These observations inspire us to *decouple geometry and texture modeling*. Instead of forcing a single 3D generative model to learn both structure and appearance from limited 3D data, we first generate a high-quality *white-mesh* geometry using an existing 3D generation model (Xiang et al., 2025a), and then leverage powerful visual generative models (Wan et al., 2025) to synthesize texture information. Finally, the generated texture is refined and baked onto the underlying geometry, producing a fully textured 3D asset.

In this work, we introduce **Ink3D**, a framework that integrates 3D generation with large-scale video generative models. Ink3D can synthesize extremely complex textures, as illustrated in Figure 1. Given a reference image, Ink3D first employs an off-the-shelf 3D generation model (Xiang et al., 2025a) to reconstruct the corresponding white-mesh geometry. Texture synthesis is then performed through a dedicated two-stage pipeline:

OrbitPainter. OrbitPainter is a conditional video generative model designed to produce dense multi-view observations of the object. Conditioned on the reference image, it generates two *360-degree orbit-scan videos*: one with horizontal camera rotation and one with vertical rotation. Together, these two trajectories ensure comprehensive viewpoint coverage of the object surface. Benefiting from large-scale pretraining on massive video data, the model possesses strong visual generalization capability, requiring only lightweight fine-tuning to adapt to arbitrary object categories.

TextureOptimizer. The generated orbit-scan videos provide a dense set of multi-view images that can be used for texture baking. A straightforward approach is to directly consolidate these views and bake them onto the geometry. In practice, however, this naive strategy often produces noticeable artifacts, such as color bleeding and blurred textures. The main reason is that the generated views may exhibit geometry inconsistencies introduced during the video generation process. To address this issue, we introduce *TextureOptimizer*, a neural baking framework that robustly integrates the dense-view observations while enforcing geometric consistency. Specifically, TextureOptimizer first converts the dense-view image set into a textured voxel representation guided by the underlying geometry, and then refines the voxelized texture using a prior-guided denoising process. This strategy effectively leverages the rich appearance information from multi-view observations while mitigating the geometry inconsistencies introduced during video generation.

Decomposing 3D generation into geometry-first and texture-second has been explored in prior works. Most existing methods (Richardson et al., 2023; Liu et al., 2024a; Tang et al., 2023; Zeng et al., 2024; Liu et al., 2024b; Youwang et al., 2024; Zhao et al., 2025; Hunyuan3D et al., 2025; Li et al., 2025; Zhang et al., 2024) rely on multi-view image generation, typically synthesizing six discrete views conditioned on a reference image and a given geometry. However, such sparse-view strategies suffer from two major limitations. First, due to incomplete view coverage, certain surface regions may remain untextured, leading to missing or blurred areas. Second, independently generated views often exhibit cross-view inconsistencies, where the same surface region appears with different colors or patterns across views. In contrast, our use of video generative models effectively alleviates both issues. The generated 360-degree orbit videos provide dense viewpoint coverage, ensuring full surface observation. Moreover, the intrinsic spatiotemporal consistency of video generation enforces coherent appearance across adjacent frames, naturally translating into strong cross-view consistency in 3D space.

2 Related Work

3D Generation. Large-scale datasets (Deitke et al., 2023b;a; Zhang et al., 2025) have enabled substantial progress in 3D asset generation. Existing systems either jointly synthesize geometry and appearance, as in Trellis (Xiang et al., 2025b;a), or generate geometry before texturing, as in Hunyuan3D (Zhao et al., 2025; Hunyuan3D et al., 2025), Clay (Zhang et al., 2024), and Step1X-3D (Li et al., 2025). Despite advances in geometric fidelity, intricate textures remain challenging

because high-quality 3D appearance data are scarce. Ink3D follows the geometry-first paradigm and draws on large-scale video generative priors for texture synthesis.

Geometry Texturing. Image diffusion priors have been applied through score-distillation optimization (Poole et al., 2023; Lin et al., 2023; Wang et al., 2023; Chen et al., 2023b) and UV-based texturing pipelines (Chen et al., 2023a; Yu et al., 2024). Another common strategy generates multiple views and projects them onto the surface (Richardson et al., 2023; Liu et al., 2024a; Tang et al., 2023; Zeng et al., 2024; Liu et al., 2024b; Youwang et al., 2024). Sparse views can leave surface regions uncovered (Liang et al., 2025), while inconsistent predictions across views can introduce seams after projection. SeqTex (Yuan et al., 2025) also uses a video generative model, but generates a sparse set of views. Ink3D instead generates dense horizontal and vertical orbit-scan videos to improve surface coverage, with geometry-aware conditioning and attention to encourage cross-view consistency. Its TextureOptimizer further addresses residual geometric inconsistencies when fusing the generated views onto the mesh.

3 INK3D

Problem Formulation. We adopt a geometry-first, texture-second paradigm for 3D asset generation. Given a reference image $\boldsymbol{I}$, we first employ an off-the-shelf 3D reconstruction model (Xiang et al., 2025a) to estimate a white-mesh geometry $\mathcal{G}$, which captures the object's structural shape without texture. The objective of Ink3D is to generate a high-quality texture consistent with $\boldsymbol{I}$ and map it onto $\mathcal{G}$, resulting in a fully textured 3D asset.

Overview. Ink3D consists of two key components:

1. *OrbitPainter* (Section 3.1). OrbitPainter (Figure 2) is a conditional video generative model that incorporates a newly proposed Geometry-Aware Sparse Attention module (Section 3.2). Given the reference image $\boldsymbol{I}$ and the white-mesh geometry $\mathcal{G}$, it synthesizes two orbit-scan videos of the object: a horizontal orbit video $\boldsymbol{V}_H$ and a vertical orbit video $\boldsymbol{V}_V$. These videos ensure comprehensive viewpoint coverage while maintaining appearance consistency with $\boldsymbol{I}$, and geometry consistency with $\mathcal{G}$.
2. *TextureOptimizer* (Section 3.3). TextureOptimizer (Figure 3) consolidates frames from the generated orbit videos to bake textures onto $\mathcal{G}$. It explicitly mitigates geometric inconsistencies arising from imperfect video generation and resolves the challenges of dense multi-view texture fusion, resulting in a coherent and high-fidelity texture map.

3.1 ORBITPAINTER

Training Data Pre-Processing. As illustrated in Figure 2(a), we render a reference image $\boldsymbol{I}$ from each training asset using a randomly sampled viewpoint and random lighting. We then define two camera orbit trajectories: a horizontal scan and a vertical scan.

Along each trajectory, we render an RGB video, a surface-normal video, and a 3D-position video, denoted by $\boldsymbol{V}_d^{RGB}$, $\boldsymbol{V}_d^{Normal}$, and $\boldsymbol{V}_d^{Pos}$, respectively, where $d \in \{H, V\}$ indicates the scan direction. Each video contains 61 frames at 512×512 resolution by default. The two scans provide complementary viewpoint coverage and multi-modal geometric supervision for training.

