DSTEX: DENSE SURFACE DECODING FOR NATIVE PBR TEXTURE GENERATION

Anonymous authors
Paper under double-blind review



Figure 1: Native PBR texturing with DSTex. Given a mesh and a reference image, our method generates detailed surface materials. Left: textured assets assembled into a lunar scene, showing fine lettering, decorative patterns, and varied material appearances under scene lighting. Right: corresponding albedo, metallic, and roughness channels (top to bottom).

ABSTRACT

Given a mesh and a reference image, native physically based rendering (PBR) texture generation synthesizes relightable surface materials directly in 3D. Latent-based frameworks typically compress these materials with an autoencoder and learn to generate the resulting latents through diffusion. Reconstructing fine lettering and intricate patterns from compact latents remains challenging, while increasing their number raises diffusion costs during training and sampling. We present DSTex, a native PBR texture generation framework that improves reconstruction by expanding the decoder's spatial feature capacity at a fixed latent count. Its autoencoder, DS-VAE, encodes material point samples into compact anchor tokens and expands them at finer locations sampled from the known mesh. Shared feature projection and relative-position modulation adapt each latent's content to these locations, constructing fine anchor tokens for continuous material queries without adding diffusion tokens. An image- and geometry-conditioned rectified-flow model generates the compact latents, which DS-VAE decodes into PBR materials on the input mesh. Quantitative and qualitative evaluations demonstrate that DSTex improves material reconstruction at matched latent counts and outperforms the evaluated open-source baselines in PBR texture generation.

1 INTRODUCTION

Image-conditioned 3D texturing generates surface appearance for a given mesh from a reference image. Recent progress in physically based rendering (PBR) texture generation extends this task to materials that can be rendered under different lighting conditions (Xiong et al., 2025; Hunyuan3D et al., 2025; Bao et al., 2026; Singh et al., 2026). Two common approaches are multi-view projection and native 3D modeling. Multi-view pipelines use pretrained image diffusion models to synthesize

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.

texture views and project them onto the mesh (Cao et al., 2023; Cheng et al., 2025; Huang et al., 2025b; Li et al., 2025a). However, inconsistent predictions across views can produce seams and texture misalignment, while occluded regions require additional completion (Liang et al., 2026). Native 3D texture generation instead predicts surface materials within a unified 3D representation, avoiding the fusion of independently generated views (Lai et al., 2026b; Zeng et al., 2026; Chen et al., 2026; He et al., 2026; Xiang et al., 2026).

Native latent generation commonly follows a two-stage pipeline: an autoencoder compresses surface materials into latents, and a diffusion model learns to generate them. The latent representation must balance the demands of detailed reconstruction and efficient diffusion modeling. Compact latents keep the diffusion sequence short, yet reconstructions from them can blur fine lettering and intricate patterns. Increasing the latent count can recover finer detail (Lai et al., 2026b;a), but longer sequences increase computation during diffusion training and at every sampling step.

Recent work explores decoder capacity in visual tokenizers (Hansen-Estruch et al., 2025; Shutkin et al., 2026), joint image decoding and upsampling through pixel diffusion (Lu et al., 2026), and token upsampling for geometry reconstruction (Hao et al., 2026). These studies motivate us to investigate how decoder design can improve reconstruction fidelity in native texture generation while retaining compact diffusion latents. The given mesh supplies finer surface locations during both autoencoding and generation, allowing the decoder to construct a dense anchor representation from encoded or generated latents.

We present DSTex (Dense Surface Texturing), a native PBR texture generation framework built on a surface-anchored VecSet representation (Zhang et al., 2023). Its DS-VAE autoencoder uses a patch-based Surface Encoder to aggregate material point samples into compact anchor tokens. The Anchor Densifier transfers each latent's content through a shared feature projection and adapts it to finer surface locations through relative-position modulation. The modulation uses each fine anchor's displacement from its parent, giving tokens derived from the same latent distinct spatial conditioning. The Dense Surface Decoder refines the resulting fine anchor tokens and supports continuous PBR material queries over the mesh surface. An image- and geometry-conditioned rectified-flow transformer, DS-DiT, generates the compact anchor tokens, which the frozen DS-VAE decodes into materials on the input mesh.

Our contributions are threefold:

- We present DSTex, an image-conditioned native PBR texturing framework that couples compact latent diffusion with dense surface decoding on input meshes.
- We introduce DS-VAE, whose Anchor Densifier constructs fine anchor tokens at mesh-sampled locations through shared feature projection and relative-position modulation, expanding spatial decoding capacity without increasing the diffusion latent count.
- Quantitative and qualitative evaluations show that densification improves reconstruction and generation at matched latent counts. DSTex also outperforms the evaluated open-source baselines in PBR texture generation quality.

2 Related Work

3D Geometry Generation. 3D representations include voxel grids (Wu et al., 2015), point sets (Achlioptas et al., 2018), meshes (Nash et al., 2020), Gaussians (Kerbl et al., 2023), and implicit fields for geometry (Park et al., 2019; Mescheder et al., 2019) or appearance (Mildenhall et al., 2020). Latent generative models use sparse voxel features (Xiang et al., 2025; He et al., 2025a; Li et al., 2025c; Xiang et al., 2026) or latent sets decoded through coordinate queries (Zhang et al., 2023; 2024; Li et al., 2025b; Chen et al., 2025b). LATTICE and UltraShape (Lai et al., 2026a; Jia et al., 2025) add explicit voxel-based anchors and support test-time token scaling while retaining continuous field decoding.

3D Texture Generation. Texture generation includes UV-space synthesis (Yu et al., 2023; 2024) and multi-view projection through sequential painting (Richardson et al., 2023; Chen et al., 2023), coordinated denoising (Cao et al., 2023; Liu et al., 2024b; Huo et al., 2024), or joint view generation (Cheng et al., 2025; Huang et al., 2025b; Zhao et al., 2025; Li et al., 2025a; Liu et al., 2026a; Feng et al., 2025). Paint3D (Zeng et al., 2024) combines multi-view initialization with UV refinement.

PBR methods use score-distillation optimization (Youwang et al., 2024; Deng et al., 2024) or learned material generation (Hunyuan3D et al., 2025; Lai et al., 2025; He et al., 2025b; Bao et al., 2026; Yeo et al., 2026; Huang et al., 2025a); MatMart (Wu et al., 2026) combines visible-surface estimation with unseen-view synthesis.

Native methods model appearance through surface functions (Oechsle et al., 2019; Siddiqui et al., 2022), explicit primitive attributes (Liu et al., 2024a; Xiong et al., 2025), or point- and grid-based latents (Lai et al., 2026b; Zeng et al., 2026; Chen et al., 2026). Field Latents (Mitchel et al., 2024) encode intrinsic texture fields in a single-mesh setting. UniTEX and Hitem3D 2.0 (Liang et al., 2026; He et al., 2026) integrate multi-view priors into 3D appearance modeling. Joint asset generation uses volumetric primitives for geometry and PBR materials (Chen et al., 2025c), while Gaussian-based methods support PBR materials (Singh et al., 2026), view-dependent radiance (Chang et al., 2026), and relighting (Li et al., 2026).

Latent Decoding and Upsampling. NTC (Vaidyanathan et al., 2023) uses per-material decoders to reconstruct compressed texture channels and mip levels. ViTok and KVAE (Hansen-Estruch et al., 2025; Shutkin et al., 2026) study decoder capacity and its effects on reconstruction and generation; DC-AE (Chen et al., 2025a) improves reconstruction at high spatial compression. INFD (Chen et al., 2024) decodes image latents through continuous neural fields, while PiD (Lu et al., 2026) combines decoding and upsampling through pixel diffusion. For geometry, MSVS-VAE (Hao et al., 2026) uses hierarchical token expansion with learned offsets and multi-scale feature aggregation. Our texture decoder instead constructs fine anchor tokens at mesh-sampled positions through relative-position modulation, increasing decoding support without adding diffusion tokens.

3 METHOD

Given a mesh $\mathcal{M}$ and a single reference image I, DSTex generates a PBR material field over its surface (Figure 2). DS-VAE's Surface Encoder aggregates material point samples into N_l surface-anchored tokens. The Anchor Densifier expands them into $N_f = fN_l$ fine anchor tokens at mesh-sampled positions, where f is the expansion factor. The Dense Surface Decoder refines and queries these tokens to recover materials at arbitrary surface points.

