GLOSS: Geometric Local Self-Similarity Learning for Faithful Reference-Guided Texture Fill

CHENYUE CAI*, Princeton University, USA
ANITA HU, NVIDIA, Canada
JAMES LUCAS, NVIDIA, UK
SZYMON RUSINKIEWICZ, Princeton University, USA
MARIA SHUGRINA†, NVIDIA, Canada and University of Toronto, Canada

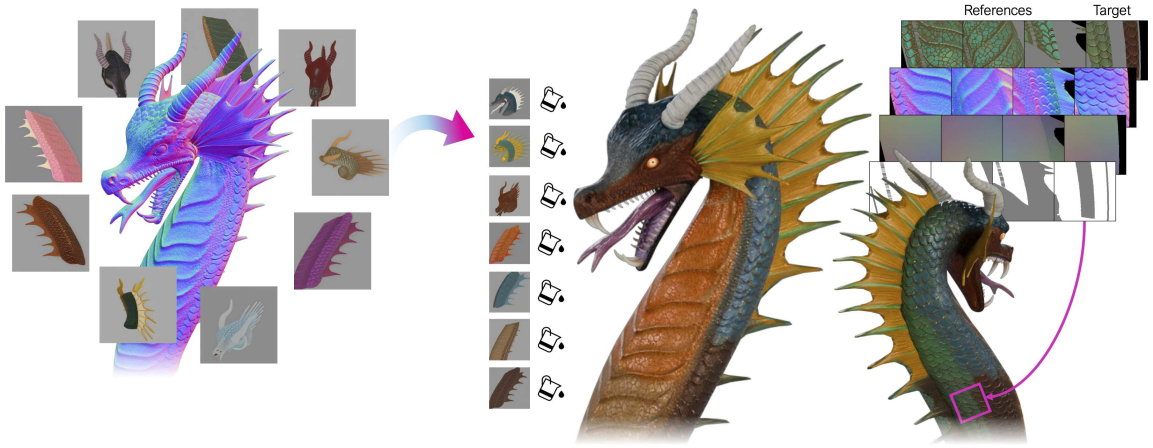


Fig. 1. We train a GLOSS network on many single-view generated images of one untextured 3D shape (left), allowing it to learn how to texture local shape regions by attending to local geometry (such as scales) and faithfully following conditional texture patches. Once trained, artists can use GLOSS to mix and match any number of references for interactive texture completion of the target shape (right), enabled through reference-guided local texture fill. We also show fully automated texturing, PBR channel completion, and transfer results across shapes.

Using conditional image generators, texture artists can explore many single-view looks for an existing 3D shape. Despite impressive progress, state-of-the-art generative methods still struggle to generate a full object texture while closely adhering to fine scale geometric detail and single view references, leaving little room for artist guidance. Furthermore, current automatic models lack the flexibility for artists to explore multiple textures from varied sources in an interactive and controllable manner. Unlike methods trained on large 3D datasets that generate full object textures from global guidance, our work explores a local and less data-hungry approach to texture with explicit artist control. We leverage the geometric self-similarity and geometry-texture correlation existing in many natural and man-made shapes, and train a shape-specific local texture generation and completion model. This model learns from existing image model priors and a single 3D shape, and is guided by attending to a set of geometry-aware reference patches. The trained shape-specific network can transfer any novel reference to the full target object texture through patchwise inpainting. We show improved or comparable quality to strong image-conditioned texture generation baselines, suggesting local texturing as a promising research direction. Our model also enables local geometry-conditioned texture inpainting, guided by artist-selected references, and generalizes to PBR materials and unseen meshes for texture transfer. We piloted our novel texture fill capability as a Blender add-on with several 3D texturing professionals who reported positive feedback on the model's controllability, practical usefulness, and creative affordances. We will release the code, the model checkpoint and Blender add-on on project page: https://chenyuecai.github.io/gloss-page/.

CCS Concepts: • **Computing methodologies** → **Texturing**; **Machine learning**; **Mesh models**.

Additional Key Words and Phrases: texture completion, texture synthesis, self-similarity, reference-guided generation, 3D shape texturing, diffusion models

1 INTRODUCTION

Recently, new workflows for texturing 3D shapes are emerging through the application of generative models, but the control and flexibility of existing approaches remain limited. Generative image models conditioned on normals and other channels (e.g. [Ye et al. 2023; Zhang et al. 2023]) and configurable pipelines like ComfyUI [Comfy-Org 2024] allow artists to explore diverse single-view renderings of a given 3D shape, respecting its geometric details (e.g., generated views in Fig. 1 follow the scales on the dragon skin). However, building consistent and faithful textures for whole meshes from such single-view references remains challenging. While recent texture generative models [Deng et al. 2024; Hunyuan3D et al. 2025; Xiang et al. 2025; Yu et al. 2024; Zhang et al. 2024a] achieve compelling results and often support single image conditioning, these methods are not designed to preserve local details of the reference. The global nature of these techniques precludes local artist

*Work done during an internship at NVIDIA.
†In support of more inclusive conference locations, this author removed herself from the SIGGRAPH Asia 2026 publication.

guidance or editing and limits overall texture resolution regardless of geometry scale. In contrast, techniques tackling local and artist-guided texture generation [Decatur et al. 2024; Hu et al. 2024] ignore the geometric details of the target geometry, making them less suitable for realistic use cases. Our work addresses reference-guided workflows in texturing, enabling greater local control and geometry-texture consistency, while circumventing the need for large-scale 3D datasets. We focus on showing promising results of artistic control and the underexplored small-data training, both highly relevant to production workflows, where custom models trained in-house on small task-specific data are common in startups and studios (e.g., InterPositive AI acquired by Netflix, Cuebric, Corridor Digital, Wonder Dynamics).

We introduce GLOSS, a technique that learns local texturing priors from a single shape's self-similarity and off-the-shelf image generative models. Once trained, the shape-specific texture generation network can automatically texture the target shape using any novel reference view by progressively synthesizing texture patches across the surface, attending to local geometric and appearance references. GLOSS enables novel interactive texturing workflows, wherein artists in-fill textures for any selected regions on the target shape, using any combination of references. For example, an artist might use one part of a reference view to texture the fins of a fish, but another for the body. Trained to inpaint, the local texturing model generates seamless and geometry-aware transitions between artist-specified regions. Thus, artists can edit or iteratively create textures. Moreover, unlike global texture generation techniques, patch-based texture generation allows our model to produce textures of any resolution. Preliminary results show GLOSS outperforms or matches strong generative baselines trained on large 3D data sets, both quantitatively and qualitatively. Indeed, 3D professionals using a prototype Blender add-on in a pilot study reported that GLOSS complements their existing workflows and could be used regularly.

Our work is inspired by classical patch-based synthesis, but with improved blending and natural variations shown by patch-based generative models [Hu et al. 2024]. Given an input shape M, we leverage image generative models to automatically create shape-specific training data. Despite their diversity, such individual views (Fig. 1) exhibit strong local self-similarity for many classes of shapes, including natural (plants, amphibians and reptiles, terrain) and man-made (pottery, objects with weathering effects, architecture). We leverage this insight to train a local texture inpainting model that learns to generate texture patches in one area, while attending to references from another area of the same single-view rendering using *batch multi-attention*. At run-time, this allows the model to closely follow multiple reference patches when generating and inpainting new textures. Unlike existing patch-based generative models, GLOSS generates textures conditioned on local geometry like bumps and wrinkles, and attends to such correlations in the provided references, ensuring faithful and controllable local texture generation. This enables our method to support previously impossible reference-guided texture-fill operations on the mesh. Fine-tuning for a similar class of objects can quickly adapt a model to new meshes.

To summarize, the contributions of our work include:

- a novel geometry-conditioned texture inpainting model with batch multi-attention that learns from geometry-texture self-similarity within a single 3D object.
- the design of a data-generation pipeline for training a local texturing model without any textured 3D models.
- support for novel interactive generative texture fill with *close adherence to reference patches* and *strong geometry-texture consistency*, with preliminary zero-shot transfer on unseen meshes and PBR materials.
- a patch-based mesh texture generation pipeline showcasing global consistency and competitive results against generative texturing models trained on large 3D datasets, demonstrating the promise of patch-based generative texturing.

Following related work (§2), we detail our local texturing model design and data generation in §3. In §4, we show applications for automatic and interactive texturing, and evaluate results in §5.

2 RELATED WORK

Synthesis via self-similarity. Classical texture synthesis methods on 3D surfaces blend patches in a coarse-to-fine manner while traversing the mesh geometry [Knöppel et al. 2015; Praun et al. 2000; Turk 2001]. Similarly, symmetries have been exploited in classical shape analysis [Harary et al. 2014; Pauly et al. 2008]. Recent approaches extend patch-based synthesis ideas into learning-based frameworks for 3D shape understanding, generation and reconstruction [Berkiten et al. 2017; Erler et al. 2020; Fogarty et al. 2025; Liang et al. 2022; Tretschk et al. 2020]. Point2Mesh [Hanocka et al. 2020], for example, employs mesh convolutions to capture local geometric self-similarity across the surface of a single shape. Similar ideas are explored in the image domain [Kulikov et al. 2023; Shaham et al. 2019; Shocher et al. 2018] and recent work [Mitchel et al. 2024] applied this to 3D shapes for texture transfer and editing. Inspired by these ideas, our work applies self-similarity to the joint distribution of local geometry and texture, learning an explicit self-similarity prior in the joint space between geometry and texture.

Texture generation via Pretrained Diffusion Models. Recent advances in 3D generation have used 2D diffusion models in place of large-scale 3D datasets. Score distillation sampling (SDS) [Poole et al. 2023; Wang et al. 2023] uses a frozen, pretrained diffusion model to score rendered geometry and update the texture representation. Subsequent work has improved visual fidelity [Lin et al. 2023; Metzer et al. 2023; Yi et al. 2024; Youwang et al. 2024], material fusing [Han et al. 2024], and multi-view consistency [Kwak et al. 2024; Ren et al. 2025; Voleti et al. 2024]. Alternatively, geometry, lighting and material conditioning have been provided explicitly [Chen et al. 2023a; Deng et al. 2024; Zhang et al. 2024a].

Other methods formulate texture synthesis as a view-conditioned inpainting task over the mesh, progressively generating textures across multiple views [Chen et al. 2023c; Richardson et al. 2023; Zeng et al. 2024a]. These methods offer faster inference than SDS-based pipelines, but often struggle with multi-view consistency and limited 3D awareness. Past work improved view sampling to enforce consistency [Cao et al. 2023], or used view-consistent noise sources and multi-view attention mechanisms [He et al. 2025; Liu et al. 2024].