Latent Extraction. Since we use a latent video diffusion model, we first encode all rendered videos into latent representations. We employ the Wan 2.2 (Wan et al., 2025) video VAE encoder, which downsamples the spatial and temporal dimensions by factors of 16 and 4, respectively. The resulting RGB, surface-normal, and 3D-position latents are denoted by $\boldsymbol{F}_d^{RGB}$, $\boldsymbol{F}_d^{Normal}$, and $\boldsymbol{F}_d^{Pos}$, where $d \in \{H, V\}$. The same encoder maps the reference image $\boldsymbol{I}$ to its latent representation $\boldsymbol{F}_I$.

Architecture. Figure 2(b) illustrates the architecture of OrbitPainter, a conditional video generative model. The model is conditioned on two types of priors:

1. A 2D appearance prior $\boldsymbol{F}_I$, extracted from the reference image.
2. 3D geometric priors $\{\boldsymbol{F}_H^{Normal}, \boldsymbol{F}_H^{Pos}, \boldsymbol{F}_V^{Normal}, \boldsymbol{F}_V^{Pos}\}$, which encode surface orientation and spatial structure under horizontal and vertical scans.

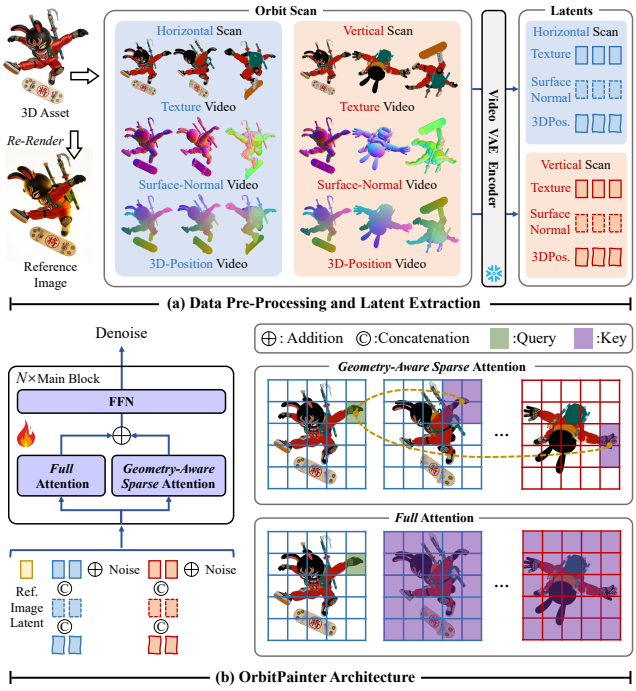


Figure 2: Overview of *OrbitPainter*. (a) Given a 3D asset, we first apply random lighting and render it from a randomly sampled viewpoint to obtain a reference image. We then perform horizontal and vertical orbit scans to render RGB videos, surface-normal videos, and 3D-position videos. All videos, together with the reference image, are encoded by a video VAE encoder into latent representations. (b) OrbitPainter is a conditional video generative model that synthesizes RGB orbit-scan videos conditioned on a 2D appearance prior (the reference image) and 3D geometry priors (surface-normal and 3D-position videos). A Geometry-Aware Sparse Attention module is introduced in parallel with full attention to better leverage the 3D geometry priors.

Given these conditional inputs, the training objective is to generate the RGB latent videos $\{\boldsymbol{F}_H^{RGB}, \boldsymbol{F}_V^{RGB}\}$ using a Flow Matching objective. The generated latents represent multi-view appearance in video form while remaining consistent with the underlying 3D geometry.

Flow Construction. We first concatenate the target RGB latent videos temporally:

$$\boldsymbol{F}^{RGB} = \text{Concat}_T(\boldsymbol{F}_H^{RGB}, \boldsymbol{F}_V^{RGB}), \tag{1}$$

where $\text{Concat}_T(\cdot)$ denotes temporal concatenation. Following the Flow Matching formulation, we sample $t \in [0, 1]$ from a logit-normal distribution and initialize Gaussian noise $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The intermediate sample is obtained via linear interpolation:

$$\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\boldsymbol{F}^{RGB}. \tag{2}$$

Injecting Geometric and Appearance Priors. To guide generation with geometry information, we first concatenate the horizontal and vertical geometric latents along the temporal dimension:

$$\boldsymbol{F}^{Normal} = \text{Concat}_T(\boldsymbol{F}_H^{Normal}, \boldsymbol{F}_V^{Normal}), \tag{3}$$

$$\boldsymbol{F}^{Pos} = \text{Concat}_T(\boldsymbol{F}_H^{Pos}, \boldsymbol{F}_V^{Pos}). \tag{4}$$

We concatenate these geometric priors with the noisy RGB latent along the channel dimension. We then prepend the reference-image latent temporally to incorporate the appearance prior:

$$\mathbf{x}'_t = \text{Concat}_C(\mathbf{x}_t, \boldsymbol{F}^{Pos}, \boldsymbol{F}^{Normal}), \tag{5}$$

$$\boldsymbol{S} = \text{Concat}_T(\boldsymbol{F}_I, \mathbf{x}'_t). \tag{6}$$

Here, Concat_C denotes channel-wise concatenation. The resulting sequence $\boldsymbol{S}$ contains the noisy RGB latents together with the appearance and geometric priors.

Network Architecture. The sequence $\boldsymbol{S}$ is fed into the OrbitPainter network. As shown in Figure 2(b), OrbitPainter consists of N Transformer blocks. Each block contains two parallel attention modules: a standard full-attention module and a proposed *Geometry-Aware Sparse Attention* module (introduced in Section 3.2). The full-attention module processes the entire sequence $\boldsymbol{S} = \text{Concat}_T(\boldsymbol{F}_I, \mathbf{x}'_t)$, while the sparse attention module operates only on the video-related tokens $\mathbf{x}'_t$ to better capture geometry-aware correspondences.

Training Objective. We adopt Flow Matching as the training objective. The model learns to predict the velocity $\mathbf{u}_t = d\mathbf{x}_t/dt$ by minimizing the mean squared error between the predicted velocity $\mathbf{v}_t$ and the ground-truth velocity:

$$\mathcal{L} = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{F}^{RGB}} \left\| \mathbf{v}_t - \mathbf{u}_t \right\|^2. \tag{7}$$

3.2 Geometry-Aware Sparse Attention

We formulate 3D texture generation as a video generation problem, where the goal is to synthesize horizontal and vertical orbit-scan videos conditioned on a reference image prior and 3D geometry priors. This setting differs from conventional video generation tasks, which typically rely only on appearance priors (e.g., a reference image) without explicit geometric information. 3D geometry priors can therefore guide the generation process more effectively.

In standard video diffusion models, the attention module performs full spatiotemporal attention. As illustrated in Figure 2(b), each query token attends to all tokens across both spatial and temporal dimensions. This design is necessary because traditional video generation lacks structural priors that could constrain the attention space.

In our setting, the video frames are generated from orbit scans of a 3D asset, and explicit geometric priors are available. This enables us to leverage the underlying 3D geometry to guide the attention mechanism. Specifically, each spatiotemporal token in the latent video corresponds to a surface point on the 3D mesh. Given a query token, we can recover its corresponding 3D location through the position-map representation. Based on this 3D correspondence, we restrict the attention to tokens that correspond to nearby or identical surface locations across different frames, rather than attending to all tokens in the video, as illustrated in Figure 2(b).