We first train DS-VAE on ground-truth materials, then freeze it and train DS-DiT to generate the compact N_l-token sequence conditioned on the image and geometry. The N_f fine anchor tokens are constructed only during decoding.

3.1 SURFACE ENCODER

The Surface Encoder forms a surface-anchored VecSet by aggregating point features into patches and integrating their context into anchor features before latent compression. During autoencoder training, we sample N_p points $\{(\mathbf{x}_j, \mathbf{n}_j, \mathbf{t}_j)\}_{j=1}^{N_p}$ from a textured mesh. Each sample contains a surface position $\mathbf{x}_j$, its mesh normal $\mathbf{n}_j$, and a material vector $\mathbf{t}_j \in [0,1]^{C_m}$ comprising albedo and MR (metallic and roughness) channels. Coordinates are normalized to $[-1,1]^3$, and albedo is expressed in linear color space.

For each sample, Point Embedding combines Fourier position features, spherical-harmonic normal features, and learned albedo and MR embeddings. We sort the points in Morton order for efficient parallel grouping into patches of K consecutive samples. With points indexed in this order, the first point in each group supplies an anchor coordinate,

$$\mathbf{a}_i = \mathbf{x}_{(i-1)K+1}, \qquad i = 1, \ldots, N_l, \qquad N_l = N_p/K. \tag{1}$$

The resulting anchor points lie directly on the mesh, as in native texture models (Lai et al., 2026b).

As shown in Figure 3, windowed self-attention first processes the sorted point features. Patch Merge concatenates and linearly projects each group into a patch token at coordinate $\mathbf{a}_i$, followed by inter-patch self-attention. In parallel, pointwise projections of the original point embedding at each $\mathbf{a}_i$ initialize the anchor features. We concatenate the anchor and patch sequences for joint self-attention, allowing the anchor features to integrate the patch context. We then retain the N_l anchor features and refine them with further self-attention. These attention operations use three-axis rotary position

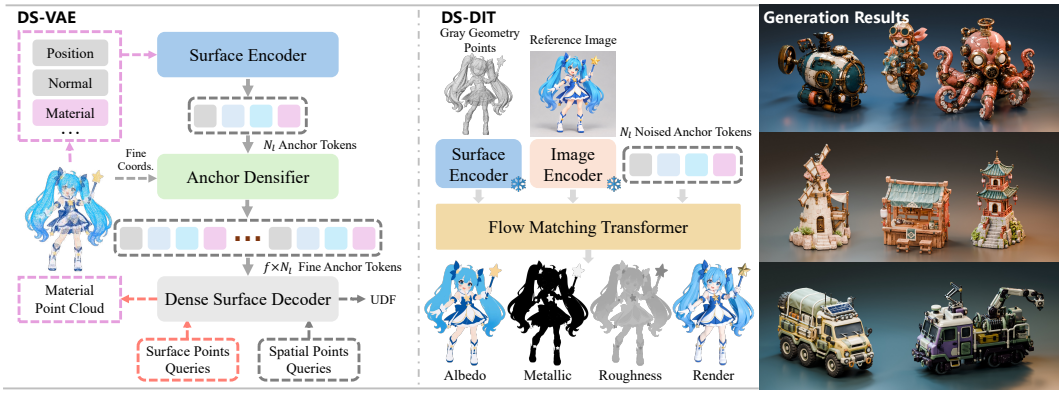


Figure 2: Overview of DSTex. DS-VAE encodes surface appearance into N_l anchor tokens and constructs $N_f = fN_l$ fine anchor tokens at mesh-sampled positions. DS-DiT generates the compact latent sequence conditioned on a single reference image and geometry. The frozen VAE decoder converts these latents into albedo and MR (metallic and roughness); the right panel shows rendered assets. Spatial queries provide auxiliary UDF supervision during VAE training.

embeddings (RoPE) (Su et al., 2024). The encoder replaces point-to-latent cross-attention (Zhang et al., 2023) with self-attention over compact anchor and patch sequences.

The bottleneck maps each encoded anchor feature $\mathbf{e}_i$ to a d-dimensional anchor token $\mathbf{z}_i$ in latent space. We apply parameter-free RMS normalization to the projected mean, as in the hyperspherical latent normalization of GAE (Liu et al., 2026b), and inject Gaussian noise at a fixed scale during autoencoder training:

$$\boldsymbol{\mu}_i = \mathrm{RMSNorm}(W_\mu \mathbf{e}_i), \qquad \mathbf{z}_i = \boldsymbol{\mu}_i + \sigma \boldsymbol{\epsilon}_i, \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, I_d), \tag{2}$$

where σ is fixed and the normalized mean lies on a hypersphere with approximately $\|\boldsymbol{\mu}_i\|_2 = \sqrt{d}$. The resulting representation pairs $Z \in \mathbb{R}^{N_l \times d}$ with mesh-derived coordinates $A = \{\mathbf{a}_i\}_{i=1}^{N_l}$, defining the anchor tokens and their positions for subsequent densification.

3.2 Anchor Densification and Surface Decoding

Given anchor tokens Z at coordinates A, the Anchor Densifier expands each token into f fine anchor tokens before decoder self-attention, yielding $N_f = fN_l$ tokens (Figure 3). We normalize and project each anchor token to $\mathbf{h}_i \in \mathbb{R}^D$, with decoder width D.

We select fine anchor positions from the same Morton-ordered points $\{\mathbf{x}_j\}_{j=1}^{N_l K}$ that define A, taking f equally strided points from each group:

$$\mathbf{b}_{ik} = \mathbf{x}_{(i-1)K+1+kK/f}, \qquad k = 0, \ldots, f-1, \tag{3}$$

where f divides K. The positions $B = \{\mathbf{b}_{ik}\}$ follow parent-major order: all f fine anchor points of anchor i precede those of anchor $i+1$. We reuse the encoder's point sample during autoencoding and the geometry-conditioning sample during generation, so $\mathbf{b}_{i0} = \mathbf{a}_i$ in both cases.

We construct fine anchor tokens by adapting each projected anchor token to sampled surface locations. We first add absolute position embeddings, then apply FiLM conditioned on the offsets from the parent anchor. Let γ be the decoder's Fourier position embedding. A shared affine projection S transforms $\mathbf{h}_i$, while a second projection P_{pos} adds the absolute position embedding at $\mathbf{b}_{ik}$:

$$\mathbf{u}_{ik} = S(\mathbf{h}_i) + P_{\mathrm{pos}}\big(\gamma(\mathbf{b}_{ik})\big). \tag{4}$$

The Offset-Modulated FiLM layer in Figure 3 then modulates $\mathbf{u}_{ik}$ using the displacement from the anchor point to the fine anchor point, $\boldsymbol{\delta}_{ik} = \mathbf{b}_{ik} - \mathbf{a}_i$ (Perez et al., 2018):

$$[\boldsymbol{\alpha}_{ik}; \boldsymbol{\beta}_{ik}] = F\big(\gamma(s\boldsymbol{\delta}_{ik})\big), \quad \mathbf{h}_{ik}^{(0)} = (\mathbf{1} + \boldsymbol{\alpha}_{ik}) \odot \mathbf{u}_{ik} + \boldsymbol{\beta}_{ik}, \quad s = \kappa\sqrt{N_f}. \tag{5}$$

Here F predicts a scale and shift for each channel, and $\odot$ denotes element-wise multiplication. The scaling factor s rescales offsets before Fourier embedding, with a fixed coefficient κ. We initialize

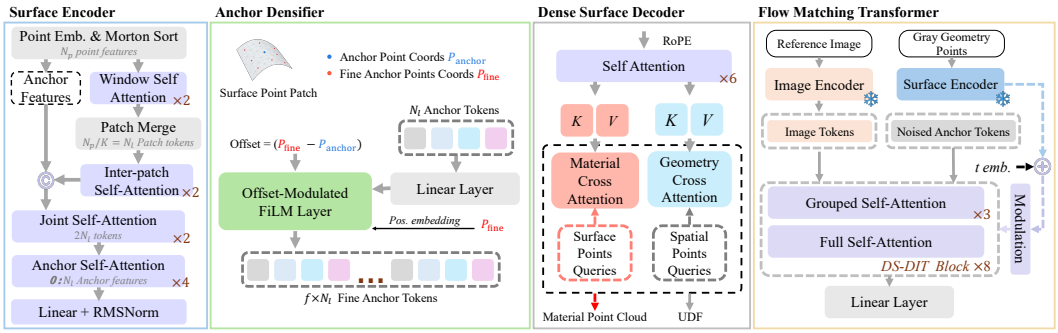


Figure 3: Network components of DSTex. The Surface Encoder concatenates anchor features and patch tokens (C) for joint attention, then refines the anchor subsequence ($0{:}N_l$) before latent projection and RMS normalization. The Anchor Densifier constructs $N_f = fN_l$ fine anchor tokens through projection and position-dependent FiLM; P_{anchor} and P_{fine} denote coordinates A and B. The Dense Surface Decoder supports albedo/MR queries and auxiliary UDF supervision. DS-DiT predicts velocities for the compact latents. Snowflakes mark the frozen surface encoder and image backbone during diffusion training.