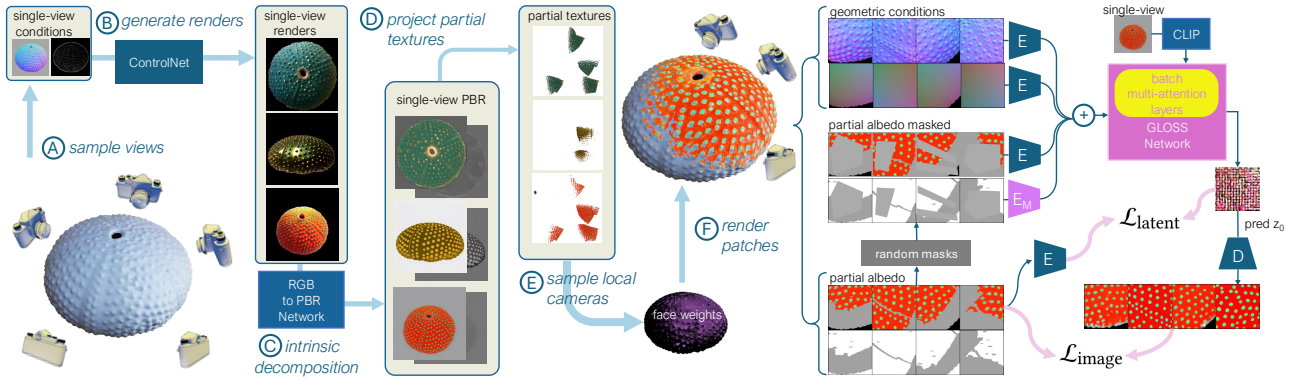


Fig. 2. **Data and Training**: we use off-the-shelf models to generate training data (§3.2) for our local texture inpainting model (§3.3, §3.4). Steps A to F show the patch data generation pipeline. Then we show how the generated data is used for model input as well as the training for the model.

Generative Texture Models Learned from 3D Datasets. Several approaches train texture generation models directly on textured meshes or multi-view renderings. Early approaches used texture fields to predict per-point color values directly from spatial coordinates [Oechsle et al. 2019]. Later methods adopted StyleGAN-inspired latent texture codes [Karras et al. 2019] to generate textures using either differentiable rendering [Chen et al. 2023b] or operating directly on surfaces [Bokhovkin et al. 2023; Siddiqui et al. 2022]. More recent methods focus on explicitly generating UV textures [Cheng et al. 2024; Yu et al. 2023]. All of these techniques generate category-specific textures, but their generalization to arbitrary shapes remains limited. Recent works further leverage large curated 3D datasets for multi-view geometry-conditioned diffusion training, scaling output UV map resolution and improving textural diversity [Bensadoun et al. 2024; Huang et al. 2025; Xiang et al. 2025; Yan et al. 2025; Yu et al. 2024; Yuan et al. 2025; Zeng et al. 2024b; Zhang et al. 2024b; Zhao et al. 2025]. While these methods produce high-quality texture on a variety of shapes, they rely on carefully curated and often proprietary 3D datasets for training. This data dependence limits their scalability and generalization to out-of-distribution shapes or rare object categories. In contrast, our method does not require large-scale 3D data and instead learns from the self-similarity within a single object, enabling high-fidelity texture synthesis without external supervision.

3 LOCAL TEXTURE INPAINTING MODEL

Our core idea is to train an image diffusion model $\mathcal{G}_M$ to complete textures in the local render space of a specific mesh M, conditioned on local geometry and paired geometry-texture references. We hypothesize that $\mathcal{G}_M$ can learn locally consistent texturing from only generated single-view renders of M, e.g. *given that these dragon scales have this texture, similar scales in another region should have a similar texture*. We describe our automatic data generation pipeline (§3.2), model (§3.3) and training details (§3.4). See §4 for inference.

3.1 Preliminaries

Our input is an untextured 3D mesh $M = \{F, V, U, T_N\}$ with faces F, vertices V, UV-map U mapping face vertices to texture coordinates, and an optional normals texture image T_N. For high-resolution shapes, geometric details may reside in F, V, or be baked into T_N, and we accept either. Our goal is to develop a model that can fill the missing albedo texture image T_ρ, and define T_α as the alpha (or known) part of T_ρ. While we focus on albedo, we also define T_{mat} as the optional multi-channel image of other material properties like roughness in the standard PBR model [Burley and Studios 2012].

To transition between texture and render spaces, we define several functions for forward rendering and backprojection:

$$I_N, I_{\text{geo}} \leftarrow \mathcal{R}_g(M, c) \tag{1a}$$

$$I_\rho, I_\alpha, I_{mat} \leftarrow \mathcal{R}_{mat}(M, c, T_\rho, T_\alpha, T_{mat}, W) \tag{1b}$$

$$T'_\rho, T'_\alpha, T'_{mat} \leftarrow \mathcal{R}^{-1}_{mat}(M, c, I_\rho, I_{mat}, W') \tag{1c}$$

where the *geometric* rendering $\mathcal{R}_g$ rasterizes M from viewpoint c at resolution W, rendering world-space positions I_{geo} and texture-mapping with T_N to produce a normals image I_N (e.g. Fig. 1, left). If no T_N is provided, only geometric normals are used, and in either case I_N is converted to *camera-relative coordinate frame*, important for network generalization later. The material rendering $\mathcal{R}_{mat}$ rasterizes and texture-maps M, producing albedo image I_ρ, alpha map I_α (texture-mapping of T_α) and an optional multi-channel image of other material properties T_{mat}. We also define an inverse function denoted $\mathcal{R}^{-1}_{mat}$ that back-projects renderings I_ρ, I_{mat} from camera c, back into partial textures of resolution W', producing the back-projected albedo texture T'_ρ, T'_α denoting areas that received a projection and optional materials T'_{mat}. In practice, all functions share an implementation and are only split here for clarity. We use a similar approach to [Hu et al. 2024]; please refer to it for details.

3.2 Training Data Generation

Our training data consists of paired local geometry patches and local texture patches. Such data is generated automatically using an ensemble of off-the-shelf image and language models to obtain diverse textures conditioned on mesh surface geometry. We begin by sampling n *global* cameras $\mathbf{C}^{(M)} = \{c_1 ... c_n\}$ around M (Fig. 2Ⓐ) and render geometric inputs $I^i_N, I^i_{\text{geo}} \leftarrow \mathcal{R}_g(M, c_i)$ (Eq. 1a). To obtain diverse and visually rich textures, we use an LLM to generate descriptive prompts for variations in color, material, and texture details. Using an off-the-shelf diffusion model with multiple ControlNets [Zhang et al. 2023], we generate single-view images $\tilde{I}^i ... \tilde{I}^n$

(Fig. 2Ⓑ) following the geometry and prompts. We then apply an off-the-shelf de-lighting network to convert each $\tilde{I}^i$ to albedo $\tilde{I}^i_\rho$ and materials $\tilde{I}^i_{mat}$ images (Fig. 2Ⓒ). We leave more prompt and single view generation details in Appendix A.

As shown in Fig. 2, even one albedo view $\tilde{I}^i$ can contain many local self-similarities. To learn from these without suffering from distortions in the rendered view, we opt to re-render M using partial textures from many close-up views. We backproject each $\tilde{I}^i_\rho$ using $\mathcal{R}^{-1}_{mat}$ (Eq. 1c) to obtain a partial texture map $\tilde{T}^i_\rho$ and alpha $\tilde{T}^i_\alpha$ for the visible area (Fig. 2Ⓓ). We use a heuristic to compute sampling face weights ω_i, prioritizing faces by both area and number of visible pixels in $\tilde{I}^i_\rho$. During training, each batch is confined to one view c_i, and its corresponding texture $\tilde{I}^i_\rho$ and ω_i are used to sample local cameras $\hat{c}_1 ... \hat{c}_b$ that fall within the known texture region of the single view c_i (Fig. 2Ⓔ). For each $\hat{c}_j$, we importance sample a face $f \in F$, get camera *at* vector by sampling a point p within f, set camera *up* vector as $v - p$ for a random vertex v of f, sample distance d along the normal $\vec{n}$ direction at p and get camera position as $p + d\vec{n}$, and finally sample a field of view within a range. By scaling M to a unit cube, we can share camera settings across all experiments. Local renderings from these views are used to train our model.

3.3 Model Architecture

Our image diffusion model $\mathcal{G}_M$ completes texture in the *local render space* of M. As shown in Fig. 2, the input is a batch of albedo patches $\mathbf{I}_\rho$ with masks $\mathbf{I}_\alpha$ rendered from local cameras, corresponding normal $\mathbf{I}_N$ and position $\mathbf{I}_{\text{geo}}$ renderings (we use normalized world positions relative to the center pixel to denote orientation relative to the object, but not exact location). The objective of $\mathcal{G}_M$ is to output $\hat{\mathbf{I}}_\rho$, inpainting missing albedo, while attending to $\mathbf{I}_N, \mathbf{I}_{\text{geo}}$ and aligned geometry-texture references within the batch.

As our backbone, we adopt Stable Diffusion v2.1 unCLIP [Ramesh et al. 2022], a latent diffusion variant accepting a CLIP image embedding in addition to the text prompt. For this, we use a full reference albedo view ($\tilde{I}^i_\rho$) to provide loose global context in the cross-attention layers, and pass an empty string for the text prompt. To accommodate additional input conditions, we expand the UNet [Ronneberger et al. 2015] with extra input channels, which are zero-initialized. The 3-channel $\mathbf{I}_N, \mathbf{I}_{\text{geo}}, \mathbf{I}_\rho$ are all encoded with the default pretrained image encoder E for our backbone, and the inpainting mask is downsampled to the latent space size. All conditioning channels are concatenated with the noisy latent input along the channel dimension and passed into the UNet.

Batch Multi-Attention. To capture the long-range self-similarity across the mesh M, we extend the standard self-attention mechanism to operate across all patches within a batch. This allows the model to leverage similarities beyond the local neighborhood of any patch, effectively enabling texture inpainting given reference patches from any part of the mesh or other sources at run-time.

We implement batch multi-attention across the spatial positions of all image patches in the batch. Let $X \in \mathbb{R}^{B\times C\times H\times W}$ denote a batch of feature maps with height H, width W, and C channels. We first flatten the spatial dimensions and stack the batch to obtain:

$$X' = \text{reshape}(X) \in \mathbb{R}^{(B\cdot H\cdot W)\times C}$$

We then compute full self-attention across all spatial positions in the batch:

$$\text{Attn}(X') = \text{softmax}\left(\frac{QK^\top}{\sqrt{C}}\right)V$$

where

$$Q = X'W_Q, \quad K = X'W_K, \quad V = X'W_V, \quad W_Q, W_K, W_V \in \mathbb{R}^{C\times C}$$

This formulation enables each pixel in a patch to attend to all other pixels across the batch, allowing the model to leverage long-range structural repetition and appearance cues across the object surface.

During training, we sample a set of local patches from the same textured mesh and treat them as a single attention context as illustrated in Fig. 2. The choice of batch size now imparts a quadratic computational complexity on model training. While a larger batch size increases the likelihood that the model can attend to geometrically similar regions across patches, this benefit comes with significantly higher memory usage during attention computation and increased training time cost per batch. This can be alleviated by implementing batch multi-attention only across subgroups of the patch-level batch. In practice, we train the model with a batch size of 8 and allow attention across the entire batch. Once trained, the attention mechanism generalizes beyond the fixed training batch size and can operate on arbitrary batch sizes during inference, providing flexibility for downstream applications.