We refer to this module as *Geometry-Aware Sparse Attention*. In practice, given a query spatiotemporal token $\mathcal{T}$, we first re-project it onto the underlying 3D geometry using its associated 3D position, thereby identifying the corresponding voxel location (i, j, k) in the 3D space. We then define a local neighborhood around (i, j, k) with an expansion radius r (set to 3 by default). For tokens in other video frames, we select those whose associated 3D positions fall within this neighborhood. Only these tokens are used as keys in the attention computation for the query token $\mathcal{T}$, while all other tokens are ignored.

3.3 TextureOptimizer

Once OrbitPainter is trained, we use it to generate dense multi-view texture observations for a given 3D geometry. Specifically, given a white-mesh geometry produced by an off-the-shelf geometry estimation model and a reference image that depicts the object's rough appearance, we first render surface-normal and 3D-position videos of the white mesh using the same orbit-scan trajectories described in Section 3.1.

These videos, together with the reference image, are encoded into latent representations using the video VAE encoder. The resulting latents, along with randomly sampled noise, are then fed into the OrbitPainter model. Through the iterative denoising process, OrbitPainter predicts texture latent videos, which are subsequently decoded by the video VAE decoder to produce orbit-scan texture videos for both horizontal and vertical scans, as illustrated in Figure 3(a).

The generated orbit-scan texture videos provide a dense set of multi-view observations, where each frame corresponds to a single viewpoint. We denote this set as $\mathcal{O} = \{\boldsymbol{O}_k\}_{k=1}^{K}$, where K is the total number of views. Since each orbit scan produces 61 frames and we perform both horizontal and vertical scans, the resulting set contains $K = 122$ views in total.

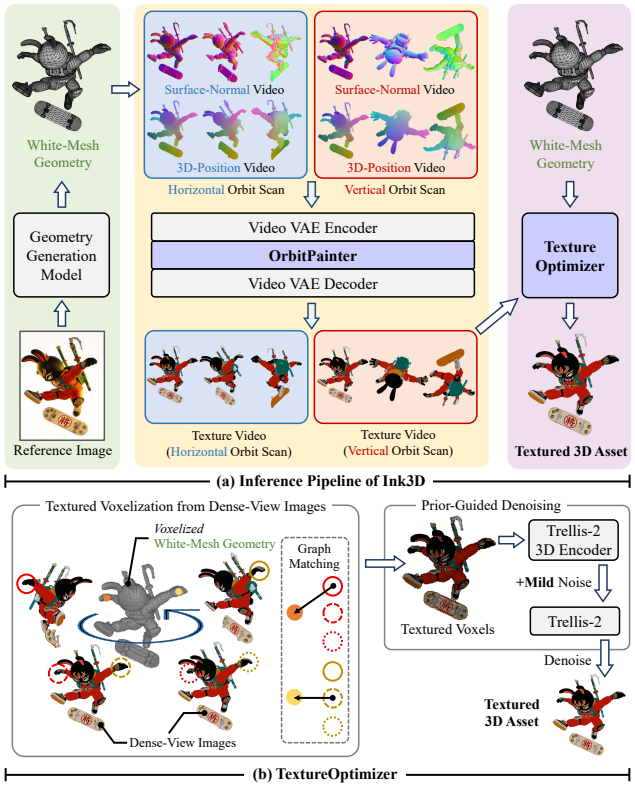


Figure 3: (a) **Inference pipeline of Ink3D.** Given a reference image, we first estimate the corresponding white-mesh geometry using an off-the-shelf geometry generation model. The trained OrbitPainter then generates orbit-scan videos of the object, whose frames are collected to form a dense-view image set. Finally, TextureOptimizer bakes these dense views onto the white-mesh geometry to produce the final textured 3D asset. (b) **TextureOptimizer.** We first convert the dense-view images into textured voxels based on the white-mesh geometry by formulating voxel–pixel correspondence as a graph-matching problem. The resulting textured voxels are then refined via prior-guided denoising: mild noise is added and Trellis-2 performs a few denoising steps to produce the final textured 3D asset.

Can Simple Baking Work? A straightforward solution is to directly use the dense view set $\mathcal{O}$ to bake textures onto the white-mesh geometry. However, in practice this naive approach produces noticeable artifacts on the resulting textured mesh, such as color bleeding and blurry textures. The root cause is that frames in the generated orbit-scan videos are not strictly geometry-consistent. Although the generated videos visually resemble an orbit scan around a 3D object, the same surface region may appear slightly inconsistent across different frames. Consequently, when these frames are reprojected back onto the mesh using the corresponding camera parameters, pixels that should correspond to the same 3D surface location may not align perfectly. This misalignment leads to inconsistent color accumulation during baking, resulting in texture artifacts.

Motivation. The above observations inspire us to design a more robust baking mechanism that can mitigate the geometry inconsistency across dense views. We thus introduce *TextureOptimizer*, a neural baking approach that leverages the capabilities of the native 3D generation model Trellis-2 (Xiang et al., 2025a) in a training-free manner. The original Trellis-2 model generates a textured 3D asset from a geometry initialized with Gaussian noise (referred to as *noisy geometry*), effectively transforming a fully noisy geometry into a textured 3D asset. However, while effective, Trellis-2 alone struggles to reproduce extremely complex textures. As illustrated in Figure 3(b), the key idea of TextureOptimizer is simple: instead of starting from a fully noisy geometry, we initialize the geometry with *mild* noise while incorporating as much texture information as possible from the dense-view image set $\mathcal{O}$. This initialization guides the model during the denoising process

toward generating a textured 3D asset that faithfully follows the appearance observed in $\mathcal{O}$, while significantly reducing baking artifacts.

Textured Voxelization from Dense-View Images. We first convert the dense-view image set $\mathcal{O}$ into a *textured voxel representation* based on the white-mesh geometry $\mathcal{G}$ to satisfy the input requirements of Trellis-2.

Concretely, given a set of occupied surface voxels $\mathcal{V} = \{\boldsymbol{v}_m\}_{m=1}^{M}$ containing M voxels (obtained by voxelizing the white-mesh geometry $\mathcal{G}$), and a dense-view image set $\mathcal{O} = \{\boldsymbol{O}_k \in \mathbb{R}^{H\times W}\}_{k=1}^{K}$ consisting of K multi-view images, each with spatial resolution (H, W), our goal is to establish a globally optimal correspondence between the M voxels and the $H \times W \times K$ image pixels. In other words, each voxel is assigned the color of one pixel selected from the dense-view image set. This correspondence should minimize texture seams while respecting geometric visibility constraints.