P_{pos} and F to zero, so each fine anchor token initially equals $S(\mathbf{h}_i)$. All three projections are shared across fine anchor tokens, with parameter shapes independent of f.

The Dense Surface Decoder refines the fine anchor tokens $\{\mathbf{h}_{ik}^{(0)}\}$ with full self-attention and three-axis RoPE, producing $H \in \mathbb{R}^{N_f \times D}$ at coordinates B. These tokens supply both keys and values for the query heads. The Material Cross Attention branch groups independent albedo and MR heads, whereas Geometry Cross Attention predicts an auxiliary unsigned distance field:

$$\hat{\mathbf{y}}(\mathbf{x}, \mathbf{n}) = \big[D_{\text{alb}}(\mathbf{x}, \mathbf{n}; H, B); D_{\text{MR}}(\mathbf{x}, \mathbf{n}; H, B)\big], \quad \hat{u}(\mathbf{x}) = D_{\text{geo}}(\mathbf{x}; H, B). \tag{6}$$

Here $\hat{\mathbf{y}}$ predicts normalized material values; their original units are recovered by $(\hat{\mathbf{y}} + 1)/2$. Each material head embeds the query position and geometric normal and attends globally to the fine anchor tokens. Surface queries produce albedo and MR values; spatial queries provide geometry supervision during autoencoder training.

The number of output queries is independent of N_l and N_f. As in decoder-side upsampling for geometry reconstruction (Hao et al., 2026), increasing f adds decoder computation while leaving the diffusion appearance sequence at N_l tokens. Changing N_l instead changes the latent budget, as in prior anchor-count scaling (Lai et al., 2026a). The baseline DS-VAE w/o densifier uses $f = 1$ with absolute position embeddings at A and no relative FiLM.

3.3 Autoencoder Objective

Consistent with the observations of LaFiTe (Chen et al., 2026), rendering-based losses can introduce texture blurring and bias. We therefore supervise material reconstruction directly at Q independently sampled surface points. These queries are sampled separately from the encoder input and use face normals from the mesh. We map each query's material target to $\mathbf{y}_q = 2\mathbf{t}_q - 1$ and compute separate Smooth L1 losses for albedo and MR, each averaged over the queries and its respective channels. The material loss combines them with relative weight λ_{MR} for MR:

$$\mathcal{L}_{\text{tex}} = \frac{\mathcal{L}_{\text{alb}} + \lambda_{\text{MR}}\mathcal{L}_{\text{MR}}}{1 + \lambda_{\text{MR}}}. \tag{7}$$

To mitigate albedo color shifts, we include an auxiliary CIELAB loss that penalizes lightness and chromaticity errors separately at each query.

We supervise the auxiliary UDF head with the truncated target $\min(d_{\mathcal{M}}(\mathbf{x}), \tau)/\tau$, where $d_{\mathcal{M}}$ is distance to the mesh and τ is the truncation distance. Its Smooth L1 loss, $\mathcal{L}_{\text{UDF}}$, combines volume and near-surface samples. The combined objective is

$$\mathcal{L}_{\text{recon}} = \mathcal{L}_{\text{tex}} + \lambda_{\text{Lab}}\mathcal{L}_{\text{Lab}}^{\text{alb}} + \lambda_{\text{UDF}}\mathcal{L}_{\text{UDF}}, \tag{8}$$

where λ_{Lab} and λ_{UDF} weight the auxiliary objectives. Latent regularization uses the token normalization and fixed posterior noise in Equation 2; no KL divergence term is applied.

3.4 DS-DiT: Latent Diffusion

We freeze the trained autoencoder and pre-encode its posterior means for diffusion training. This produces a latent sequence $Z_0 \in \mathbb{R}^{N_l \times d}$ paired with anchor coordinates A. We standardize each latent channel using its training-set mean and standard deviation, obtaining $\overline{Z}_0$.

The Gray Geometry Points branch in Figure 2 uses the frozen Surface Encoder to obtain an aligned geometry latent set $G \in \mathbb{R}^{N_l \times d}$. We retain the sampled positions, normals, and Morton ordering, and replace material values with a neutral matte dielectric. Encoding this point set produces one geometry token per anchor token. After channel-wise standardization, a projection of geometry token $\overline{\mathbf{g}}_i$ is added to the timestep-conditioning vector $\mathbf{e}_t$:

$$\mathbf{c}_i(t) = \mathbf{e}_t + W_g \overline{\mathbf{g}}_i. \tag{9}$$

These vectors drive per-anchor modulation of appearance tokens (Figure 3).

Following multi-layer image conditioning in Luce (Singh et al., 2026), we extract features from layers 8, 16, 24, and 32 of a frozen DINOv3 ViT-H/16+ encoder (Siméoni et al., 2025). We RMS-normalize each layer's patch features separately, concatenate corresponding features along channels, and project them to the transformer width before refinement, preserving one image token per patch. The Flow Matching Transformer jointly processes noisy anchor and image tokens (Peebles & Xie, 2023; Esser et al., 2024). Grouped attention layers process appearance windows and the context tokens separately; interleaved full-attention layers allow interaction across modalities. Four-axis RoPE encodes modality and spatial positions. A final projection reads a d-channel velocity from each of the N_l anchor tokens.

We train the generator with the rectified-flow objective (Liu et al., 2023). For Gaussian noise $Z_1 \sim \mathcal{N}(0, I)$ and timestep t, we interpolate between clean standardized latents and noise:

$$Z_t = (1-t)\overline{Z}_0 + tZ_1, \qquad \mathcal{L}_{\text{flow}} = \mathbb{E}\left[\frac{1}{N_l d}\left\|v_\theta(Z_t, t; \mathcal{C}) - (Z_1 - \overline{Z}_0)\right\|_F^2\right], \tag{10}$$

where $\mathcal{C}$ collects the anchor coordinates, geometry latents, and image tokens. At inference, we sample Gaussian anchor tokens and integrate the learned velocity from $t = 1$ to $t = 0$ under the input conditions. After reversing channel standardization, the frozen DS-VAE decodes the generated tokens into surface materials.

4 Experiments

Datasets and evaluation benchmarks. We use TRELLIS-500K (Xiang et al., 2025) together with a 120K-asset PBR subset of TexVerse (Zhang et al., 2025) as data sources for training the VAE and latent generator. From TRELLIS-500K, we randomly hold out 20,000 assets from training, then randomly select 1,000 GLB assets with ground-truth PBR materials from this split for both VAE reconstruction and quantitative texture generation evaluation. Quantitative reference images are near-frontal renders of these assets under a fixed studio HDRI environment. For qualitative generation experiments, we use reference images generated with GPT Image 2.5 (OpenAI, 2026), paired with meshes produced by commercial 3D generation tools (Meshy AI, 2026).

Baselines. For texture generation, we compare with LumiTex (Bao et al., 2026), UniTEX (Liang et al., 2026), and the texturing pipeline of TRELLIS.2 (Xiang et al., 2026) on the same meshes and reference images. For reconstruction, we compare with the texture SC-VAE of TRELLIS.2 at voxel resolutions 512, 1024, and 1536, alongside controlled ablations of our autoencoder. All GLB assets used for method comparisons and visualizations have 2K UV texture maps. All methods use the same UV layout per mesh. For relighting, color-only outputs use fixed default metallic and roughness values. Unless otherwise specified, our VAE and DS-DiT use 8K anchor tokens for training and 32K for inference. DS-DiT uses 25 sampling steps and classifier-free guidance (CFG) scale 3.