3.4 Model Training

Training Batches. For each batch we sample a partial texture $\tilde{I}^i_\rho$ with alpha $\tilde{I}^i_\alpha$ and local cameras $\hat{c}_1 ... \hat{c}_b$ sampled for that view. We next render both geometric and material images (Fig. 2Ⓕ):

$$\mathbf{I}_N, \mathbf{I}_{\text{geo}} \leftarrow \mathcal{R}_g(M, \hat{c}_1 ... \hat{c}_b) \tag{2a}$$

$$\mathbf{I}_\rho, \mathbf{I}_\alpha \leftarrow \mathcal{R}_{mat}(M, \hat{c}_1 ... \hat{c}_b, \tilde{I}^i_\rho, \tilde{I}^i_\alpha, w) \tag{2b}$$

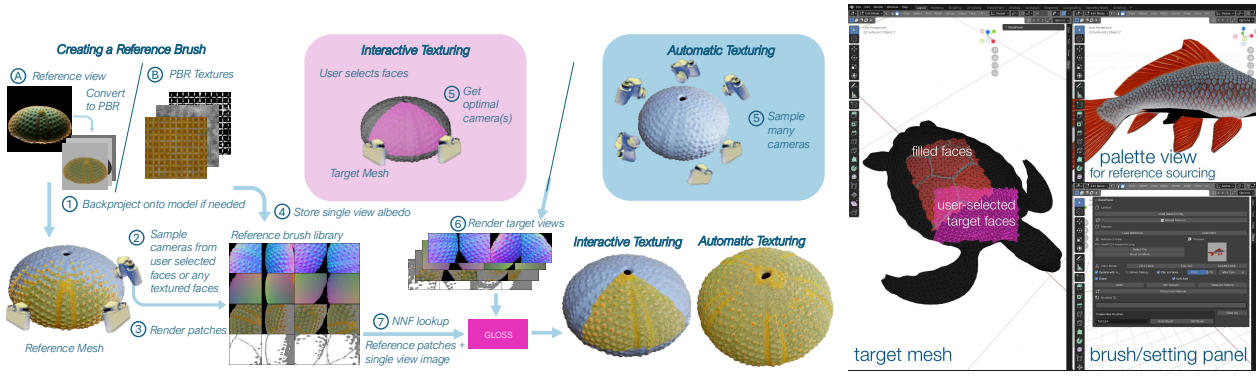
where $\mathbf{I}_\rho$ is the ground truth with only unmasked known areas. To train, we apply additional masks to $\mathbf{I}_\rho$ and $\mathbf{I}_\alpha$ using a combination of full masks and irregular masks from LaMa [Suvorov et al. 2021].

Diffusion Training. During training, we follow the standard denoising diffusion of Stable Diffusion v2.1 [Rombach et al. 2022]. Given a ground truth albedo latent z_0, we randomly sample a timestep $t \sim \mathcal{U}(1, T)$ and compute noisy z_t using the pre-defined DDPM scheduler. We train with v-prediction [Salimans and Ho 2022], restricting the loss to the masked regions in the latent space:

$$\mathcal{L}_{\text{latent}} = \|M_D \cdot (\hat{v}_t - v_t)\|_2^2, \tag{3}$$

where M_D is the downsampled mask $\mathbf{I}_\alpha$, $\hat{v}_t$ is the model output on z_t, and v_t is the target velocity. See Appendices B and C for more model and training details.

Masking Details. The downsampled mask does not fully prevent unknown pixels in the albedo channel from influencing the loss computation through convolutional kernels. This unintended information leakage, where masked (unknown) pixels affect neighboring activations, leads to a "fuzzy" supervision signal that often fails to preserve fine-grained texture details. To mitigate this, we apply two strategies. First, we replace the unknown pixels in the input albedo with the average color of the known texture regions, reducing the impact of unknown texture values during encoding. Second, we



(a) Brush construction and inference applications (b) Blender UI prototype

Fig. 3. **Model Inference Applications:** Reference brush creation from (A) a single view image or (B) any PBR texture map. The model can be used interactively to complete select faces, or fully automatically to complete the whole mesh.

introduce an auxiliary loss in image space. For near-zero diffusion timesteps (e.g., $t < 10$), we decode the predicted latent $\hat{z}_0$ back to the image space using the pretrained decoder D, and compute the LPIPS [Zhang et al. 2018] loss between the masked prediction and the masked ground truth albedo:

$$\mathcal{L}_{\text{image}} = LPIPS(\mathbf{I}_\alpha \cdot \hat{\mathbf{I}}_\rho, \mathbf{I}_\alpha \cdot \mathbf{I}_\rho), \tag{4}$$

where $\hat{\mathbf{I}}_\rho = D(\hat{z}_0)$. This perceptual loss complements the latent-space supervision, encouraging both structural coherence and high-frequency detail in the synthesized textures.

4 TEXTURING APPLICATIONS

Once trained, $\mathcal{G}_M$ opens many applications for texturing the source mesh M. We explored interactive texture fill and editing (§4.2), automatic texture generation (§4.4), and extension to PBR material authoring (§4.3), all guided by patch-level references (§4.1).

4.1 Reference Brushes

The batch multi-attention design (§3.3) enables fine-grained control over output texture via input patches $\mathbf{I}_\rho$, allowing the model to retrieve relevant context from known regions. This enables us to construct reference "brushes" during inference. Given target patches with empty or partial albedo $\mathbf{I}^t = (\mathbf{I}^t_\rho, \mathbf{I}^t_N, \mathbf{I}^t_{\text{geo}}, \mathbf{I}^t_\alpha)$ and reference patches $\mathbf{I}^* = (\mathbf{I}^*_\rho, \mathbf{I}^*_N, \mathbf{I}^*_{\text{geo}}, \mathbf{I}^*_\alpha)$, we jointly process $\mathbf{I}^t, \mathbf{I}^*$ with $\mathcal{G}_M$ in one batch and retain only predictions for $\mathbf{I}^t$. However, different regions may require different references due to geometric differences (e.g. fish fins vs. scales). Thus, we construct a reference library (we call "brush") $\mathcal{L}$ containing texture-geometry patch sets $(I^*_\rho, I^*_N, I^*_{\text{geo}}, I^*_\alpha)$ for a *single particular texture*. For each target batch, we select references via Nearest Neighbor Feature Matching (NNFM) [Zhang et al. 2022] on VGG features of normal renderings, and filter out candidates with large color mismatch to existing target albedo.

To build a brush $\mathcal{L}$, users can explore **single-view looks** for M with generative models following our data-generation pipeline (§3.2), creating references through patch rendering. We experiment with two modes for sampling $\mathcal{L}$. In automatic mode, we sample from known texture areas similarly to training, relying on NNFM to select fitting references. In user-guided mode, the user can restrict face sampling to selected areas of interest. Our interactive prototype implements both modes. We experiment with other ways to create $\mathcal{L}$, without a single view rendering of M. For example, we apply a **PBR material** to a square mesh, and sample reference patches from it in a similar way, leveraging aligned normals and albedo textures to create meaningful references (Fig. 3a Ⓑ). Alternatively, we can use existing techniques such as [Abu Alhaija et al. 2023; Liang et al. 2025a; Zeng et al. 2024c] to convert **any image** into PBR material maps or images with depth, albedo and normals, containing enough information to render references. Finally, we can use a **single view or existing texture of another mesh** M' to sample reference patches for texture transfer. In all cases, we simply provide a single CLIP-encoded albedo for the unCLIP image embedding (§3.3). See Video for examples of all the above brush construction strategies.

4.2 Interactive Texture Fill

GLOSS enables novel interactive texturing applications using $\mathcal{G}_M$, including reference-guided texture fill, supporting seamless inpainting and blending of different textures. This workflow allows 3D artists to more easily integrate references into the interactive texturing workflow. Given any single view $\tilde{I}$ of interest, we construct a reference library $\mathcal{L}$ using the strategy in §4.1, and then apply $\mathcal{G}_M$ by sampling local views $\hat{c}_i$ on M, inpainting them and back-projecting inpainted albedo patches I^*_ρ into the texture map T_ρ using Eq. 1c while using T_α for blending with texture already present. Given any reference brush $\mathcal{L}$ constructed in §4.1, the artist can specify a set of target faces to be filled. Using pre-computed local cameras and face-camera visibility matrix, we greedily sample local views $\hat{c}_i$ on M covering the target area and generate target patches. We implement additional camera sampling in areas where precomputed local cameras fail to cover due to complex mesh topology or occlusion (e.g. small faces in between the dragon scalp and horn). We enable both small area and large area filling by creating multiple target cameras. Naively filling patches iteratively on a large selected area where multiple cameras are sampled will cause fine variation and

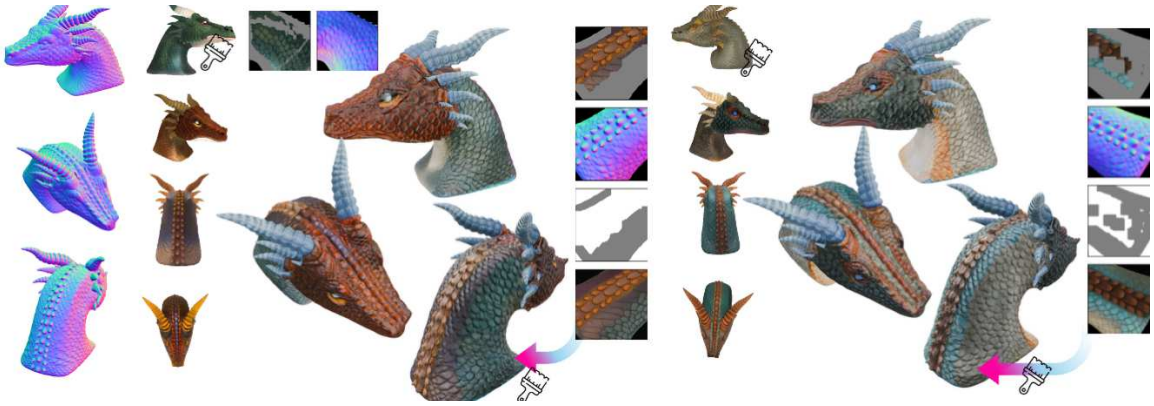


Fig. 4. **Interactive texturing** with multiple source single views on various parts of the object. We show the reference texture patch from the single view along with the normal, and where the texture brush created with that reference was applied to inpaint on the existing painted texture.

inconsistency due to the generative nature of the model. To improve consistency, we first apply a synchronized diffusion strategy [Liu et al. 2024] for the first 8/20 diffusion steps. Then, we add noise for 18/20 diffusion steps to this globally consistent yet over-smoothed texture and denoise from there. Using the same views $\hat{c}_i$, we add noise to the per-camera rendering of this guidance texture, and run the remaining diffusion steps in render space. The user can mix-and-match multiple references to texture from scratch or edit an existing or automatically generated texture of M. We implement all of these capabilities into our Blender prototype (Fig. 3b).

4.3 PBR Material Fill and Transfer

Although $\mathcal{G}_M$ is only trained to inpaint albedo, preliminary results reveal that it generalizes to other PBR channels, such as metallic and roughness concatenated with zeros into an RGB image. We use the strategies in §4.1 to create brushes $\mathcal{L}$ containing material, not albedo, references. This extends prior applications to PBR material generation and interactive editing as well. Materials and albedo are painted independently, but both are conditioned on underlying geometry, which results in partial but not full consistency.

4.4 Automatic Texture Generation

GLOSS also enables automatic texture generation (Fig. 3a), based on a novel single view $\tilde{I}$ of M using $\mathcal{G}_M$. For the automatic setting, we sample batches of cameras over the entire mesh greedily, selecting those that contain at least 20% overlap with fully completed texture (for consistency) and would update the most texture. We construct batches of 8 target and 8 reference patches. We also enforce non-overlapping patches by applying rejection sampling, ensuring that independently filled patches within a batch do not conflict with each other. We also apply the synchronized diffusion strategy to enforce global consistency. Inpainted patch albedo is back-projected into the still empty areas of the texture T_ρ, updating T_α accordingly. We post-process the texture with dilation to fill the small seams due to the imprecision of backprojection.