To this end, we formulate the problem as a *graph-matching* task and optimize the following objective using a Markov Random Field (MRF) over the 6-connected voxel graph $\mathcal{E}$:

$$\min_{\{k_m\}_{m=1}^{M}} \sum_{m=1}^{M} D_m(k_m) + \lambda \sum_{(m,n)\in\mathcal{E}} w_{m,n}\mathbf{1}[k_m \neq k_n], \tag{8}$$

where $k_m \in \{1, \ldots, K\}$ is the assigned view index for voxel $\boldsymbol{v}_m$. By determining k_m, voxel $\boldsymbol{v}_m$ is explicitly assigned the pixel value $\boldsymbol{O}_{k_m}(i_m, j_m)$, where (i_m, j_m) represents the 2D projected coordinates of the voxel in the k_m-th view. The data term (the first term) $D_m(k_m)$ evaluates the geometric visibility and projection reliability of this specific pixel, which is measured based on the angle between the voxel's surface normal and the camera's viewing direction. The smoothness term (the second term) utilizes color-gradient-adaptive weights $w_{m,n}$ to penalize texture seams, which occur when adjacent voxels draw colors from different views (*i.e.*, $k_m \neq k_n$). Finally, we optimize Eq. (8) via alpha-expansion (Boykov et al., 2001) and fill any unobserved voxels via breadth-first color diffusion. Optimization yields a textured voxel representation of the geometry, $\mathcal{V}'$.

Prior-Guided Denoising. The original Trellis-2 model takes untextured voxels corrupted with randomly sampled Gaussian noise as input and gradually denoises them to produce textured geometry. During denoising, the intermediate representations gradually accumulate texture information.

In our setting, however, we already obtain textured voxels $\mathcal{V}'$ that contain substantial texture information. This observation motivates us to leverage Trellis-2 to perform a *prior-guided denoising* process. Specifically, as illustrated in Figure 3(b). we first use the Trellis-2 3D VAE encoder to extract latent representations from $\mathcal{V}'$. We then perturb the latents with *mild* Gaussian noise and feed the resulting mildly noisy latents into Trellis-2. Since the initialization is only mildly corrupted, the model performs only a few denoising steps (corresponding to the later stages of the original denoising trajectory) to generate the final textured geometry.

4 Experiment

Implementation Details. *OrbitPainter* is trained on the Sketchfab subset of Objaverse-XL (Deitke et al., 2023a), using 60K samples with albedo textures. The model is initialized from Wan-2.2-14B (Wan et al., 2025). The geometry-aware spatial attention module is initialized using the weights of the corresponding full-attention layers. During training, we adopt LoRA-style fine-tuning with a rank of 32. We use AdamW (Loshchilov & Hutter, 2017) with a learning rate of 1e-4. The model is trained for 45k iterations with a batch size of 1 on 8× NVIDIA A100 GPUs. Unless otherwise specified, we generate two orbit-scan videos for each object: one with horizontal camera rotation and one with vertical rotation. Each video has a resolution of 512×512 and contains 61 frames. Additional implementation details are provided in the appendix.

Evaluation Data. We select 85 high-quality samples from the Texverse (Zhang et al., 2025) dataset. As our focus is on generating highly complex textures, the original assets in Texverse do not sufficiently meet this requirement. Therefore, for each sample we retain only its geometry and discard the original texture. We then render the frontal-view projection of the white mesh and feed it into the commercial image generation model Nano Banana to synthesize a reference image with complex textures. As a result, for each of the 85 samples we obtain a pair consisting of a white-mesh geometry and a corresponding reference image that exhibits rich and complex textures.

Evaluation Metrics. For all 85 samples, all methods receive the same ground-truth geometry and reference texture image to generate textured 3D assets. We evaluate front-view renderings against references using *FID*, *CLIP-FID*, *LPIPS*, and *CLIP-I*. FID measures the Fréchet distance between Inception feature distributions to assess visual quality and realism. CLIP-FID uses CLIP features to capture semantic similarity. LPIPS measures perceptual differences using deep features, while CLIP-I measures semantic consistency through the cosine similarity of CLIP image embeddings.

Table 1: Quantitative comparison of texture synthesis quality. Our primary competitor is Trellis-2, a recently released state-of-the-art 3D generation model. Ink3D significantly outperforms Trellis-2 and other existing open-source approaches in texture synthesis.

Method	FID↓	CLIP-FID↓	LPIPS↓	CLIP-I↑
Paint3D (Zeng et al., 2024)	146.0	570.6	0.3417	0.7980
TEXGEN (Yu et al., 2024)	123.7	291.8	0.2280	0.8409
SeqTex (Yuan et al., 2025)	148.3	387.9	0.2601	0.8052
Trellis-2 (Xiang et al., 2025a)	120.4	223.9	0.2714	0.8694
Ink3D (Ours)	103.7	195.6	0.2029	0.8979

4.1 Main Results

Table 1 presents a quantitative comparison between Ink3D and several representative open-source approaches capable of generating textures given a white-mesh geometry. Our primary comparison is with Trellis-2, a recently released state-of-the-art 3D generation model that supports both joint geometry–texture generation and texture synthesis conditioned on a given geometry and reference image. To ensure a fair comparison, we adopt the second mode of Trellis-2, where the model is provided with the same white-mesh geometry and reference image as inputs. As shown in Table 1, Ink3D significantly outperforms Trellis-2 and other competing methods in texture synthesis quality. This improvement highlights the advantage of introducing geometry-aware video generation into the texture creation pipeline, enabling Ink3D to synthesize richer and more detailed surface appearances. Additional qualitative results are provided in the supplementary material.

4.2 Ablation Studies

Table 2: Ablation study on leveraging 3D geometry priors in OrbitPainter. OrbitPainter incorporates geometry priors to generate geometry-consistent orbit-scan videos at two levels: (1) *input level*, by directly providing a 3D-position video and a surface-normal video as additional inputs; and (2) *network level*, by introducing the proposed Geometry-Aware Sparse Attention (GASA), which restricts attention to pixels corresponding to the same 3D location across frames.

Setting	FID↓	CLIP-FID↓	LPIPS↓	CLIP-I↑
Baseline	103.7	195.6	0.2030	0.8980
w/o Surface Normal	113.9	218.3	0.2128	0.8841
w/o 3D-Position	149.2	361.5	0.2367	0.8282
w/o GASA	109.9	203.7	0.2077	0.8893

3D Geometry Prior in OrbitPainter. As introduced in Section 3.1, OrbitPainter is trained with two conditioning priors: (1) a 2D texture appearance prior provided by the reference image, and (2) a 3D geometry prior provided by the white-mesh geometry. The latter prior goes beyond traditional image-to-video generation settings. To incorporate this 3D geometry prior, we introduce two additional conditioning signals: a surface-normal video and a 3D-position video. In Table 2, we compare our default model with two ablated variants: one without the surface-normal video prior and another without the 3D-position video prior. As shown in Table 2, removing either prior degrades the performance, with the removal of the 3D-position prior causing a particularly significant drop. This is because the 3D-position signal provides strong geometric guidance, helping OrbitPainter maintain geometry-consistent appearance across orbit-scan frames.