Metrics. *Generation:* We organize the comparison into texture and relighting evaluation, following LumiTex (Bao et al., 2026). Texture evaluation compares baked albedo UV maps with ground truth; relighting evaluation uses six matched views under a different HDRI from the reference image, shared by predictions and ground truth. Each group reports CLIP-FID, CMMD (Jayasumana et al., 2024), CLIP-I, and LPIPS, separating texture-map quality from rendered appearance. We report end-to-end runtime from input GLB loading to textured asset export (Appendix C). *Reconstruction:* We report



Figure 4: Qualitative comparison of open-source texture generation methods. From left to right: inputs, LumiTex, TRELLIS.2, UniTEX, and ours. The input column pairs each reference image with its untextured mesh. The first four rows show two views per generated asset; the last row shows a furnished-room example. Colored boxes mark corresponding detail crops.

PSNR, SSIM, and LPIPS separately for albedo and metallic–roughness (MR). All three metrics compare reconstructed and ground-truth material-channel images rendered from six shared views per asset at 1024×1024 resolution. Appendix B defines the metrics and rendering settings.

4.1 Main Results

Table 1: Texture generation on 1,000 evaluation assets with shared meshes and reference images. We compare albedo UV maps and six relit views per asset against ground truth. UniTEX uses metallic 0 and roughness 0.5. Both DSTex variants use 32K inference tokens. Time is averaged over the same 1,000 assets for all methods, from GLB loading through textured GLB/2K PNG export. The **best** and second-best results are highlighted.

		Texture Evaluation (UV level)				Relighting Evaluation (image level)				
Method	Type	CLIP-FID ↓	CMMD ↓	CLIP-I ↑	LPIPS ↓	CLIP-FID ↓	CMMD ↓	CLIP-I ↑	LPIPS ↓	Time (s) ↓
LumiTex	PBR	13.78	1.148	0.9492	0.2223	13.36	1.251	0.9433	0.0912	190.4
UniTEX	Texture	16.67	1.228	0.9377	0.2710	15.02	1.323	0.9361	0.1134	83.5
TRELLIS.2	PBR	13.37	1.023	0.9506	0.2142	11.39	1.069	0.9518	0.0767	**28.6**
DSTex w/o densifier	PBR	11.13	0.778	0.9598	0.1833	10.45	0.920	0.9559	0.0734	73.61
DSTex (Ours)	PBR	**10.80**	**0.774**	**0.9600**	**0.1767**	**9.98**	**0.875**	**0.9581**	**0.0663**	99.42

Texture generation. Table 1 compares the open-source pipelines on shared inputs; DSTex achieves the best scores across all four metrics in both texture and relighting evaluation. Figure 4 shows sharper patterns and more accurate placement than TRELLIS.2, including robot lettering (row 4) and collar and chest motifs (row 3).

Compared with LumiTex and UniTEX, which use pretrained image generators (Bao et al., 2026; Liang et al., 2026), our results retain fine bottle ornamentation (row 2) that their outputs blur or distort. Appendices E.1 and E.2 provide ten further comparisons and additional PBR generations with material channels.

Reconstruction. At matched training and inference latent budgets, DS-VAE improves all six albedo/MR reconstruction metrics over its counterpart without the Anchor Densifier (Table 2(e)). Figure 5 shows clearer small lettering and separation between strokes and lines, where the unexpanded model blurs and merges details. These results support expanding the decoder's surface representation to recover finer material details while keeping the latent sequence compact.



Figure 5: Fine-detail reconstruction. Left to right: ground truth, DS-VAE, TRELLIS.2 ($R = 1536$), and DS-VAE without the Anchor Densifier. Green boxes mark the enlarged label regions.

DS-VAE surpasses TRELLIS.2's texture SC-VAE at resolutions 512 and 1024 on all six metrics. At 1536, TRELLIS.2 has higher albedo PSNR and SSIM; DS-VAE achieves comparable albedo LPIPS and better MR scores. Both recover the printed-label details in Figure 5.

Table 2: Texture reconstruction on 1,000 evaluation assets using six 1024×1024 material-channel renders per asset: (a–d) design ablations; (e) autoencoder comparison. MR: metallic–roughness; f: expansion factor; K: points per patch; R: voxel resolution. PSNR is in dB. The first DS-VAE row in (e) is the shared reference for (b–d). The **best** and second-best scores are marked separately for the 4K/16K configurations and the 32K DS-VAE models together with TRELLIS.2. See Appendix B.1.

	Anchor tokens		Up	Patch	Albedo			MR		
Model / configuration	Train	Infer	factor f	size K	PSNR ↑	LPIPS ↓	SSIM ↑	PSNR ↑	LPIPS ↓	SSIM ↑
(a) Surface encoding; decoder fixed to $f = 1$										
Cross-attention encoder	4K	16K	1	–	29.40	0.2465	0.8386	31.51	0.1205	0.8719
Surface Encoder	4K	16K	1	32	31.63	0.1820	0.8732	33.81	0.0746	0.8848
(b) Anchor expansion; encoder and position conditioning fixed										
Direct expansion	4K	16K	4	32	33.17	0.1448	0.9045	34.59	0.0625	0.9162
Progressive expansion	4K	16K	$2 \to 4 \to 8$	32	33.30	**0.1116**	0.9115	33.35	0.0577	**0.9208**
(c) Position conditioning; encoder and $f = 8$ fixed										
Absolute PE only	4K	16K	8	32	33.34	0.1349	0.9043	34.89	**0.0552**	0.9169
(d) Patch size; $f = 8$ and position conditioning fixed										
Patch size $K = 16$	4K	16K	8	16	33.03	0.1430	0.9025	33.12	0.0642	0.9075
Patch size $K = 64$	4K	16K	8	64	33.47	0.1265	0.8955	34.00	0.0611	0.8841
(e) Autoencoder comparison										
DS-VAE (Ours)	4K	16K	8	32	**34.22**	0.1342	**0.9154**	**35.21**	0.0618	0.9091
DS-VAE w/o densifier	8K	32K	1	32	36.21	0.0888	0.9357	38.46	0.0326	0.9476
DS-VAE (Ours)	8K	32K	8	32	38.94	**0.0442**	0.9606	**40.61**	**0.0160**	**0.9659**
TRELLIS.2	Voxel resolution $R = 512$				31.41	0.1686	0.8890	26.45	0.1668	0.8348
TRELLIS.2	Voxel resolution $R = 1024$				36.61	0.0755	0.9452	29.99	0.1203	0.8781
TRELLIS.2	Voxel resolution $R = 1536$				**39.87**	0.0450	**0.9631**	32.24	0.1060	0.8968

4.2 DS-VAE Ablations

Table 2(a–d) evaluates four design choices at 4K training and 16K inference tokens. Groups (b–d) use DS-VAE (4K/16K) in (e) as their reference configuration.

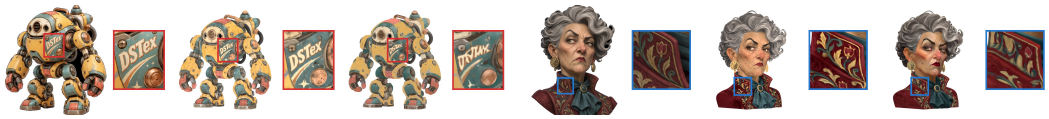
Surface encoding. With densification disabled ($f = 1$), the Surface Encoder improves all six metrics over point-to-latent cross-attention (group (a)).

Anchor expansion. Direct expansion to $f = 4$ and $f = 8$ improves albedo and MR PSNR over $f = 1$ in (a). Direct $f = 8$ gives the highest PSNR, while progressive expansion ($2 \to 4 \to 8$) gives better LPIPS and MR SSIM.



Reference (a) 16K, $f = 1$ (b) 16K, $f = 8$ (c) 32K, $f = 1$ (d) 32K, $f = 8$

Figure 6: Albedo generation at 16K and 32K inference anchor tokens. At each budget, DS-VAE without the densifier ($f = 1$) and the full model ($f = 8$) use their corresponding generators. The leftmost panel shows the reference with a geometry inset; red and blue boxes mark enlarged patterns.



Reference Multi-layer Single-layer Reference Multi-layer Single-layer

Figure 7: Image-conditioning ablation. Albedo outputs use DINOv3 layers 8, 16, 24, and 32 (Multi-layer) or layer 32 (Single-layer). Each group includes a reference image and detail crops.

Position conditioning. Relative FiLM improves albedo/MR PSNR and albedo SSIM over absolute PE alone (Table 2(c,e)); absolute PE gives better MR LPIPS and SSIM.