5 RESULTS AND EVALUATION

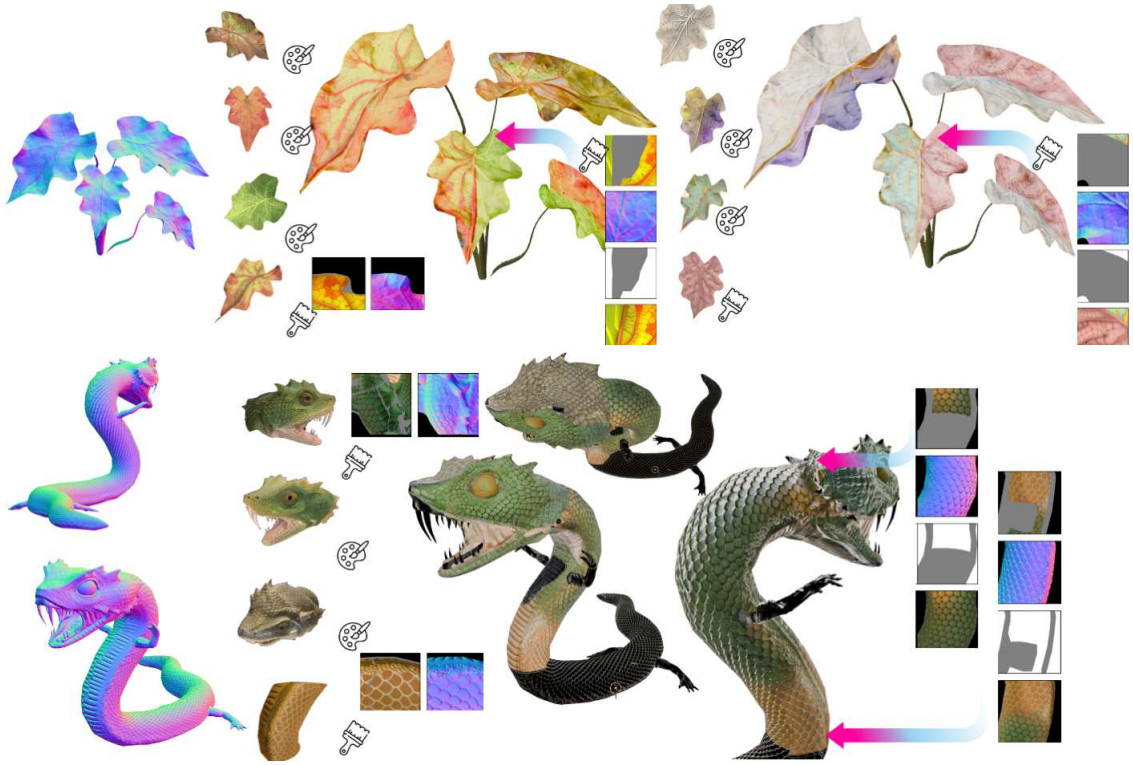
In §5.2, we show novel interactive capabilities, texture transfer and generalization results, and a pilot user study. We evaluate our method on automatic texture generation in §5.3, with competitive performance against strong generative baselines. We present ablations in §5.4. We include more evaluation details, experimental result and attention analysis in Appendix D.

5.1 Experimental Settings

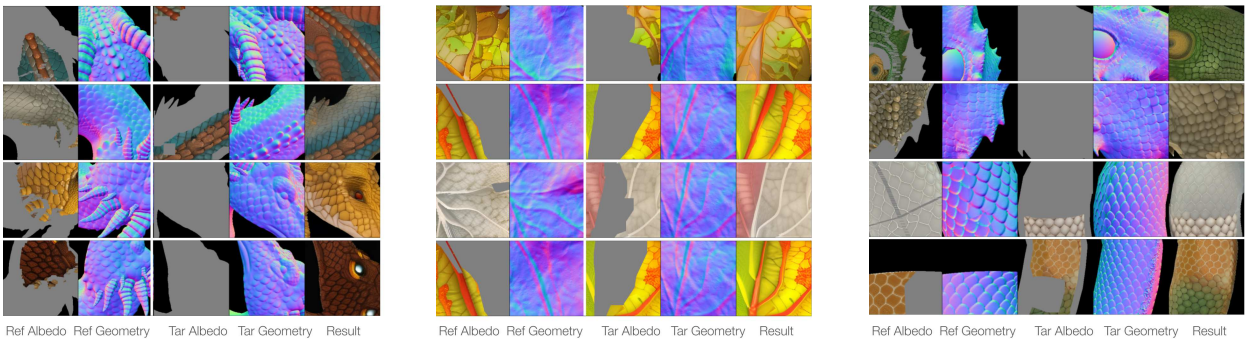
We select the following baselines. **TEXGen** [Yu et al. 2024], a representative feed-forward method working directly in texture UV space, **Hunyuan 2.1** (Hunyuan3D-Paint model) [Hunyuan3D et al. 2025], a state-of-the-art method predicting multi-view consistent albedo renderings which are fused into the final 3D shape texture, and **Paint3D** [Zeng et al. 2024a], a coarse-to-fine approach combining both multi-view diffusion and learning in UV space. **TRELLIS 2** [Xiang et al. 2025] is a state-of-the-art method that leverages a structured sparse 3D latent space for image to 3D generation. **MV-Adapter** [Huang et al. 2025] is a consistent multi-view generation method that supports texture generation from an image. All of these models support image conditioned texture generation. For the from-scratch experiment setting, we use AdamW optimizer with learning rate $1e-4$ with batch size of 32 for 80k steps and EMA decay of 0.999. For the finetune experiment setting, we use 100 single views, batch size 16, and 20k steps. We use the finetuned model for interactive applications.

5.2 Interactive Applications

We demonstrate interactive affordances of GLOSS for artist-guided fill, inpainting and blending (§4.2) of textures using reference brushes constructed using varied methods §4.1, compare with another generative interactive technique (§5.2.3) and pilot our prototype with several 3D professionals (§5.3.3). We show that with GLOSS, artists can interactively paint, and can create texture with close adherence to the reference texture and to the underlying geometry in Fig. 5a and Fig. 5b. We compare our method to other automatic methods, which fail to respond to the artist intent of interaction, and produce bad alignment to reference and geometry in Fig. 5c.



(a) Interactive texture completion by combining multiple texture in single views. In the lizard example, one can source texture from the head and transfer the texture onto the body.



(b) Texture filling given various reference source and geometry condition.

(c) Comparison to automatic generation methods.

Fig. 5. **Interactive texturing**: our model supports faithful and controllable texture fills from multiple sources.

Method	Per-mesh				Fine-tune			
	LPIPS↓	FID↓	DS↓	CMMD↓	LPIPS↓	FID↓	DS↓	CMMD↓
TexGen	-	106.917	0.436	0.883	-	124.460	0.538	1.143
MV-Adapter	0.358	76.98	0.370	0.666	0.366	122.865	0.473	0.961
Hunyuan2.1	0.383	**73.702**	0.437	0.494	0.299	**93.045**	0.356	**0.570**
Trellis2	0.356	102.22	0.411	0.545	0.307	110.773	0.371	0.777
Ours	**0.302**	80.258	**0.344**	**0.478**	**0.282**	104.108	**0.352**	0.694

Table 1. **Quantitative Results:** Overall (DreamSim) and patch-level (LPIPS, FID, CMMD) metrics on automatic texture generation for per-mesh and fine-tuned models. Best in bold.

5.2.1 Interactive Texture Fill. Many references are used in an interactive workflow to generate the final result. Using our interface, artists fill target areas of the mesh using references generated via diverse methods. For example, in Fig. 1, single-view references of the dragon were used to paint textures that are geometrically consistent with its surface. We show more interactive examples in Fig. 5. In both the dragon head and foliage examples, ours is faithful to reference texture and geometry, while other methods show saturated color and fail to align with local dragon scales in Fig. 5c. Further, we show that we can source brushes from images and PBR materials in Fig. 6d, and show preliminary results of $\mathcal{G}_M$ transferring to PBR material painting. For example, in Fig. 6d, painting the urchin with references from the waffled metal PBR material causes the distressed material artifacts to appear on the sea urchin bumps by attending to normal map similarities. This achieves a much more natural texture transfer than naïve application of the material.

5.2.2 Generalization. While the data generation and pre-training of $\mathcal{G}_M$ for a specific mesh M are automated, such pre-processing limits the practical application of our method. To probe our model's capacity for generalization, we apply it *zero-shot* to novel meshes. We show that a model $\mathcal{G}_M$ trained on M can also texture other similar shapes in Fig. 6b. While these are preliminary results, future work could extend the generalization of our method across shapes.

5.2.3 Comparison with Diffusion Texture Painting. Recent work, Diffusion Texture Painting (DTP) [Hu et al. 2024], demonstrated the utility of generative inpainting models for local texture painting on meshes, a similar setting. For progressively filling strokes one camera at a time (original DTP setting), both methods produce seamless strokes. In Fig. 6c left, DTP ignores geometry and produces only minor variations from the single reference patch, while our method adapts multiple references to match geometric details (see rendered strokes for texture/geometry misalignment). This allows DTP to paint with references that have no relation to underlying geometry (rocks over scales), whereas our method produces geometrically consistent texture transfer (rocks turn into scales), making the two approaches useful for orthogonal use cases. For texture fill with unordered local cameras (Fig. 6c, right), DTP produces chaotic results due to its reliance on stroke orientation, while our method uses local geometry to create consistent fills.

5.3 Automatic Texture Generation

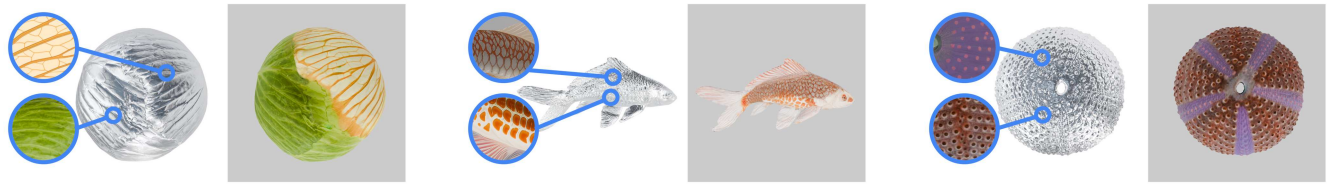
We showed that our model is powerful for local texture completion. Because this modality is difficult to compare quantitatively, we also compare the per-mesh and fine-tuned models we trained against large scale generative models on the full texture generation task. For our selected meshes, our data-limited texture model still shows competitive behavior.

5.3.1 Task and Metrics. We evaluate our model on from-scratch and finetune settings over 10 base meshes and 9 transfer meshes. To assess overall consistency and adherence to $\tilde{I}$, we render each mesh from the opposite side of the test view and compute **DreamSim** [Fu et al. 2023], a perceptual metric aligned with human visual similarity. For local texture quality, we estimate distributional metrics per-shape and per-view between these two patch samples using Fréchet Inception Distance (**FID**) [Heusel et al. 2017] and CLIP-based Maximum Mean Discrepancy (**CMMD**) [Jayasumana et al. 2024], a more recent metric that is robust for assessing image quality. Finally, we also compute **LPIPS** [Zhang et al. 2018] between patches I_ρ rendered from known texture region using original backprojected view and generated textures, respectively. This metric is *omitted for TEXGen*, where the texture in this region is the same as the back-projection.