Geometry-Aware Sparse Attention. In Section 3.2, we introduce a Geometry-Aware Sparse Attention (GASA) layer, which operates in parallel with the full attention layer, as illustrated in Figure 2(b). GASA explicitly encourages the network to learn correspondences across frames by link-

ing pixels that share the same original 3D location when re-projected into 3D space. In Table 2, we compare our default model with a variant that removes GASA. The observed performance drop demonstrates that better utilization of geometric priors improves geometry consistency across the generated orbit-scan frames, which in turn leads to higher-quality textured 3D assets. While the injected 3D-position and surface-normal videos introduce geometry priors at the input level, GASA incorporates geometry priors directly within the network attention mechanism.

Table 3: Ablation study of orbit-scan video resolution and frame count for geometry texturing.

Resolution	#Frame	FID↓	CLIP-FID↓	LPIPS↓	CLIP-I↑
384^2	61	124.2	269.4	0.2221	0.8694
512^2	61	103.7	195.6	0.2030	0.8980
512^2	41	122.8	215.3	0.1909	0.8988
512^2	25	124.8	215.2	0.1958	0.8963

Video Quality Determines Texture Quality. We study two factors that affect the quality of the orbit-scan videos generated by OrbitPainter: the video resolution and the number of frames. The first directly influences the fidelity of the final baked texture, as higher-resolution frames provide richer visual details. The second controls the effective camera motion speed. Since the generated video always corresponds to a full 360-degree orbit scan, using fewer frames results in faster camera motion. Table 3 analyzes the impact of these two factors. The results suggest that, when computational cost is not the primary concern, higher-resolution video generation leads to better final texture quality. In addition, using too few frames significantly degrades performance. This is mainly because faster camera motion introduces motion blur in the generated orbit-scan videos, which subsequently leads to blurred texture artifacts after baking.

Table 4: Comparison of TextureOptimizer with traditional methods, namely Average Baking and Differentiable Rendering, for baking dense-view image textures onto the geometry.

Baking Strategy	FID↓	CLIP-FID↓	LPIPS↓	CLIP-I↑
Average Baking	121.0	227.7	0.2498	0.8888
Differentiable Rendering	123.3	226.8	0.2746	0.8944
TextureOptimizer	103.7	195.6	0.2030	0.8980

TextureOptimizer. In Section 3.3, we describe how to bake the dense-view image set (containing all orbit-scan frames) onto the white-mesh geometry, and introduce *TextureOptimizer*, a training-free neural baking strategy. In Table 4, we compare TextureOptimizer with two baseline baking methods: (1) *Average Baking*: each pixel from every view is re-projected onto the geometry, and the final texture value at each 3D location is obtained by averaging all pixels projected to that location. (2) *Differentiable Rendering*: widely used in modern 3D generation models such as Trellis-1 (Xiang et al., 2025b), where a texture map is optimized through gradient descent so that rendering the textured 3D model matches the dense-view observations as closely as possible. Since the generated orbit-scan videos are not perfectly geometry-consistent, all approaches must handle this issue. The results in Table 4 suggest that TextureOptimizer better integrates dense-view observations while reducing artifacts caused by cross-view inconsistencies.

5 Conclusion

We introduce Ink3D, a framework that bridges 3D generation and large-scale video generative models to synthesize extremely complex textures for 3D assets. Given a reference image, Ink3D first reconstructs a white-mesh geometry using an off-the-shelf 3D generation model, then employs a conditional video model, OrbitPainter, to generate dense 360° orbit-scan videos for comprehensive multi-view observations. These observations are integrated by TextureOptimizer, a neural baking framework that consolidates multi-view appearance while mitigating artifacts caused by geometric inconsistencies. Leveraging the strong visual modeling capability of large-scale video models, Ink3D produces rich textures that are difficult for existing 3D generation methods to achieve.

AI USE STATEMENT

An AI assistant (OpenAI Codex) was used to assist with LaTeX template migration, formatting checks, and editing the Introduction, Related Work, and method descriptions for concision. As described in the experimental section, Nano Banana was used to generate complex-texture reference images for the evaluation dataset. The proposed method uses pretrained generative models as described in the main text.

REFERENCES

Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

Huanqia Cai, Sihan Cao, Ruoyi Du, Peng Gao, Aiming Hao, Steven Hoi, Zhaohui Hou, Shijie Huang, Dengyang Jiang, Yuming Jiang, et al. Z-image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025.

Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. In *2023 IEEE/CVF international conference on computer vision (ICCV)*, pp. 18512–18522. IEEE, 2023a.

Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22189–22199. IEEE, 2023b.

Rui Chen, Jianfeng Zhang, Yixun Liang, Guan Luo, Weiyu Li, Jiarui Liu, Xiu Li, Xiaoxiao Long, Jiashi Feng, and Ping Tan. Dora: Sampling and benchmarking for 3d shape variational auto-encoders. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 16251–16261, 2025.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36:35799–35813, 2023a.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, et al. Objaverse: A universe of annotated 3d objects. In *CVPR*, 2023b.

Xianglong He, Zi-Xin Zou, Chia-Hao Chen, Yuan-Chen Guo, Ding Liang, Chun Yuan, Wanli Ouyang, Yan-Pei Cao, and Yangguang Li. Sparseflex: High-resolution and arbitrary-topology 3d shape modeling. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14822–14833. IEEE, 2025.

Team Hunyuan3D, Shuhui Yang, Mingxin Yang, Yifei Feng, Xin Huang, Sheng Zhang, Zebin He, Di Luo, Haolin Liu, Yunfei Zhao, et al. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material. *arXiv preprint arXiv:2506.15442*, 2025.

Weiyu Li, Xuanyang Zhang, Zheng Sun, Di Qi, Hao Li, Wei Cheng, Weiwei Cai, Shihao Wu, Jiarui Liu, Zihao Wang, et al. Step1x-3d: Towards high-fidelity and controllable generation of textured 3d assets. *arXiv preprint arXiv:2505.07747*, 2025.

Yixun Liang, Kunming Luo, Xiao Chen, Rui Chen, Hongyu Yan, Weiyu Li, Jiarui Liu, and Ping Tan. Unitex: Universal high fidelity generative texturing for 3d shapes. *arXiv preprint arXiv:2505.23253*, 2025.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 300–309, 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 *The Twelfth International Conference on Learning Representations*, 2024a.

Yuxin Liu, Minshan Xie, Hanyuan Liu, and Tien-Tsin Wong. Text-guided texturing by synchronized multi-view diffusion. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024b.

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

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023.

Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. In *ACM SIGGRAPH 2023 conference proceedings*, pp. 1–11, 2023.

Shitao Tang, Fuyang Zhang, Jiacheng Chen, Peng Wang, and Yasutaka Furukawa. Mvdiffusion: Enabling holistic multi-view image generation with correspondence-aware diffusion. *Advances in neural information processing systems*, 36:51202–51233, 2023.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

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. *Advances in neural information processing systems*, 36:8406–8441, 2023.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025a.

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

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, et al. Native and compact structured latents for 3d generation. *arXiv preprint arXiv:2512.14692*, 2025a.

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

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and Bill Freeman. Improved distribution matching distillation for fast image synthesis. *Advances in neural information processing systems*, 37:47455–47487, 2024.

Kim Youwang, Tae-Hyun Oh, and Gerard Pons-Moll. Paint-it: Text-to-texture synthesis via deep convolutional texture map optimization and physically-based rendering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4347–4356. IEEE, 2024.