Patch size. Among the patch sizes at fixed latent counts and $f = 8$, $K = 32$ gives the highest albedo/MR PSNR, while $K = 64$ gives lower LPIPS but lower SSIM. Since $N_p = KN_l$, input sample count also varies with patch size.

4.3 DS-DiT Ablations

Densification and test-time scaling. At matched latent counts, densification improves all four texture and relighting metrics (Table 1). Figure 6 scales each VAE–DM pair from 16K to 32K anchor tokens with fixed weights. Densification preserves clearer linked-diamond motifs (blue crops) at both scales, but increases end-to-end inference time (Table 1). The persistent detail gap at 32K tokens suggests that decoder-side densification remains complementary to increasing the latent budget at inference.

Image conditioning. At matched image token counts and widths, multi-layer conditioning produces more legible robot lettering and more coherent collar motifs than final-layer conditioning (Figure 7).

5 Conclusion

We present DSTex, a native PBR texture generation framework for recovering fine surface details from compact latents. DS-VAE expands anchor tokens at mesh-sampled locations, increasing decoder capacity without lengthening the diffusion sequence. DS-DiT generates these compact latents from an image and geometry for DS-VAE to decode into surface materials. Experiments show improved reconstruction at matched latent counts and better texture and relighting scores than the evaluated open-source baselines. These findings highlight decoder-side surface capacity as an effective direction for native PBR texture generation.

Limitations and future work. We observe slower diffusion convergence with larger VAE expansion factors, motivating joint study of decoder capacity and latent learnability for more efficient training. This makes the expansion factor a trade-off between reconstruction fidelity and the training effort required by the latent generator. Tracking both reconstruction and generation quality over training could help select decoder configurations that offer stronger material detail within a fixed training compute budget. Single-image conditioning also leaves unobserved surfaces weakly constrained; multi-view guidance could improve control over these regions. Iterative denoising and dense surface decoding remain costly. Adaptive densification could focus decoder computation on regions with intricate surface appearance. Diffusion distillation and model quantization could reduce sampling steps and memory requirements, respectively, while preserving texture detail and material fidelity.

AI USE STATEMENT

We use GPT Image 2.5 to generate reference images for qualitative experiments. The corresponding input grids are produced using commercial 3D modeling software. In addition, we employ an AI assistant powered by large language models to provide feedback on experimental design, assist with literature search and review writing, help draft and polish the initial manuscript, refine the logical structure, and format references. The design of the research plan, the selection of methods, and the experimental procedures are carried out independently by the authors. Before submission, the authors individually verify that the AI-assisted outputs are consistent with the original materials and experimental records, and take full responsibility for all methodological decisions, experimental results, and the final conclusions.

ETHICS STATEMENT

DSTex is intended to support 3D content creation by generating surface materials on existing meshes. Our experiments use assets from TRELLIS-500K and TexVerse and identify the AI-generated reference images and meshes used for qualitative generation evaluation. Potential concerns include reproducing protected designs or recognizable branding and creating misleading synthetic depictions of objects. Use and redistribution of source assets and generated content should respect the applicable permissions and preserve provenance information. The learned appearance distribution may also reflect biases in the asset collection and reference-image generator; performance on these inputs does not establish equal coverage across object categories or visual styles.

REPRODUCIBILITY STATEMENT

Section 3 specifies the representation, network operations, and training objectives. Appendix A provides network dimensions, conditioning features, loss settings, and training and inference token counts. Section 4 describes the data source, evaluation-split construction, and comparison methods, while Appendix B records the reconstruction and generation evaluation protocols. Table 2 lists the training and inference token counts, expansion factors, and patch sizes for the reconstruction comparisons, making the settings of the design ablations and final models explicit.

REFERENCES

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J. Guibas. Learning representations and generative models for 3d point clouds. In *ICML*, 2018.

Jingzhi Bao, Hongze Chen, Lingting Zhu, Chenyu Liu, Runze Zhang, Keyang Luo, Zeyu Hu, Weikai Chen, Yingda Yin, Xin Wang, Zehong Lin, Jun Zhang, and Xiaoguang Han. LumiTex: Towards high-fidelity PBR texture generation with illumination context. In *ICLR*, 2026.

Tianshi Cao, Karsten Kreis, Sanja Fidler, Nicholas Sharp, and Kangxue Yin. Texfusion: Synthesizing 3d textures with text-guided image diffusion models. In *ICCV*, 2023.

Jen-Hao Rick Chang, Xiaoming Zhao, Dorian Chan, and Oncel Tuzel. LiTo: Surface light field tokenization. In *ICLR*, 2026.

Chia-Hao Chen, Yuan-Chen Guo, Zi-Xin Zou, Ze Yuan, Guan Luo, Xiaojuan Qi, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Lafite: A generative latent field for 3d native texturing. In *CVPR*, 2026.

Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. In *ICCV*, 2023.

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In *ICLR*, 2025a.

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 *CVPR*, 2025b.

Yinbo Chen, Oliver Wang, Richard Zhang, Eli Shechtman, Xiaolong Wang, and Michael Gharbi. Image neural field diffusion models. In *CVPR*, 2024.

Zhaoxi Chen, Jiaxiang Tang, Yuhao Dong, Ziang Cao, Fangzhou Hong, Yushi Lan, Tengfei Wang, Haozhe Xie, Tong Wu, Shunsuke Saito, Liang Pan, Dahua Lin, and Ziwei Liu. 3DTopia-XL: Scaling high-quality 3D asset generation via primitive diffusion. In *CVPR*, 2025c.

Wei Cheng, Juncheng Mu, Xianfang Zeng, Xin Chen, Anqi Pang, Chi Zhang, Zhibin Wang, Bin Fu, Gang Yu, Ziwei Liu, and Liang Pan. Mvpaint: Synchronized multi-view diffusion for painting anything 3d. In *CVPR*, 2025.

Kangle Deng, Timothy Omernick, Alexander Weiss, Deva Ramanan, Jun-Yan Zhu, Tinghui Zhou, and Maneesh Agrawala. FlashTex: Fast relightable mesh texturing with LightControlNet. In *ECCV*, 2024.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024.

Yifei Feng, Mingxin Yang, Shuhui Yang, Sheng Zhang, Jiaao Yu, Zibo Zhao, Yuhong Liu, Jie Jiang, and Chunchao Guo. RomanTex: Decoupling 3D-aware rotary positional embedded multi-attention network for texture synthesis. In *ICCV*, 2025.

Philippe Hansen-Estruch, David Yan, Ching-Yao Chung, Orr Zohar, Jialiang Wang, Tingbo Hou, Tao Xu, Sriram Vishwanath, Peter Vajda, and Xinlei Chen. Learnings from scaling visual tokenizers for reconstruction and generation. *arXiv:2501.09755*, 2025.

Dehao Hao, Kaiyi Zhang, Tanghui Jia, Xiangjun Gao, Dongyu Yan, Weikai Chen, Zeyu Hu, Lingting Zhu, Yingda Yin, Runze Zhang, Li Yuan, Xin Wang, and Long Quan. MSVS-VAE: Multi-scale anchored VecSet for high-fidelity 3D reconstruction. *arXiv:2607.24436*, 2026.

Huiang He, Shengchu Zhao, Jianwen Huang, Jie Li, Jiaqi Wu, Hu Zhang, Pei Tang, Heliang Zheng, Yukun Li, and Rongfei Jia. Hitem3d 2.0: Multi-view guided native 3d texture generation. *arXiv:2604.09231*, 2026.

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 *ICCV*, 2025a.

Zebin He, Mingxin Yang, Shuhui Yang, Yixuan Tang, Tao Wang, Kaihao Zhang, Guanying Chen, Yuhong Liu, Jie Jiang, Chunchao Guo, and Wenhan Luo. MaterialMVP: Illumination-invariant material generation via multi-view PBR diffusion. In *ICCV*, 2025b.

Xin Huang, Tengfei Wang, Ziwei Liu, and Qing Wang. Material anything: Generating materials for any 3D object via diffusion. In *CVPR*, 2025a.

Zehuan Huang, Yuan-Chen Guo, Haoran Wang, Ran Yi, Lizhuang Ma, Yan-Pei Cao, and Lu Sheng. Mv-adapter: Multi-view consistent image generation made easy. In *ICCV*, 2025b.

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:2506.15442*, 2025.

Dong Huo, Zixin Guo, Xinxin Zuo, Zhihao Shi, Juwei Lu, Peng Dai, Songcen Xu, Li Cheng, and Yee-Hong Yang. TexGen: Text-guided 3D texture generation with multi-view sampling and resampling. In *ECCV*, 2024.

Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. Rethinking FID: Towards a better evaluation metric for image generation. In *CVPR*, 2024.

Tanghui Jia, Dongyu Yan, Dehao Hao, Yang Li, Kaiyi Zhang, Xianyi He, Lanjiong Li, Yuhan Wang, Jinnan Chen, Lutao Jiang, Qishen Yin, Long Quan, Ying-Cong Chen, and Li Yuan. UltraShape 1.0: High-fidelity 3D shape generation via scalable geometric refinement. *arXiv:2512.21185*, 2025.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *TOG*, 2023.

Zeqiang Lai, Yunfei Zhao, Haolin Liu, Zibo Zhao, Qingxiang Lin, Huiwen Shi, Xianghui Yang, Mingxin Yang, Shuhui Yang, Yifei Feng, et al. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details. *arXiv:2506.16504*, 2025.

Zeqiang Lai, Yunfei Zhao, Zibo Zhao, Haolin Liu, Qingxiang Lin, Jingwei Huang, Chunchao Guo, and Xiangyu Yue. Lattice: Democratize high-fidelity 3d generation at scale. In *CVPR*, 2026a.

Zeqiang Lai, Yunfei Zhao, Zibo Zhao, Xin Yang, Xin Huang, Jingwei Huang, Xiangyu Yue, and Chunchao Guo. Natex: Seamless texture generation as latent color diffusion. In *CVPR*, 2026b.

Hong Li, Chongjie Ye, Houyuan Chen, Weiqing Xiao, Ziyang Yan, Lixing Xiao, Zhaoxi Chen, Jianfeng Xiang, Shaocong Xu, Xuhui Liu, Yikai Wang, Baochang Zhang, Xiaoguang Han, Jiaolong Yang, and Hao Zhao. NeAR: Coupled neural asset-renderer stack. In *CVPR*, 2026.

Weiyu Li, Xuanyang Zhang, Zheng Sun, Di Qi, Hao Li, Wei Cheng, Weiwei Cai, Shihao Wu, Jiarui Liu, Zihao Wang, Xiao Chen, Feipeng Tian, Jianxiong Pan, Zeming Li, Gang Yu, Xiangyu Zhang, Daxin Jiang, and Ping Tan. Step1X-3D: Towards high-fidelity and controllable generation of textured 3D assets. *arXiv:2505.07747*, 2025a.

Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, et al. Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models. *TPAMI*, 2025b.

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

Yixun Liang, Kunming Luo, Xiao Chen, Rui Chen, Hongyu Yan, Weiyu Li, Jiarui Liu, Fei-Peng Tian, and Ping Tan. Unitex: Universal high fidelity generative texturing for 3d shapes. In *CVPR*, 2026.

Chenyu Liu, Hongze Chen, Jingzhi Bao, Lingting Zhu, Runze Zhang, Weikai Chen, Zeyu Hu, Yingda Yin, Keyang Luo, and Xin Wang. CaliTex: Geometry-calibrated attention for view-coherent 3D texture generation. In *CVPR*, 2026a.

Hangyu Liu, Jianyong Wang, and Yutao Sun. Geometric autoencoder for diffusion models. *arXiv:2603.10365*, 2026b.

Jialun Liu, Chenming Wu, Xinqi Liu, Xing Liu, Jinbo Wu, Haotian Peng, Chen Zhao, Haocheng Feng, Jingtuo Liu, and Errui Ding. Texoct: Generating textures of 3d models with octree-based diffusion. In *CVPR*, 2024a.

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

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

Yifan Lu, Qi Wu, Jay Zhangjie Wu, Zian Wang, Huan Ling, Sanja Fidler, and Xuanchi Ren. PiD: Fast and high-resolution latent decoding with pixel diffusion. *arXiv:2605.23902*, 2026.

Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019.

Meshy AI. Meshy AI, 2026. URL `https://www.meshy.ai/`. Accessed September 25, 2026.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Thomas W. Mitchel, Carlos Esteves, and Ameesh Makadia. Single mesh diffusion models with field latents for texture generation. In *CVPR*, 2024.

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *ICML*, 2020.

Michael Oechsle, Lars M. Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *ICCV*, 2019.

OpenAI. Introducing ChatGPT Images 2.5, 2026. URL `https://openai.com/index/introducing-chatgpt-images-2-5/`.

Jeong Joon Park, Peter R. Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

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

Andrey Shutkin, Denis Parkhomenko, Ivan Kirillov, Kirill Chernyshev, Kirill Malakhov, Ilia Vasiliev, Ilia Trushkin, Valeriya Kobenko, David Chikovani, Alexander Ivanov, Azat Saginbaev, Egor Silvestrov, Ivan Mikheev, and Konstantin Zakharov. KVAE: Family of tokenizers for multimodal generative models. *arXiv:2608.05798*, 2026.

Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Texturify: Generating textures on 3d shape surfaces. In *ECCV*, 2022.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *arXiv:2508.10104*, 2025.

Mayank Singh, Michele Stoppa, Alvise Memo, Rui Yu, Harsha Kalli, Srimanth Gunturi, Muhammad Ahmed Riaz, Behrooz Shahsavari, Waleed Abdulla, and David E. Jacobs. Luce: Relightable gaussians for 3D asset generation. *arXiv:2608.23943*, 2026.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024.

Karthik Vaidyanathan, Marco Salvi, Bartlomiej Wronski, Tomas Akenine-Möller, Pontus Ebelin, and Aaron Lefohn. Random-access neural compression of material textures. *TOG*, 2023.

Xiuchao Wu, Pengfei Zhu, Jiangjing Lyu, Xinguo Liu, Jie Guo, Yanwen Guo, Weiwei Xu, and Chengfei Lyu. MatMart: Material reconstruction of 3D objects via diffusion. In *CVPR*, 2026.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015.

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 *CVPR*, 2025.

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. In *CVPR*, 2026.

Bojun Xiong, Jialun Liu, Jiakui Hu, Chenming Wu, Jinbo Wu, Xing Liu, Chen Zhao, Errui Ding, and Zhouhui Lian. Texgaussian: Generating high-quality PBR material via octree-based 3d gaussian splatting. In *CVPR*, 2025.

Kyeongmin Yeo, Yunhong Min, Jaihoon Kim, and Minhyuk Sung. MatLat: Material latent space for PBR texture generation. In *CVPR*, 2026.

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 *CVPR*, 2024.

Xin Yu, Peng Dai, Wenbo Li, Lan Ma, Zhengzhe Liu, and Xiaojuan Qi. Texture generation on 3d meshes with point-uv diffusion. In *ICCV*, 2023.

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. *TOG*, 2024.

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 *CVPR*, 2024.

Yifei Zeng, Yajie Bao, Jiachen Qian, Shuang Wu, Youtian Lin, Hao Zhu, Buyu Li, Feihu Zhang, Xun Cao, and Yao Yao. Textrix: Latent attribute grid for native texture generation and beyond. In *CVPR*, 2026.

Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *TOG*, 2023.

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. *TOG*, 2024.

Yibo Zhang, Li Zhang, Rui Ma, and Nan Cao. TexVerse: A universe of 3D objects with high-resolution textures. *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:2501.12202*, 2025.

A Architecture and Implementation Details

A.1 Training Details

We select TexVerse (Zhang et al., 2025) training assets from its CC BY 4.0 and CC0 1.0 categories, which contain 699,075 and 6,139 assets, respectively. Filtering within these two categories yields the 120K-asset PBR subset used in our experiments.

DS-VAE is trained in two stages: the 4K-token variants in Table 2 receive 5M sample presentations, and the 8K-token variants continue from their corresponding 4K checkpoints for another 5M, reaching 10M in total. Both stages use a combined Muon and AdamW optimizer, a constant learning rate of 10^{-4}, and a global batch size of 64 (two per GPU). DS-VAE is trained on 32 NVIDIA A100 SXM4 GPUs for approximately three days. For diffusion training and inference, we freeze the corresponding 8K-token VAE checkpoints at 10M sample presentations. DS-DiT is trained on 64 NVIDIA A100 SXM4 80GB GPUs for 300M sample presentations at a global batch size of 128, corresponding to approximately 2.34M updates. It uses Muon and AdamW, warming up over the first 1M sample presentations before maintaining a constant learning rate of 10^{-3}. These budgets count global batch size $\times$ training iterations, including repeated presentations of the same assets.