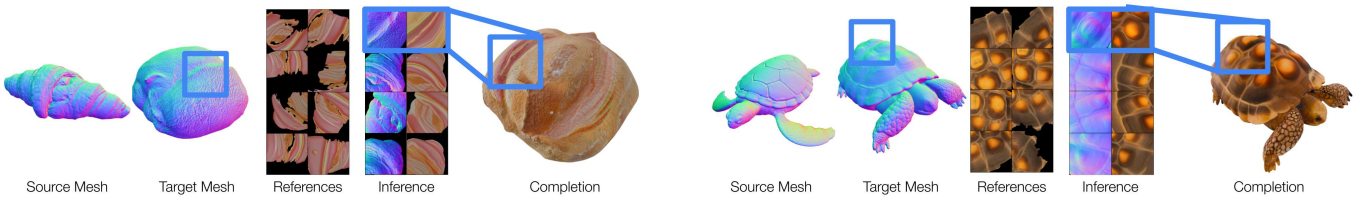
5.3.2 Results. Quantitative results in Tb. 1 reveal that our method achieves the best overall score in 3/4 metrics. Our method remains competitive when transferring to 9 other meshes with shorter finetuning steps, suggesting that a single texturing model could be learned for a class of shapes for more practical artist use cases. We report full per shape metrics and visual comparisons in Supplement. Qualitative results in Fig. 8 (and video) likewise show competitive performance against state-of-the-art methods. Our method has limited generalization across shapes but shows the surprising power of local reference-guided texture generation. While GLOSS lacks global context during generation and sometimes has trouble enforcing globally consistent texturing, its reference-guided design ensures close adherence to the target $\tilde{I}_\rho$, which is critical for user-guided applications.

5.3.3 User Study. To get early feedback on the interactive potential of GLOSS (§4.1, §4.2), we piloted our Blender prototype (Fig. 3b) with five professional 3D artists. Participants experimented with interactive fill and reference brush selection, and also tried fully automated texturing using the Hunyuan2.1 baseline. See Appendix G for more details.

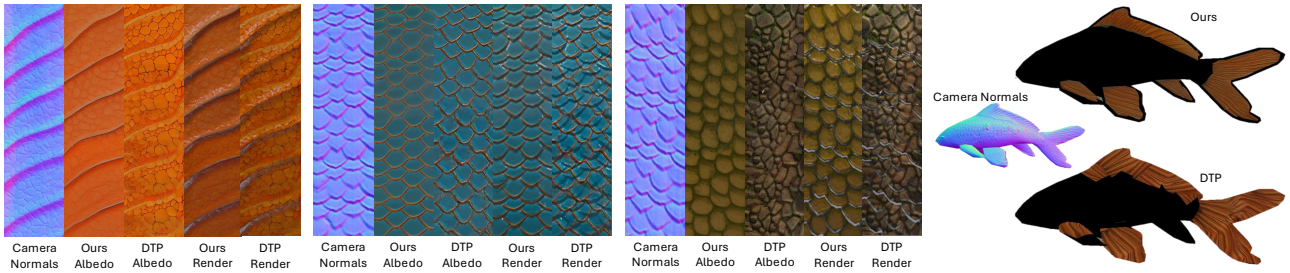
All participants acknowledged the value of our iterative workflow, with 4/5 artists reporting that it would make their work more efficient (2 strongly agree, 2 agree, 1 neutral on a 5-point Likert scale) and all reporting that they would integrate the interactive reference and geometry-guided fill feature into their texturing workflow (2 strongly agree, 3 agree). Artist 1 noted that it "opens up a lot of creative possibilities, like creatively texturing different areas from multiple texture style images" and found selecting reference brushes "very intuitive and convenient." Artist 2 remarked that the tool "blends well with [sic] multiple textures that [they] want to paint." After comparing with automatic Hunyuan2.1 texturing, all participants reported that automatic and interactive texturing approaches are complementary, quoting Artist 3: "the two tools could be combined to make a more powerful and flexible tool, giving creatives both a speed-up and greater creative and quality control." This suggests that the finer local and reference control of GLOSS complements existing texture generation solutions. Participants



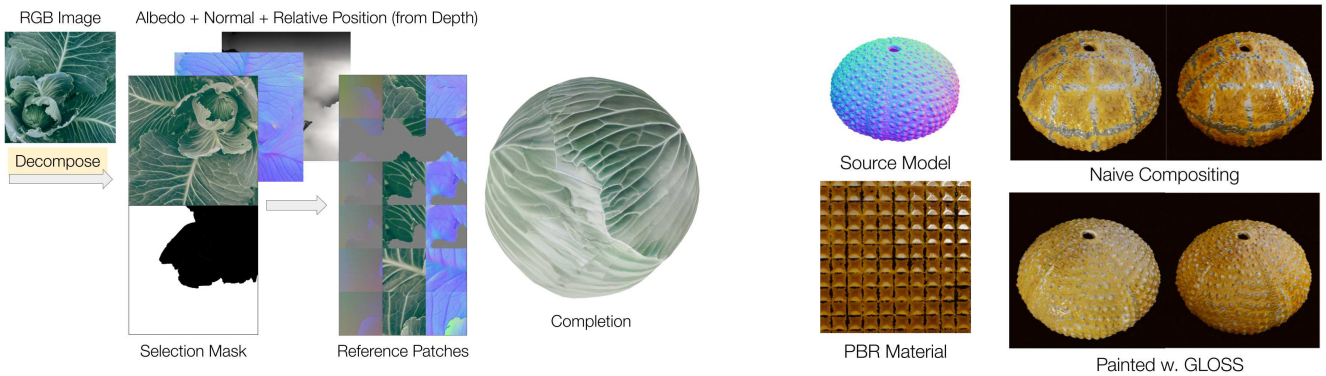
(a) **Blending** multiple textures seamlessly: results of interactive application with references shown in insets.



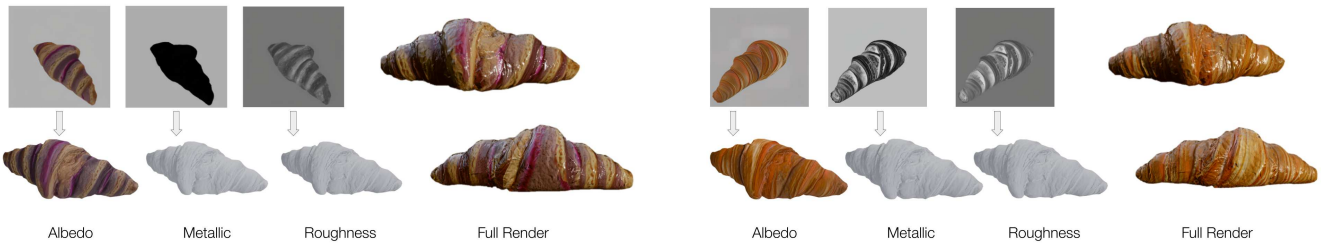
(b) **Generalization:** a model $\mathcal{G}_M$ trained on the mesh M (left), can be applied to texture similar unseen shape N (right), with references sourced from either, facilitating texture transfer and other applications (§5.2.2).



(c) **Comparison to Diffusion Texture Painting** on painting a stroke (left) and fill with random cameras (right) (see §5.2.3).



(d) **Brush Creation Methods:** generating a brush from an RGB image (left), and from a PBR material (right). See §4.1, §5.2.1.



(e) **PBR Material Completion:** GLOSS can generalize to metallic and roughness channels (also previous line, right). See §4.3, §5.2.1.

Fig. 6. **Interactive applications, transfer, comparisons**: our model supports a wide range of texturing applications.

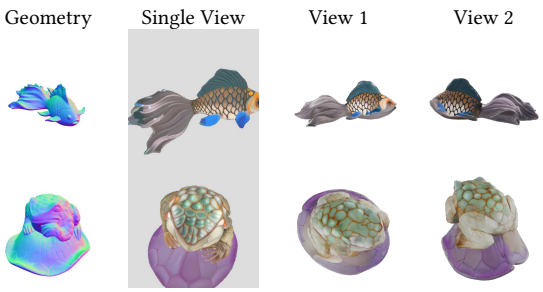


Fig. 7. **Automatic texturing** with fine-tuned models for the fish and reptile categories.

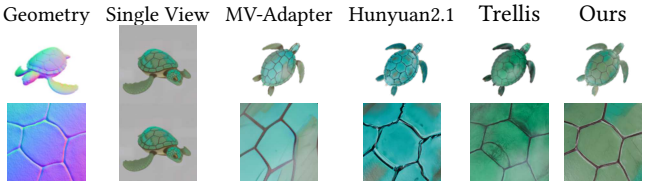


Fig. 8. An example of **Automatic texturing** visual comparisons.

Variant	Croissant		Koi Fish		Sea Shell		Overall	
	LPIPS↓	FID↓	LPIPS↓	FID↓	LPIPS↓	FID↓	LPIPS↓	FID↓
No batch-attn.	0.4277	100.93	0.2248	57.09	0.2923	81.06	0.3262	79.70
No geom. cond.	0.3838	64.62	0.2512	79.73	0.2810	90.48	0.3175	78.27
No img. loss	0.3006	62.31	0.2027	53.25	0.2628	**76.76**	0.2517	64.11
No fine-tuning	0.3020	59.85	0.2011	**51.77**	**0.2623**	78.53	0.2518	**63.38**
Full model	**0.2830**	**59.45**	**0.2009**	55.54	0.2673	77.11	**0.2420**	64.03

Table 2. **Ablations** on 3 shapes and a truncated training run using our validation set suggest that batch-attention and geometric conditioning have the greatest effect on local texture quality.

also identified key areas for improvement, including speed and PBR integration. We further iterated on the prototype by speeding up inference time and dilating the inpainting area to avoid model's over-inpainting over geometric cues.

5.4 Ablations

We conduct ablation studies to validate design choices of our model. In Tb. 2, we compare our proposed method with the following variations with the same metrics. **No batch-wise attention**: standard self-attention layers within $\mathcal{G}_M$. **No geometry conditioning**: No relative position and normal conditioning, $\mathbf{I}_{\text{geo}}$, which serves as a DTP-style baseline where the texture guidance formulation ignores local geometry. **No image loss**: The auxiliary image-space loss, $\mathcal{L}_{\text{image}}$, is omitted. **No fine-tuning**: The model is trained without initializing from the pretrained network weights. The batch-wise attention and geometry conditioning have the greatest effect on texture generation quality.

6 CONCLUSION

GLOSS integrates generative models into artist workflows while preserving control and flexibility. Our data pipeline and training strategy require no additional 3D data, yet maintain geometric consistency and high-quality outputs. We demonstrate that GLOSS is competitive with state-of-the-art automatic methods while also advancing the frontier of interactive texturing.

Limitations and Future Work. Operating in the small-data regime is a key strength, but it limits generalization and increases cost per shape. Future work could further develop GLOSS to improve scalability and generalization across shapes. Another limitation is that our model leverages a local geometry-texture joint prior; when there is little such correlation (e.g. semantic texture, weak geometry assets), our method would not be applicable. We present such failure cases in the supplemental material. We also acknowledge that PBR generation is preliminary as the given material channels are not jointly trained; thus consistency across channels is not guaranteed and remains an important direction for future work.