Xin Yu, Ze Yuan, Yuan-Chen Guo, Ying-Tian Liu, Jianhui Liu, Yangguang Li, Yan-Pei Cao, Ding Liang, and Xiaojuan Qi. Texgen: a generative diffusion model for mesh textures. *ACM Transactions on Graphics (TOG)*, 43(6):1–14, 2024.

Ze Yuan, Xin Yu, Yang-Tian Sun, Yuan-Chen Guo, Yan-Pei Cao, Ding Liang, and Xiaojuan Qi. Seqtex: Generate mesh textures in video sequence. *arXiv preprint arXiv:2507.04285*, 2025.

Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. Paint3d: Paint anything 3d with lighting-less texture diffusion models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4252–4262. IEEE, 2024.

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(4):1–20, 2024.

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

Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025.

We provide additional implementation details, quantitative and qualitative evaluations, and ablation studies. The supplementary material is organized as follows:

1. Implementation Details (Appendix A)
2. Additional Quantitative Evaluation (Appendix B)
3. Qualitative Comparison (Appendix C)
4. Variety Test (Appendix D)
5. Robustness to Reconstructed Geometry (Appendix E)
6. Ablations (Appendix F)
7. Inference Cost and Latency (Appendix G)
8. Limitations and Future Work (Appendix H)

A Implementation Details

Training data. We use the Sketchfab subset of Objaverse-XL (Deitke et al., 2023a). Following the filtering procedure used by TRELLIS (Xiang et al., 2025b), we apply an aesthetic score threshold of 5.5 and retain assets with PBR materials. This yields 60K training samples. We use their albedo textures to construct the training videos.

Training configuration. OrbitPainter is initialized from Wan-2.2-14B (Wan et al., 2025). We initialize the geometry-aware spatial attention module from the corresponding full-attention layers and use LoRA fine-tuning with rank 32. We train for 45K iterations on eight NVIDIA A100 GPUs using AdamW (Loshchilov & Hutter, 2017), a learning rate of 10^{-4}, and a batch size of 1.

Default inference configuration. For each object, OrbitPainter generates one horizontal and one vertical orbit-scan video. Each video contains 61 frames at 512×512 resolution. TextureOptimizer projects the generated colors into an O-voxel representation, performs GraphCut-based texture assignment, and applies prior-guided denoising with TRELLIS.2 (Xiang et al., 2025a). The default refinement uses a nominal noise level of 10% and six denoising steps. Appendix F.4 compares this setting with alternative refinement configurations.

B Additional Quantitative Evaluation

B.1 Evaluation Protocol

Evaluation sets. The main paper evaluates 85 objects with edited front-view reference images. We additionally select 87 objects with complex textures from TexVerse (Zhang et al., 2025). Together, these form a set of 172 objects for evaluating consistency between generated views. We evaluate the final baked textures on the 87 added objects, which provide textured ground-truth assets for multi-view comparison.

Consistency of generated views. We evaluate the generated views before texture baking using Consistency Error (CE). We uniformly sample eight frames from each orbit video and back-project them into UV space. We then compute the mean absolute difference in overlapping regions. Lower CE indicates better agreement between the generated views. This metric measures cross-view consistency rather than similarity to a reference image.

Quality of baked textures. For each of the 87 added objects, we render 16 views of the final textured asset: eight at 0° elevation and eight at 60°. We compare these views with the corresponding ground-truth renderings and report FID, CLIP-FID, LPIPS, and CLIP-I, averaged over the viewpoints. This evaluates the final baked asset beyond the front-view comparison in the main paper.

B.2 Comparison with Multi-View Methods

Table 5 compares Ink3D with Step1X-3D (Li et al., 2025) and HY3D-Paint (Hunyuan3D et al., 2025). Ink3D achieves the lowest CE, FID, CLIP-FID, and LPIPS among the compared methods. HY3D-Paint obtains the highest CLIP-I, with a score of 0.9099 compared with 0.9081 for Ink3D. These results distinguish consistency of the generated views from the quality of the final baked textures.

Table 5: Additional quantitative comparison. CE evaluates generated views on all 172 objects before baking. The other metrics evaluate 16-view renderings of the final textured assets on the 87 added objects. Best values are bold.

	Generated views	Baked textures			
Method	CE↓	FID↓	CLIP-FID↓	LPIPS↓	CLIP-I↑
Step1X-3D	0.2253	184.2	309.4	0.1572	0.9005
HY3D-Paint	0.0777	153.9	267.8	0.1304	**0.9099**
Ink3D	**0.0709**	**144.1**	**258.2**	**0.1266**	0.9081

C Qualitative Comparison

Figures 5 and 6 compare Ink3D with TRELLIS.2 (Xiang et al., 2025a), HY2.1 (Hunyuan3D et al., 2025), SeqTex (Yuan et al., 2025), TEXGEN (Yu et al., 2024), and Paint3D (Zeng et al., 2024). The examples include flat surfaces, high-frequency patterns, and text. Ink3D preserves fine texture details in these cases by combining dense orbit-scan videos with texture optimization in native 3D space. The reconstructed-geometry study in Appendix E further examines how the input geometry affects the results.

C.1 Real-World Reference Images

We also test Ink3D with product photographs collected from shopping websites. Figure 4 shows the reference images and generated textured assets arranged in a scene. These examples extend the qualitative evaluation beyond the edited references used in the main benchmark. They demonstrate results on real-world inputs but do not constitute a separate quantitative benchmark.



Figure 4: Real-world reference images and textured assets generated by Ink3D. The product photographs are shown around a scene containing the generated assets.



Figure 5: **Qualitative Comparison.**



Figure 6: **Qualitative Comparison.**

D VARIETY TEST

Fig. 7 presents additional generation results. Leveraging the powerful capabilities of WAN (Wan et al., 2025), Ink3D demonstrates strong generalization across a diverse range of styles. Furthermore, thanks to its 3D-space baking strategy, Ink3D effectively preserves complex texture details, achieving high fidelity even for text, which is particularly sensitive to distortion.



Figure 7: **Variety Test.**

E Robustness to Reconstructed Geometry

The main benchmark provides ground-truth geometry. To assess sensitivity to imperfect geometry, we also run Ink3D with geometry reconstructed by TRELLIS.2. Table 6 reports the available multi-view baking metrics. Replacing ground-truth geometry with reconstructed geometry increases FID from 144.1 to 149.4 and CLIP-FID from 258.2 to 265.5. CLIP-I changes only slightly, from 0.9081 to 0.9080. Thus, the reported setting shows a modest loss in rendering quality, rather than complete invariance to geometry errors.

Figure 8 compares Ink3D results using geometry from TRELLIS.2 and Step1X-3D. All textures in this figure are generated by Ink3D; the labels identify the geometry providers. In these examples, the TRELLIS.2 geometries lead to better texture results. This indicates that the quality of reconstructed geometry remains relevant to texture generation.

Table 6: Ink3D with reconstructed or ground-truth geometry. All rows use Ink3D for texture generation and follow the 87-object, 16-view baking evaluation protocol in Appendix B.