Final evaluations use both VAE–DM pairs at the 8K training stage with 32K (32,768) anchor tokens for inference and 16K (16,384) for scaling comparisons. The 4K ablations use 16K inference anchor tokens. Scaling comparisons keep each VAE–DM pair fixed.

A.2 Texture Autoencoder

At 8K anchor tokens, the autoencoder uses $N_p = 262{,}144$ surface samples and patch size $K = 32$. Each latent has $d = 64$ channels. Input features combine 384-dimensional Fourier position features, 36 spherical-harmonic coefficients through degree five for normals, and learned albedo and MR embeddings. The point encoder has two windowed self-attention blocks with windows of 2,048 samples, a linear merge over each group of 32 samples, and two inter-patch blocks. Two joint blocks combine anchor features with patch tokens, followed by four anchor-only blocks. The encoder width is 768, with 12 attention heads.

The latent mean uses parameter-free RMS normalization with fixed posterior noise $\sigma = 0.05$. The decoder projects the latents to width $D = 768$ and expands them by $f = 8$, producing $N_f = fN_l$ fine anchor tokens before six full self-attention blocks. Its position embedding has 384 channels, and the relative-offset coefficient in Equation 5 is $\kappa = 0.5$. Each material head uses one global cross-attention block and an output projection for its target channels. Figure 3 details the decoder.

The progressive-expansion variant in Table 2 retains six decoder self-attention blocks. It expands the sequence to $2N_l$, $4N_l$, and $8N_l$ tokens before blocks 0, 2, and 4, respectively, with blocks indexed from zero. Each resolution therefore spans two blocks, whereas direct expansion uses the final token count throughout all six blocks.

A.3 Reconstruction Supervision

We draw $Q = 131{,}072$ texture supervision points independently of the encoder input. Material values are mapped from $[0, 1]$ to $[-1, 1]$, retaining linear-light albedo. The albedo and MR Smooth L1 losses use threshold one and are each averaged over their respective channels and supervision points. We combine them as $(\mathcal{L}_{\mathrm{alb}} + 0.2\mathcal{L}_{\mathrm{MR}})/1.2$, with $\lambda_{\mathrm{MR}} = 0.2$ in Equation 7. The albedo Lab loss maps predictions and targets back to $[0, 1]$ and clamps them before converting linear RGB to CIELAB. It uses unnormalized CIELAB coordinates, with weight one on the L^* mean squared error and weight two on the mean squared error of (a^*, b^*). The auxiliary loss coefficients are $\lambda_{\mathrm{Lab}} = 0.002$ and $\lambda_{\mathrm{UDF}} = 1$; the Lab term applies only to albedo. UDF supervision uses truncation distance $\tau = 0.2$, with 32,768 volume samples and 32,768 near-surface samples. Its Smooth L1 threshold is also one.

A.4 DS-DiT

DS-DiT is trained on posterior-mean latents from its corresponding frozen DS-VAE. Appearance and geometry latents are each standardized using their own per-channel training-set statistics. The geometry condition retains positions and normals, with albedo 0.5, metallic zero, and roughness one.

The 2B-parameter DS-DiT has 32 transformer blocks of width 1,536, with 12 heads of width 128. The first 16 blocks apply attention and feed-forward layers sequentially; the remaining 16 apply them in parallel. One full-attention block per group of four permits appearance-to-condition interaction, giving eight such blocks in total. The other blocks process appearance windows separately from the image context group. Anchor coordinates are quantized to a 128^3 grid for the modality-aware four-axis RoPE. The final output projection predicts 64 velocity channels per anchor token.

Image conditioning uses one reference image at 1,024-pixel resolution. The frozen DINOv3 ViT-H/16+ backbone provides features from layers 8, 16, 24, and 32, numbering transformer blocks from one. We RMS-normalize each layer's patch features separately and concatenate corresponding features along channels. An image refiner projects the concatenated features to 1,536 channels, applies RMS normalization, and processes them with two transformer blocks, each containing self-attention and a feed-forward layer. This preserves the sequence of 4,096 image tokens. The single-layer variant instead supplies only layer-32 features to the refiner, with the same output width and sequence length.

For rectified flow, we sample $u \sim \mathcal{N}(0.3, 1)$, set $t_0 = \mathrm{sigmoid}(u)$ and $t = 3t_0/(1 + 2t_0)$, and weight the velocity loss uniformly. During training, we independently drop the image condition for each sample with probability 0.2 by zeroing the input RGB values while retaining the image tokens. Geometry conditioning is retained for every sample.

We sample with a first-order rectified-flow Euler solver using timestep shift 2.0, compared with 3.0 during training. Unless otherwise specified, sampling uses 25 steps and CFG scale 3 throughout the full noise range. The two CFG branches share the geometry condition and differ only in whether the reference image is retained or zeroed, so guidance acts on appearance conditioning. We restore the autoencoder's channel scale before decoding.

B Evaluation Protocol

Reconstruction and quantitative generation use the same 1,000 GLB assets with ground-truth PBR materials, randomly selected from our 20,000-asset evaluation split of TRELLIS-500K, which is excluded from our training. For each asset, predictions from all methods are baked onto the same geometry and UV layout as the ground truth, using 2K texture maps.

B.1 Material Reconstruction

Table 2 reports PSNR, SSIM, and LPIPS on paired reconstructed and ground-truth material-channel images, rendered without illumination from six views per asset at 1024×1024 resolution. DS-VAE and TRELLIS.2 use identical cameras, backgrounds, texture filtering, and linear RGB encoding for all three metrics. Albedo is evaluated as a three-channel RGB image; MR packs metallic and roughness into the first two channels of an image, with the third fixed to zero. LPIPS uses the AlexNet backbone. We average each metric over views and then equally over assets, reporting albedo and MR separately.

External texture autoencoder. TRELLIS.2 uses its texture SC-VAE to encode and decode voxelized ground-truth materials; the decoded field is sampled on the original mesh. Errors include input voxelization and surface readout. Table 2(e) compares TRELLIS.2 at voxel resolutions 512, 1024, and 1536 with both final DS-VAE variants at 32K inference anchor tokens. Although excluded from our training, these evaluation assets may overlap with the training data of the pretrained TRELLIS.2 models.

B.2 Generation and Rendering

For each asset, we render its ground-truth PBR materials from a near-frontal view under a fixed studio HDRI environment to obtain a single reference image. All open-source methods receive the same mesh and reference image.

UV texture evaluation. We compare each baked albedo UV map with its ground-truth counterpart, applying the same mask to untextured atlas regions for every method. CLIP-I and LPIPS are computed for each asset pair and then averaged over the test set.

Relighting evaluation. We use a different HDRI from the reference-image environment, rendering predicted and ground-truth materials under this shared illumination and six matched camera views at 1024×1024 resolution. RGBA outputs are composited onto a white background. CLIP-I and LPIPS compare corresponding views and are averaged over all asset–view pairs. UniTEX outputs are rendered with metallic set to zero and roughness to 0.5.

Both evaluation groups follow LumiTex (Bao et al., 2026) for metric implementations and preprocessing, with identical settings across methods.

C Runtime and Sampling Steps

Table 1 reports total runtime from reading the input GLB and constructing the scene to exporting the textured GLB and 2K PNG maps. The interval includes input preparation, conditioning, diffusion sampling, material decoding, and texture baking. Inference timing uses NVIDIA A100 SXM4 GPUs. All methods report mean per-asset runtime over the same 1,000 test assets. LumiTex includes the MR generation stage required for its PBR outputs.

Sampling steps. Figure 8 compares texture generation with 1, 4, 10, and 25 sampling steps at a fixed sequence length of 32K anchor tokens. One-step outputs capture the broad color layout, but oversaturate the blue armchair and green cloak and simplify surface color variation. Additional steps bring these colors closer to the references and recover more variation in the wooden furnishings and clothing. The overall appearance changes less between 10 and 25 steps in these two examples. Denoising time increases from 3.56 s at one step to 40.98 s at 25 steps (Table 3). This interval covers only diffusion sampling and excludes VAE decoding and texture baking.



Figure 8: Sampling-step ablation at 32K anchor tokens. Each row shows the reference image followed by generated textures using 1, 4, 10, and 25 sampling steps. Table 3 reports the denoising times.

Table 3: Mean DS-DiT denoising time over the same 1,000 evaluation assets at 32K anchor tokens, excluding VAE decoding and texture baking. The default 25-step setting is shown in bold.