Acknowledgments

We would like to thank members of the PIXL group for their feedback on the project.

References

Hassan Abu Alhaija, James Lucas, Alexander Zook, Michael Babcock, David Tyner, Rajeev Rao, and Maria Shugrina. 2023. Interactive AI Material Generation and Editing in NVIDIA Omniverse. In *ACM SIGGRAPH 2023 Real-Time Live!* (Los Angeles, CA, USA) *(SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.1145/3588430.3597248

Raphael Bensadoun, Yanir Kleiman, Idan Azuri, Omri Harosh, Andrea Vedaldi, Natalia Neverova, and Oran Gafni. 2024. Meta 3d texturegen: Fast and consistent texture generation for 3d objects. *arXiv preprint arXiv:2407.02430* (2024).

Sema Berkiten, Maciej Halber, Justin Solomon, Chongyang Ma, Hao Li, and Szymon Rusinkiewicz. 2017. Learning detail transfer based on geometric features. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 361–373.

Alexey Bokhovkin, Shubham Tulsiani, and Angela Dai. 2023. Mesh2Tex: Generating Mesh Textures from Image Queries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 1–10. https://doi.org/10.1109/ICCV.2023.00001

Brent Burley and Walt Disney Animation Studios. 2012. Physically-based shading at disney. In *ACM SIGGRAPH*, Vol. 2012. ACM New York, NY, USA, 1–7.

Tianshi Cao, Karsten Kreis, Sanja Fidler, Nicholas Sharp, and Kangxue Yin. 2023. Texfusion: Synthesizing 3d textures with text-guided image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4169–4181.

Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. 2023c. Text2tex: Text-driven texture synthesis via diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*. 18558–18568.

Qimin Chen, Zhiqin Chen, Hang Zhou, and Hao Zhang. 2023b. ShaDDR: interactive example-based geometry and texture generation via 3D shape detailization and differentiable rendering. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.

Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. 2023a. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 22246–22256.

An-Chieh Cheng, Xueting Li, Sifei Liu, and Xiaolong Wang. 2024. TUVF: Learning Generalizable Texture UV Radiance Fields. *International Conference on Learning Representations* (2024).

Comfy-Org. 2024. ComfyUI. https://github.com/Comfy-Org/ComfyUI.

Blender Online Community. 2026. Blender – a 3D modelling and rendering package. https://www.blender.org.

Dale Decatur, Itai Lang, Kfir Aberman, and Rana Hanocka. 2024. 3d paintbrush: Local stylization of 3d shapes with cascaded score distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4473–4483.

Kangle Deng, Timothy Omernick, Alexander Weiss, Deva Ramanan, Jun-Yan Zhu, Tinghui Zhou, and Maneesh Agrawala. 2024. FlashTex: Fast Relightable Mesh Texturing with LightControlNet. In *European Conference on Computer Vision (ECCV)*.

Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, Niloy J Mitra, and Michael Wimmer. 2020. Points2surf learning implicit surfaces from point clouds. In *European Conference on Computer Vision*. Springer, 108–124.

Kyle Fogarty, Chenyue Cai, Jing Yang, Zhilin Guo, and Cengiz Öztireli. 2025. Self-Supervised Implicit Attention Priors for Point Cloud Reconstruction. *arXiv preprint arXiv:2511.04864* (2025).

Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. 2023. DreamSim: Learning New Dimensions of Human Visual Similarity using Synthetic Data. *Advances in Neural Information Processing Systems* 36 (2023), 50742–50768.

Junlin Han, Filippos Kokkinos, and Philip Torr. 2024. VFusion3D: Learning Scalable 3D Generative Models from Video Diffusion Models. *European Conference on Computer*

Vision (ECCV) (2024).
Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. 2020. Point2Mesh: a self-prior for deformable meshes. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 126–1.
Gur Harary, Ayellet Tal, and Eitan Grinspun. 2014. Context-based coherent surface completion. *ACM Transactions on Graphics (TOG)* 33, 1 (2014), 1–12.
Zebin He, Mingxin Yang, Shuhui Yang, Yixuan Tang, Tao Wang, Kaihao Zhang, Guanying Chen, Yuhong Liu, Jie Jiang, Chunchao Guo, et al. 2025. MaterialMVP: Illumination-Invariant Material Generation via Multi-view PBR Diffusion. *arXiv preprint arXiv:2503.10289* (2025).
Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).
Anita Hu, Nishkrit Desai, Hassan Abu Alhaija, Seung Wook Kim, and Maria Shugrina. 2024. Diffusion texture painting. In *ACM SIGGRAPH 2024 Conference Papers*. 1–12.
Zehuan Huang, Yuan-Chen Guo, Haoran Wang, Ran Yi, Lizhuang Ma, Yan-Pei Cao, and Lu Sheng. 2025. Mv-adapter: Multi-view consistent image generation made easy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 16377–16387.
Team Hunyuan3D, Shuhui Yang, Mingxin Yang, Yifei Feng, Xin Huang, Sheng Zhang, Zebin He, Di Luo, Haolin Liu, Yunfei Zhao, et al. 2025. Hunyuan3D 2.1: From Images to High-Fidelity 3D Assets with Production-Ready PBR Material. *arXiv preprint arXiv:2506.15442* (2025).
Sadeep Jayasumana, Srikumar Ramalingam, Andreas Veit, Daniel Glasner, Ayan Chakrabarti, and Sanjiv Kumar. 2024. Rethinking fid: Towards a better evaluation metric for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9307–9315.
Tero Karras, Samuli Laine, and Timo Aila. 2019. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4401–4410.
Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2015. Stripe patterns on surfaces. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.
Vladimir Kulikov, Shahar Yadin, Matan Kleiner, and Tomer Michaeli. 2023. Sinddm: A single image denoising diffusion model. In *International conference on machine learning*. PMLR, 17920–17930.
Jeong-gi Kwak, Erqun Dong, Yuhe Jin, Hanseok Ko, Shweta Mahajan, and Kwang Moo Yi. 2024. ViVid-1-to-3: Novel View Synthesis with Video Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 6775–6785.
Ruofan Liang, Zan Gojcic, Huan Ling, Jacob Munkberg, Jon Hasselgren, Zhi-Hao Lin, Jun Gao, Alexander Keller, Nandita Vijaykumar, Sanja Fidler, and Zian Wang. 2025a. DiffusionRenderer: Neural Inverse and Forward Rendering with Video Diffusion Models. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
Ruofan Liang, Zan Gojcic, Huan Ling, Jacob Munkberg, Jon Hasselgren, Zhi-Hao Lin, Jun Gao, Alexander Keller, Nandita Vijaykumar, Sanja Fidler, and Zian Wang. 2025b. DiffusionRenderer: Neural Inverse and Forward Rendering with Video Diffusion Models. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
Yaqian Liang, Shanshan Zhao, Baosheng Yu, Jing Zhang, and Fazhi He. 2022. Meshmae: Masked autoencoders for 3d mesh data analysis. In *European conference on computer vision*. Springer, 37–54.
Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. 2023. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 300–309.
Yuxin Liu, Minshan Xie, Hanyuan Liu, and Tien-Tsin Wong. 2024. Text-guided texturing by synchronized multi-view diffusion. In *SIGGRAPH Asia 2024 Conference Papers*. 1–11.
Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. 2023. Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12663–12673.
Thomas W Mitchel, Carlos Esteves, and Ameesh Makadia. 2024. Single mesh diffusion models with field latents for texture generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7953–7963.
Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. 2019. Texture Fields: Learning Texture Representations in Function Space. In *Proceedings IEEE International Conf. on Computer Vision (ICCV)*.
Mark Pauly, Niloy J Mitra, Johannes Wallner, Helmut Pottmann, and Leonidas J Guibas. 2008. Discovering structural regularity in 3D geometry. In *ACM SIGGRAPH 2008 papers*. ACM New York, NY, USA, 1–11.
Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2023. DreamFusion: Text-to-3D using 2D Diffusion. In *The Eleventh International Conference on Learning Representations*.
Emil Praun, Adam Finkelstein, and Hugues Hoppe. 2000. Lapped textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 465–470.
Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical Text-Conditional Image Generation with CLIP Latents. *arXiv preprint arXiv:2204.06125* (2022).
Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. 2025. GEN3C: 3D-Informed World-Consistent Video Generation with Precise Camera Control. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*. 6121–6132.
Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. 2023. Texture: Text-guided texturing of 3d shapes. In *ACM SIGGRAPH 2023 conference proceedings*. 1–11.
Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10684–10695.
Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 234–241.
Tim Salimans and Jonathan Ho. 2022. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512* (2022).
SG_161222. 2024. Realistic Vision. https://openlaboratory.ai/models/realistic-vision.
Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. 2019. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF international conference on computer vision*. 4570–4580.
Assaf Shocher, Nadav Cohen, and Michal Irani. 2018. "Zero-Shot" Super-Resolution Using Deep Internal Learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3118–3126.
Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. 2022. Texturify: Generating Textures on 3D Shape Surfaces. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 1–18. https://doi.org/10.1007/978-3-031-19803-8_1
Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. 2021. Resolution-robust Large Mask Inpainting with Fourier Convolutions. *arXiv preprint arXiv:2109.07161* (2021).
Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Carsten Stoll, and Christian Theobalt. 2020. Patchnets: Patch-based generalizable deep implicit 3d shape representations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*. Springer, 293–309.
Greg Turk. 2001. Texture synthesis on surfaces. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 347–354.
Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitrii Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. 2024. SV3D: Novel Multi-view Synthesis and 3D Generation from a Single Image using Latent Video Diffusion. In *European Conference on Computer Vision (ECCV)*.
Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. 2023. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12619–12629.
Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, et al. 2025. Native and compact structured latents for 3d generation. *arXiv preprint arXiv:2512.14692* (2025).
Dongyu Yan, Leyi Wu, Jiantao Lin, Luozhou Wang, Tianshuo Xu, Zhifei Chen, Zhen Yang, Lie Xu, Shunsi Zhang, and Yingcong Chen. 2025. FlexPainter: Flexible and Multi-View Consistent Texture Generation. *arXiv preprint arXiv:2506.02620* (2025).
Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. 2023. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721* (2023).
Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. 2024. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6796–6807.
Kim Youwang, Tae-Hyun Oh, and Gerard Pons-Moll. 2024. Paint-it: Text-to-texture synthesis via deep convolutional texture map optimization and physically-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4347–4356.
Xin Yu, Peng Dai, Wenbo Li, Lan Ma, Zhengzhe Liu, and Xiaojuan Qi. 2023. Texture Generation on 3D Meshes with Point-UV Diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 4206–4216.
Xin Yu, Ze Yuan, Yuan-Chen Guo, Ying-Tian Liu, Jianhui Liu, Yangguang Li, Yan-Pei Cao, Ding Liang, and Xiaojuan Qi. 2024. Texgen: a generative diffusion model for mesh textures. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–14.
Ze Yuan, Xin Yu, Yangtian Sun, Yuan-Chen Guo, Yan-Pei Cao, Ding Liang, and Xiaojuan Qi. 2025. SeqTex: Generate Mesh Textures in Video Sequence. *arXiv preprint arXiv:2507.04285* (2025).
Zongsheng Yue, Kang Liao, and Chen Change Loy. 2025. Arbitrary-steps Image Super-resolution via Diffusion Inversion . In *2025 IEEE/CVF Conference on Computer Vision*

and Pattern Recognition (CVPR). IEEE Computer Society, Los Alamitos, CA, USA, 23153–23163. https://doi.org/10.1109/CVPR52734.2025.02156

Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. 2024a. Paint3d: Paint anything 3d with lighting-less texture diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4252–4262.

Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. 2024b. Paint3D: Paint Anything 3D with Lighting-Less Texture Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 4252–4262.

Zheng Zeng, Valentin Deschaintre, Iliyan Georgiev, Yannick Hold-Geoffroy, Yiwei Hu, Fujun Luan, Ling-Qi Yan, and Miloš Hašan. 2024c. RGB-to-X: Image decomposition and synthesis using material- and lighting-aware diffusion models. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. Association for Computing Machinery, New York, NY, USA, Article 75, 11 pages. https://doi.org/10.1145/3641519.3657445

Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. 2022. Arf: Artistic radiance fields. In *European Conference on Computer Vision*. Springer, 717–733.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. In *IEEE International Conference on Computer Vision (ICCV)*. 3813–3824.

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. 2024b. CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–20.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.

Yuqing Zhang, Yuan Liu, Zhiyu Xie, Lei Yang, Zhongyuan Liu, Mengzhou Yang, Runze Zhang, Qilong Kou, Cheng Lin, Wenping Wang, et al. 2024a. Dreammat: High-quality pbr material generation with geometry-and light-aware diffusion models. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–18.

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

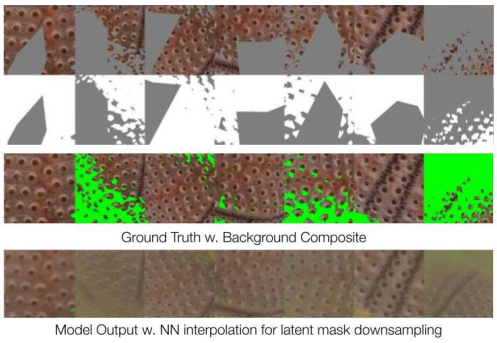


Fig. 9. Color leak effect from training model using nearest neighbor interpolation for latent mask downsampling. Top two rows show masked albedo and corresponding mask as model inputs; the third row is ground truth composited with green; the last row shows the model output.

A Data Generation

Prompt generation. We use OpenAI's o5 model to generate detailed prompts for single-view image generation. The system prompt used is "Generate concise, diverse and vivid prompts (~15-25 words) for objects. Output *one prompt per line*, no numbering." This is followed by the user prompt: "Generate 10 prompts for different kinds of **subject** with different appearances in a studio setting. Use vivid short object centric adjectives instead of long prose" where "**subject**" is replaced by a short description of the mesh object e.g. "sea urchin shell". If satisfied with the response, the model is prompted to continue generating prompts with the following user prompt: "Great! Now generate 500 more prompts for **subject** with different appearances just like that." Here is an example of a prompt generated for the sea urchin: "A sea urchin shell in smoky purple, its texture lightly powdered, rows of tiny spines etched as shallow lumps."

Single View Generation. We apply CLIP-based aesthetic score filtering to remove low-quality ControlNet-generated views. We use DiffusionRenderer [Liang et al. 2025a] for intrinsic decomposition. In our experiments, the predicted albedo generally has sufficient quality for training, and we do not observe severe artifacts propagating to the final textured results.

B Additional Model Details

Inpainting mask down-sampling. We apply a min-pooling operation to downsample masks to the latent dimension instead of nearest neighbor interpolation used in stable diffusion inpainting. This ensures that a latent pixel is masked if any image pixel in that region is masked. This is especially important during training for masking the latent space loss as nearest neighbor interpolation will leak invalid untextured regions into the loss. As shown in Fig. 9, the model trained with a nearest neighbor interpolation mask shows signs of green from the invalid texture regions. With min-pooling, the model does not show signs of colour leak.

C Additional Training Details

Diffusion loss formulation. We adopt the noise schedule and loss formulation from Stable Diffusion 2.1. Our noisy input, z_t, is computed using the DDPM formulation:

$$z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}). \tag{5}$$

The parameters α_t are computed via the scaled linear DDPM schedule. Specifically, $\beta_{\min} = 8.5 \times 10^{-4}$ and $\beta_{\max} = 1.2 \times 10^{-2}$. Define

$$\beta_t = \left(\sqrt{\beta_{\min}} + \frac{t-1}{T-1}\left(\sqrt{\beta_{\max}} - \sqrt{\beta_{\min}}\right)\right)^2, \qquad t = 1, \ldots, T. \tag{6}$$

Then set $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

For the velocity prediction [Salimans and Ho 2022], we compute,

$$v_t = \sqrt{\bar{\alpha}_t}\,\epsilon - \sqrt{1-\bar{\alpha}_t}\, z_0. \tag{7}$$

This is the target for our trained model $\mathcal{G}_M$.

Hyperparameters. For the from-scratch experiment setting, we use AdamW optimizer with learning rate $1e-4$ with batch size of 32 for 80k steps and EMA decay of 0.999. The training for a single object takes around 40 hours to train on 4 NVIDIA A100 GPUs. For the finetune experiment setting, we use 100 single views, batch size 16, and 20k steps on 2 NVIDIA L40 GPUs for 8hr. We use the finetuned model for interactive applications for the foliage, the dragon head and the lizard creature.

Training Data Generation. To train a shape-specific GLOSS model on each mesh, we generate 550 single-view renderings (512×512) using Realistic Vision v5.1 [SG_161222 2024], a community photorealistic variant of Stable Diffusion v1.5 [Rombach et al. 2022], augmented with ControlNets [Zhang et al. 2023] for normals, depth, and canny edges. To extract albedo, we use Liang et al. [2025b] and upscale it to 2048×2048 using Yue et al. [2025]. Each single-view dataset is split into 50 **eval**, 50 **test** and 450 **train** views. To finetune a new shape-specific model from a base model we generate 200 single views, 100 for training, 50 for **eval**, 50 for **test**. We choose these based on ablations of the number of single views, and the number of finetuning steps in Tb. 6 and Tb. 7.

D Evaluation Details

D.1 Baseline Details

For fair comparison, we set the text conditioning for all methods as the object name, as ours takes in an empty string and relies solely on the single view conditioning. Although Paint3D includes intrinsic image decomposition (de-lighting) in its pipeline, for fair comparison, we bypass this step and instead directly provide the albedo extracted from the single input image as supervision.

D.2 Metrics Details

For all patch-level experiments, we render patches at a resolution of 256 by 256. We fix the camera–mesh distance to 0.25 and vary the field of view from 0.4 to 0.8 to introduce variations in perceived viewing distance. For LPIPS evaluation, we sample camera views on faces with existing textures to enable comparison against ground-truth textures. For FID and CMMD, we select views that are entirely untextured.

Metric	Method	Overall	by Example									
			Brick	Cabbage	Croissant	Tire	Fire Hydrant	Gourd	Koi Fish	Barrel	Sea Shell	Turtle
LPIPS↓	Paint3D	0.378	0.580	0.501	0.370	0.379	0.265	0.432	0.337	0.328	0.348	0.242
	MV-Adapter	0.358	0.650	0.485	0.340	0.352	0.226	0.429	0.286	0.295	0.302	0.211
	Hunyuan2.1	0.383	0.577	0.498	0.352	0.370	0.272	0.604	0.303	0.314	0.328	0.214
	Trellis2.0	0.356	0.547	0.482	0.341	0.358	0.283	0.407	0.299	0.307	0.327	0.213
	Ours	**0.302**	**0.462**	**0.386**	**0.256**	**0.319**	**0.208**	**0.361**	**0.281**	**0.283**	**0.283**	**0.181**
FID↓	Paint3D	98.765	124.633	131.087	83.139	149.776	82.467	97.246	84.359	76.715	79.076	79.149
	TexGen	106.917	224.421	130.891	60.657	120.995	105.353	80.442	80.190	52.466	109.557	104.197
	MV-Adapter	76.98	166.61	83.54	48.58	**97.36**	85.45	**51.04**	51.10	53.52	64.99	67.60
	Hunyuan2.1	**73.702**	**93.570**	**88.890**	63.040	99.270	87.750	73.760	52.510	**47.840**	**51.560**	78.830
	Trellis2.0	102.22	185.81	153.14	75.48	108.47	106.83	96.52	61.30	51.99	96.71	85.91
	Ours	80.258	130.522	95.320	**50.175**	119.024	**63.553**	**59.799**	60.633	75.219	88.413	**59.925**
DreamSIM↓	Paint3D	0.412	0.480	0.368	0.360	0.543	0.396	0.348	0.368	0.507	0.387	0.359
	TexGen	0.436	0.626	0.391	0.353	0.526	0.377	0.398	0.361	0.547	0.422	0.361
	MV-Adapter	0.370	0.629	0.301	0.314	0.430	0.356	0.301	0.300	0.458	0.318	0.293
	Hunyuan2.1	0.437	0.503	0.372	0.394	0.493	0.438	0.509	0.395	0.525	0.380	0.362
	Trellis2.0	0.411	0.484	0.354	0.395	0.453	0.464	0.358	0.381	0.497	0.373	0.352
	Ours	**0.344**	**0.396**	**0.296**	**0.289**	**0.439**	**0.340**	**0.278**	**0.331**	**0.486**	**0.316**	**0.270**
CMMD↓	Paint3D	0.728	**0.597**	1.198	0.709	0.836	0.827	0.706	0.923	0.274	0.607	0.604
	TexGen	0.883	1.538	1.262	0.743	0.942	0.800	0.897	0.803	**0.257**	0.845	0.738
	MV-Adapter	0.666	1.424	0.896	0.816	0.507	0.678	0.521	0.570	0.224	0.476	0.546
	Hunyuan2.1	0.494	0.806	0.664	0.476	0.591	0.470	**0.345**	**0.407**	0.312	**0.417**	0.456
	Trellis2.0	0.545	0.913	0.812	0.619	0.475	0.685	0.512	0.370	0.179	0.361	0.521
	Ours	**0.478**	0.888	**0.430**	**0.384**	**0.573**	**0.415**	0.365	0.506	0.291	0.518	**0.405**

Table 3. **Quantitative Results:** Overall (DreamSim) and patch-level (LPIPS, FID, CMMD) metrics on texture generation task conditioned on a single view, across 10 shapes with 50 views each, reveal competitive performance of our method, despite using no textured 3D training data for training.