Input geometry	FID↓	CLIP-FID↓	CLIP-I↑
TRELLIS.2 reconstruction	149.4	265.5	0.9080
Ground truth	144.1	258.2	0.9081

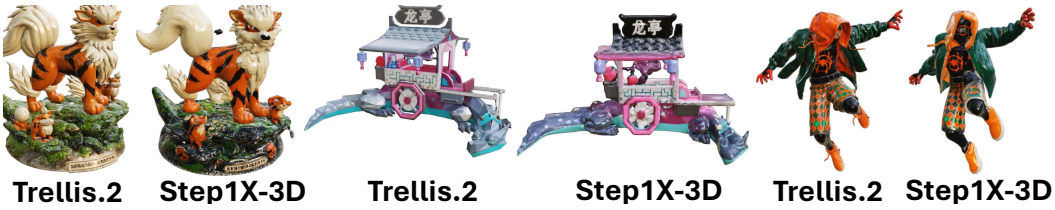


Figure 8: Ink3D results using reconstructed geometry. Each pair compares geometry supplied by TRELLIS.2 and Step1X-3D. Both members of each pair are textured by Ink3D.

F Ablations

F.1 Ablations on Video Generation

For clearer evaluation of multi-view consistency, we use a flat-geometry asset (tent) edited via nanobanana to generate a grid-textured reference image. In Fig. 9, we find that reduced resolution and fewer input views (*i.e.*, faster rotation) drastically diminish multi-view consistency. Therefore, **Ink3D aims to offer a new perspective:** while prior methods have leveraged video generation models to improve multi-view consistency, they suffer from insufficient view density. We argue that **video models trained on massive real-world data inherently possess robust 3D consistency.** Consequently, when the generation domain aligns closely with the data domain, it is possible to achieve 3D-consistent textures with high-fidelity details. Despite the largely increased computational cost, we strive to demonstrate the **upper bound** of video generation models in preserving complex texture details.