Sampling steps	Denoising time (s)
1	3.56
4	8.03
10	17.00
25	**40.98**

D APPLICATIONS

We use the surface appearance representation for reference-guided material style transfer and adapt it to color-coded part segmentation.

D.1 MATERIAL STYLE TRANSFER

Material style transfer reuses the PBR checkpoint from the main generation experiments, changing the reference image to specify the desired appearance. We generate the original reference and its style-edited variants with GPT Image 2.5 (OpenAI, 2026), and apply each variant to the mesh constructed from the original reference. Figure 9 shows the resulting ceramic, leather, steampunk, and wood interpretations.



Figure 9: Reference-guided material style transfer. **All results share the same input geometry.** Reference images are grouped on the left. Each result shows albedo, metallic, and roughness from top to bottom beside the final rendering.

D.2 PART SEGMENTATION

For part segmentation, we train a separate model on 100K examples with color-coded part annotations, following the surface label-field formulation in native texture models (Lai et al., 2026b; Zeng et al., 2026). Conditioned on a single segmentation image, the model predicts a color field on the input mesh. We query this field at mesh vertices to obtain the vertex-color segmentations shown in Figure 10.



Figure 10: Part segmentation as a surface label field. Each of the four groups shows the asset's original appearance and two views of the predicted vertex-color segmentation. Colors distinguish parts within each asset.

E ADDITIONAL QUALITATIVE RESULTS

E.1 OPEN-SOURCE COMPARISONS

Figure 11 extends the main comparison with ten additional assets, spanning stylized characters, decorative sculptures, furniture, and a pottery workspace. We compare DSTex with LumiTex (Bao et al., 2026), TRELLIS.2 (Xiang et al., 2026), and UniTEX (Liang et al., 2026) using the same reference images and input meshes.

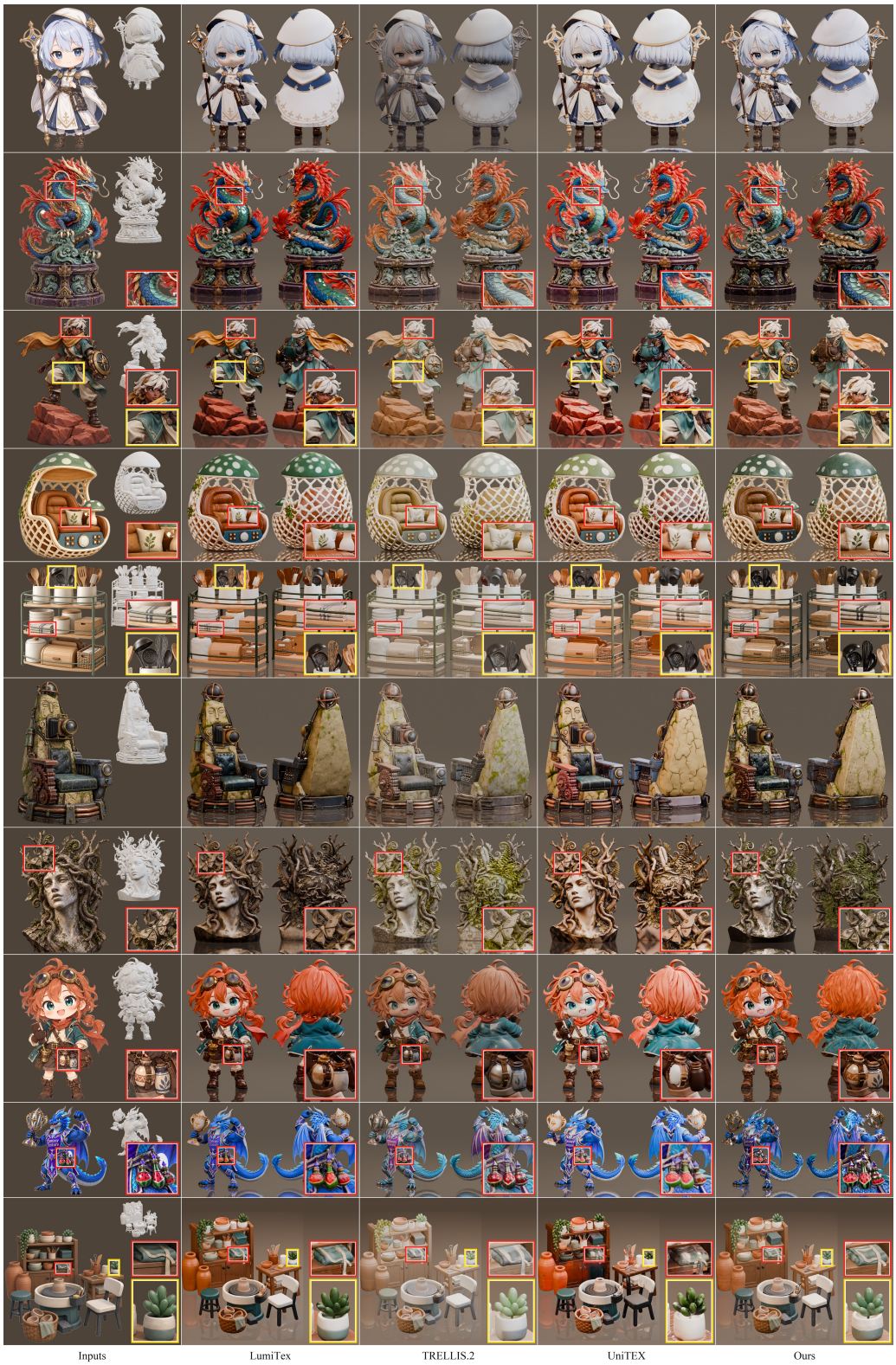


Figure 11: Additional qualitative comparisons with open-source texture generation methods. Columns show inputs, LumiTex, TRELLIS.2, UniTEX, and ours. Each input pairs a reference image with an untextured mesh; each method shows two views of the textured asset. Red and yellow boxes identify corresponding enlarged regions. Row references in Appendix E.1 count from top to bottom.

Fine decorative patterns (Figure 11). DSTex preserves small motifs that are simplified or blurred in the other results. In the adventurer example (row 8), the two bottles retain distinct painted designs: a light geometric band on the brown bottle and a blue leaf motif on the pale bottle. These designs are less distinct in LumiTex and UniTEX, while TRELLIS.2 produces a simplified bottle motif. The dragon character (row 9) provides a finer-scale example: our hanging ornaments retain the red-and-green watermelon pattern with individually visible dark seeds, whereas the competing results largely reduce the red regions to smooth patches.

Placement and separation of surface details. Our results also preserve where decorative elements lie relative to the underlying parts. The folded towels on the kitchen rack (row 5) retain distinct dark stripes on the fabric. In the pottery workspace (row 10), DSTex follows the reference's pale border across the folded teal cloth and its hanging end. UniTEX shows fragmented color patches in this region, while the border is less clearly defined in TRELLIS.2.

Surface appearance and detail across views. In the stone-and-metal chair (row 6), DSTex retains fine mottling and green weathering on the stone while distinguishing it from the dark upholstery and metallic fittings. The paired views of the character in row 1 also show blue-and-gold garment trim continuing onto the rear cape without an abrupt change in appearance.

E.2 Generation Gallery

Figure 12 illustrates how DSTex combines fine color detail with spatially varying material predictions across different assets. The humanoid robot retains small chest lettering and distinct painted panels, while the gown and spotted creature show decorative patterns across their surfaces. Beyond albedo, the metallic predictions distinguish the cowboy's mechanical fittings from its coat, and the chair's roughness map separates the stone body from the padded seat. The paired views also reveal details on the rear surfaces, including ornamentation on the gown and weathering on the chair, extending the textured appearance beyond the reference-facing regions. Figure 13 complements these individual-asset views with an assembled scene, shown alongside its geometry and PBR channels.



Figure 12: Additional PBR texture generation results from DSTex on ten input meshes. The Input column pairs a reference image with an untextured mesh. The remaining columns show two views of the rendered result and the predicted albedo, metallic, and roughness channels.



Figure 13: An assembled scene of assets textured by DSTex. The top panel shows the PBR rendering, including lettering on cargo crates, decorative surface patterns, and varied material appearances under scene lighting. The lower panels show the same scene and viewpoint as a clay rendering and albedo (middle row), followed by metallic and roughness (bottom row), each ordered left to right.