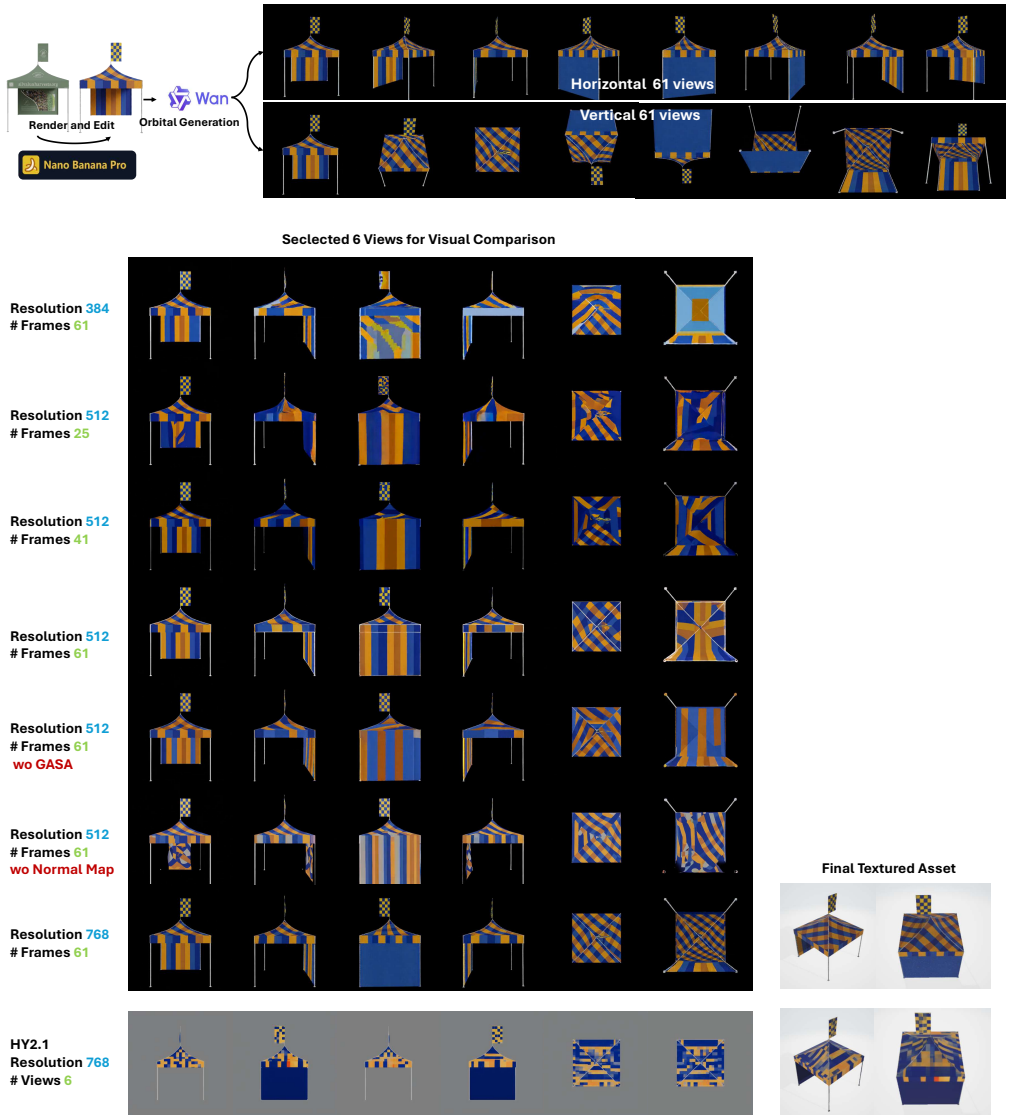


Figure 9: **Ablations on Video Generation.**

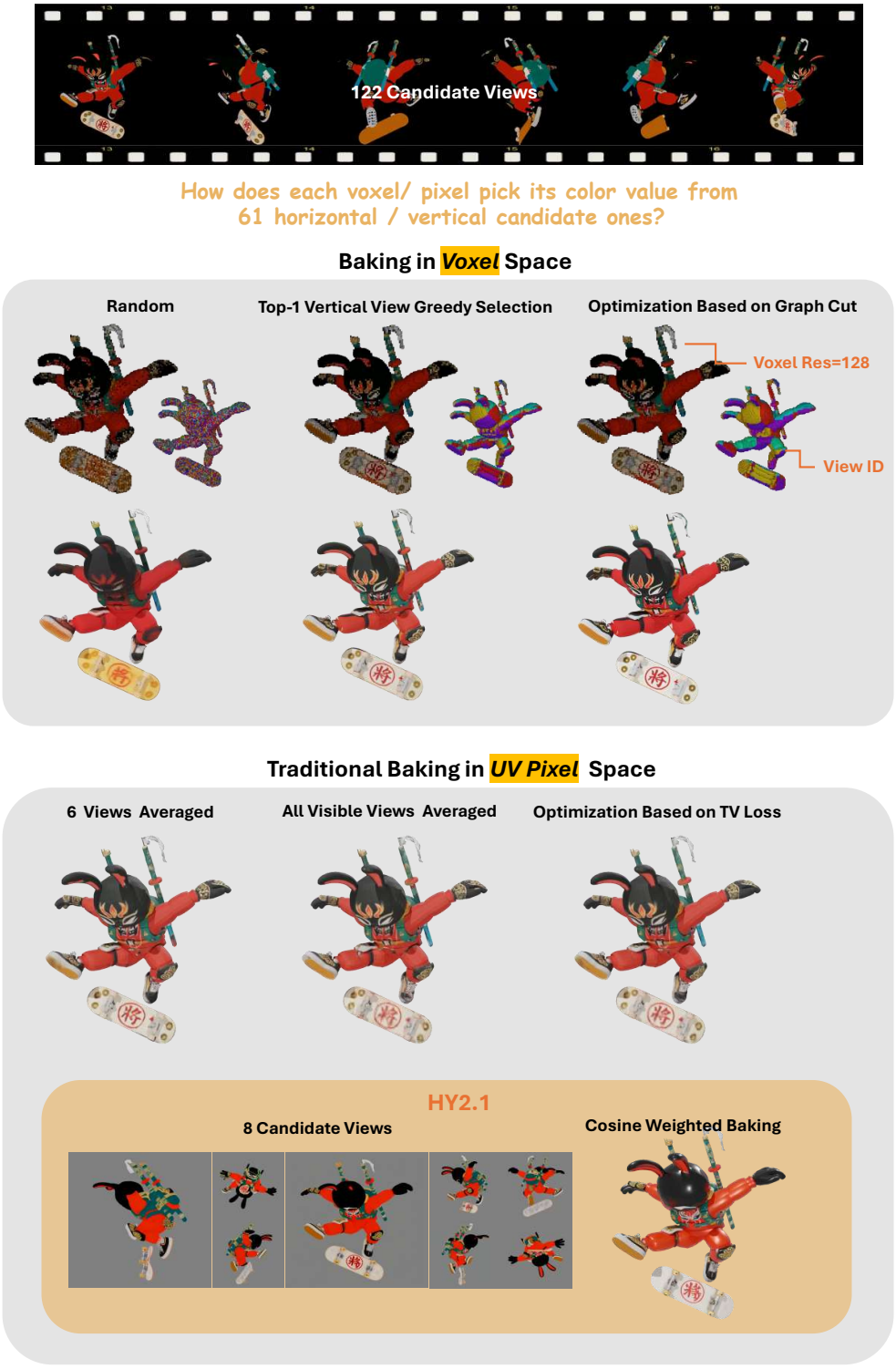


Figure 10: **Ablations on Baking.**

F.2 Ablations on Baking

Fig. 10 illustrates that while video generation provides dense, smooth, and reliable viewpoints, previous baking methods encounter two critical issues when voxels are presented with an abundance of candidate color values: ***(1)*** Texture Blurring on Flat Geometries: Due to the inherent averaging across all viewpoints, increasing the number of views leads to blurrier texture results on flat surfaces. ***(2)*** Color Bleeding on Complex Geometries: Color bleeding artifacts caused by view misalignment at complex geometric boundaries remain obvious. To address the blurring of high-frequency textures, we propose a GraphCut-based optimization algorithm that effectively mitigates this issue. Regarding color bleeding at complex boundaries, we leverage TRELLIS.2's native 3D perception of intricate geometries. Specifically, we convert the model into an O-voxel representation and project multi-view colors directly into this voxel space for baking and following refinement, thereby preserving sharp boundary details.

F.3 Orbit Trajectories

We compare the default horizontal and vertical orbit configuration (A) with two alternative trajectories (B and C) shown in Figure 11. The compared configurations use the same video duration. Table 7 reports surface coverage, trajectory length, and CE. Configuration A attains 98.2% coverage and the lowest CE of 0.0709. Configurations B and C reach 100% coverage but have longer trajectories and higher CE. Under a fixed duration, longer paths require faster camera motion. The observed results favor the default configuration as a trade-off between surface coverage and consistency.

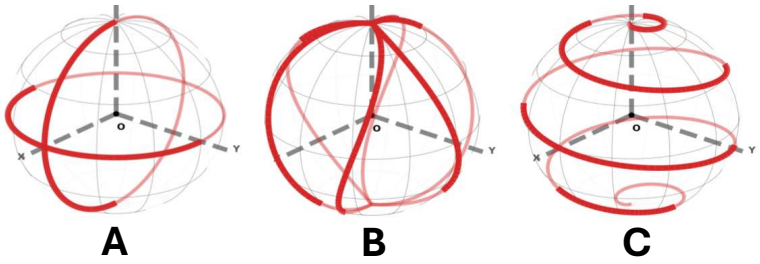


Figure 11: Orbit configurations used in the trajectory ablation. A is the default horizontal and vertical configuration; B and C are alternative camera paths.

Table 7: Trajectory ablation with the same video duration. Higher coverage alone does not yield lower consistency error.

Configuration	Coverage (%)	Trajectory length	CE↓
A (default)	98.2	18.85	**0.0709**
B	100.0	29.04	0.0814
C	100.0	24.74	0.1453

F.4 Prior-Guided Denoising

TextureOptimizer uses TRELLIS.2 as a refinement prior for an already textured asset. It does not ask TRELLIS.2 to generate the complete texture from scratch. We encode the textured O-voxel representation into VAE latents, add a small amount of noise, and apply a short denoising process.

Figure 12 compares the effect of noise level and denoising steps. The result without prior-guided refinement contains color bleeding. With the nominal noise level fixed at 10%, three denoising steps leave visible artifacts, while the default six-step setting better preserves boundaries and texture details. Raising the noise level to 30% with six steps alters the texture patterns in the shown examples. These observations motivate a mild refinement setting that corrects local artifacts while preserving the generated appearance.



Figure 12: Prior-guided denoising ablation. (a) Reference image. (b) Without TRELLIS.2 refinement. (c) 10% noise with three steps. (d) Default: 10% noise with six steps. (e) 30% noise with six steps.

G Inference Cost and Latency

On a single NVIDIA H100 GPU, OrbitPainter takes approximately 12 minutes to generate two 61-frame orbit videos at 512×512 resolution, with a peak GPU memory usage of about 45 GB. TextureOptimizer takes approximately one minute. The two texture-generation stages therefore take about 13 minutes in total, with video generation accounting for most of the latency.

Future work will explore acceleration methods such as DMD2 (Yin et al., 2024) to reduce video generation time. We will evaluate the resulting speed gains together with texture fidelity and cross-view consistency.

H Limitations and Future Work

Ink3D has several limitations. First, dense video generation is computationally expensive and dominates inference time. Second, the method remains sensitive to input geometry quality, as shown in Appendix E. Third, prior-guided denoising requires a balance between artifact removal and appearance preservation: insufficient refinement can leave color bleeding, while excessive noise can alter texture patterns. Finally, the real-world examples provide qualitative evidence, but broader evaluation is needed to characterize performance across diverse photographs and reconstruction errors.

Future work includes faster video generation, more robust refinement under imperfect geometry, and broader real-world evaluation. Another direction is to explore whether complex-texture assets produced by Ink3D can improve the training data for native 3D generative models.