Metric	Method	Overall	by Example								
			Cabbage1	Cabbage3	Cantaloup	Fish1	Fish3	Fish4	Toad	Tortoise	Turtle
LPIPS↓	MV-Adapter	0.366	0.319	0.299	0.598	0.301	0.409	0.247	0.399	0.387	0.334
	Hunyuan2.1	0.299	0.287	0.266	0.440	0.253	0.335	0.187	0.327	0.318	0.279
	Trellis2.0	0.307	0.284	0.279	0.471	0.268	0.379	0.188	0.321	0.306	0.269
	Ours	**0.282**	**0.278**	**0.255**	**0.407**	**0.248**	**0.308**	**0.172**	**0.315**	**0.287**	**0.268**
FID↓	TexGen	124.460	**116.061**	84.111	290.178	130.762	120.826	**107.264**	52.956	113.996	103.989
	MV-Adapter	122.865	202.740	117.259	175.604	106.143	113.812	146.131	63.382	84.237	96.476
	Hunyuan2.1	**93.045**	196.583	**60.828**	**96.139**	**89.571**	**84.585**	119.341	51.564	**70.519**	**68.278**
	Trellis2.0	110.773	191.301	99.255	134.525	113.500	116.402	120.003	62.463	77.913	81.596
	Ours	104.108	123.885	83.397	198.546	98.445	101.567	110.265	**50.494**	87.696	82.681
DreamSim↓	TexGen	0.538	0.459	0.499	0.493	0.564	0.509	0.571	0.595	0.599	0.552
	MV-Adapter	0.473	0.318	0.361	0.540	0.507	0.487	0.481	0.525	0.525	0.512
	Hunyuan2.1	0.356	**0.264**	**0.317**	0.280	0.425	**0.317**	0.355	**0.415**	0.441	0.387
	Trellis2.0	0.371	0.293	0.341	**0.263**	0.427	0.367	0.372	0.458	0.417	0.401
	Ours	**0.352**	0.265	0.342	0.319	**0.407**	0.340	**0.352**	0.427	**0.392**	**0.326**
CMMD↓	TexGen	1.143	1.346	1.041	2.980	0.990	0.883	0.584	0.585	0.912	0.969
	MV-Adapter	0.961	1.376	0.873	1.798	0.904	0.942	0.867	0.482	0.590	0.821
	Hunyuan2.1	**0.570**	1.455	**0.546**	**0.491**	**0.497**	**0.444**	**0.540**	**0.309**	**0.380**	**0.471**
	Trellis2.0	0.777	2.058	0.835	0.793	0.571	0.626	0.602	0.437	0.492	0.577
	Ours	0.694	**1.336**	0.721	0.905	0.640	0.609	0.598	0.383	0.548	0.510

Table 4. **Quantitative Results on Transfer Meshes:** Overall and per-example LPIPS, FID, DreamSim, and CMMD on the texture-transfer evaluation set (9 sub-meshes spanning 3 base shapes). Best per column in bold.

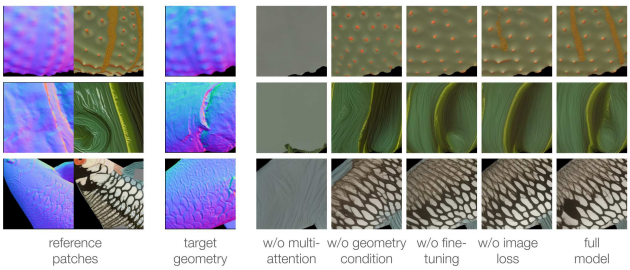
D.3 Model Ablations

We present visual comparisons to analyze key architectural and training design choices. Removing multi-attention causes the model to completely fail at texture generation. Without geometry conditioning, the model relies primarily on the inpainting mask shape and texture priors, leading to geometry-inconsistent results: in the croissant examples, surface creases are misaligned with the underlying geometry, and in the fish example, fin textures incorrectly extend onto the body. (See Fig. 10)

The variant without the image-space loss performs competitively with the full model but occasionally loses fine-grained texture details due to the absence of image-level supervision. Similarly, the model without fine-tuning produces competitive results overall, but sometimes fails to capture subtle texture variations, as illustrated by the urchin shell example.

D.4 Completion Ablations

We report quantitative metrics for completion results produced by different variants of our automatic texturing pipeline (Tb. 5). Our results show that setting SyncMVD latent guidance strength to 2 yields the highest quantitative texture quality, whereas increasing the guidance strength to 10 degrades performance. All ablated variants underperform the original completion setting, indicating that each component of the pipeline is essential for GLOSS to achieve high-quality automatic texture generation.



(a) ablations

Fig. 10. We show a visual comparison of our model ablation experiments. We show reference albedo and geometry patches on the left and each model variant's output.

Experiment	LPIPS↓	DreamSim↓	FID↓
guidance_2 (final)	**0.3031**	0.3441	80.24
wo_syncmvd	0.3040	**0.3393**	**74.81**
guidance_4	0.3044	0.3514	86.64
guidance_10	0.3169	0.3829	120.97
overlap_0.4	0.3099	0.3644	97.85
wo_color_correct	0.3044	0.3497	84.41
wo_custom_weight	0.3070	0.3493	82.12
wo_nnfm	0.3090	0.3507	84.79

Table 5. Quantitative evaluation of textures generated with variations of final completion experiment settings. We report average LPIPS, DreamSim, and FID metrics over 10 objects.

	Cabbages		Fishes		Turtles		Overall	
	LPIPS↓	FID↓	LPIPS↓	FID↓	LPIPS↓	FID↓	LPIPS↓	FID↓
v100–10k	0.260	**75.87**	0.176	111.95	0.263	**76.32**	0.233	**88.05**
v100–20k	**0.253**	80.78	**0.169**	109.63	**0.253**	76.33	**0.225**	88.91
v100–50k	0.255	80.16	0.170	**104.27**	0.268	88.24	0.231	90.89

Table 6. **Step-count ablation** on the 100 view variant. LPIPS / FID at 10k, 20k, and 50k fine-tuning steps. We show that finetuning results are best at an early stage, enabling faster convergence.

	Cabbages		Fishes		Turtles		Overall	
	LPIPS↓	FID↓	LPIPS↓	FID↓	LPIPS↓	FID↓	LPIPS↓	FID↓
50 views	0.255	**72.80**	0.170	108.66	**0.257**	**78.77**	**0.227**	**86.74**
100 views	**0.255**	80.16	**0.170**	**104.27**	0.268	88.24	0.231	90.89
200 views	0.255	83.40	0.172	110.26	0.268	82.68	0.232	92.11

Table 7. **View-count ablation** comparing v50, v100, and v200 fine-tunes. We show that we can finetune with a small number of views.

D.5 Finetune Ablations

We conduct finetune experiment ablation on single view numbers and training steps.

D.6 Attention Analysis

In Fig. 11 we visualize the batch multi-attention weights from a single attention head in the UNet. The heatmap overlaid on the model output is obtained using the attention weights from the center patch of the target image as the query. The attention weights are reshaped and upsampled to match the image resolution. The weights are then normalized across the batch before applying a colour map. While some attention heads may attend to similar parts of the image, some heads like the one shown in Fig. 11 show distinct differences

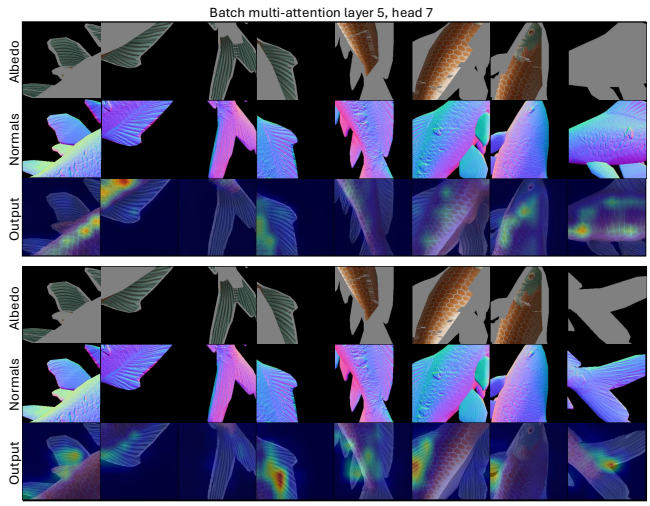


Fig. 11. Batch multi-attention visualization comparing two batches with the same reference images but a different target image (last in the batch). The overlaid heatmap shows how the center patch token in the target image attends to all patch tokens across the batch.

based on the target image. We see that the first batch highlights regions with fish scales while the second batch highlights more fins.

E Inference Settings and Timings

We use a patch size of 256x256 at train and inference time, and 1K or 4K textures for our experiments (client-server architecture of our add-on makes 1K textures faster to work with). For interactive completion, typically 7 references and 1 target patch are used.

Unoptimized timings. Rendering patches and NNFM reference matching takes around 1s. Model inference takes 1.4s. Backprojection from camera image space to texture space takes 1s for 1K texture resolution and almost 3s for 4K texture resolution. Total processing time is 3.4s for 1K texture painting and 5.3s for 4K textures. This is not including GLOSS server and Blender add-on communication time. Model inference and texture processing timings can be further optimized to near real-time speeds, as shown to be possible by Hu et al. [2024].

F Failure Cases

Our model relies on learned local correlations between geometry and texture to texture an asset. While it excels at generating fine-grained details with high geometric and reference fidelity, its ability to infer semantic appearance from limited local reference context remains constrained. For example, the turtle's face and mouth lack fine-grained semantic details, while the koi fish eyes are not rendered with sufficiently high-quality textures. We also examine failure cases in regions where the correlation between geometry and texture is weak. Although it can reproduce repetitive patterns to some extent, the fidelity and consistency of these patterns remain limited. The model may also fail to preserve small, stochastic, non-repetitive strokes that are weakly correlated with the local geometry. We show such cases in Fig. 12.

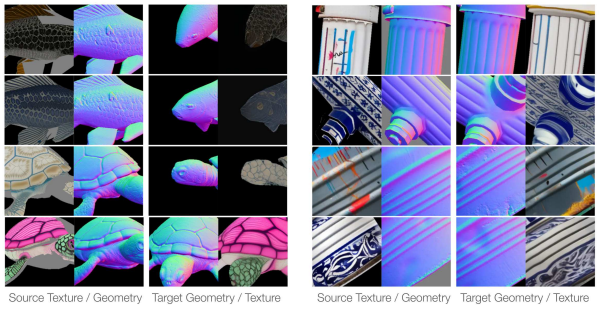


Fig. 12. On the left we show examples of semantic texture failure. On the right we show examples of failure cases due to low geometry and texture correlation.

G User Study Details

To evaluate the potential of our interactive texture painting model, we conducted a pilot user study in which artists interacted with a Blender-based prototype built on our method. This section details the study design, participant background, experimental procedure, and collected feedback.

G.1 Participants

We recruited five participants with prior experience in 3D texturing and general familiarity with Blender [Community 2026]. The study was advertised through online 3D artist communities. Applicants were screened based on their experience with 3D texture painting, and both students and professional artists were considered. Each study session lasted approximately 90 minutes. Participation was voluntary, and no compensation was provided.

G.2 Methodology and Tasks

The study consisted of three phases: a pre-study questionnaire, hands-on texturing tasks, and post-task interviews. Prior to the tasks, participants completed a questionnaire covering their background in 3D texturing, typical workflows, tools used in practice, and familiarity with AI-assisted texturing methods. Participants were also asked to share their perspectives on usability and creative control in existing tools. Each session began with a brief introduction to generative AI concepts and conditional texture generation from rendered views, ensuring that participants shared a baseline technical understanding. Participants then evaluated two texturing tools: (1) an automatic texturing baseline using the Hunyuan2.1 texture generation backbone [Hunyuan3D et al. 2025], and (2) our interactive texture painting system. Both tools are implemented as Blender add-ons. The order of the tools was randomized across participants. For both tools, participants selected one of three objects—a koi fish, a sea urchin, or a croissant—along with corresponding single-view reference images. For the automatic baseline, participants selected a single view of the mesh and triggered texture generation, after which they examined the resulting texture mapped onto the object in Blender. For the interactive system, participants completed two tasks: (1) editing an existing partial texture map, and (2) painting a texture map from scratch. Before starting, participants were given a brief tutorial on the Blender add-on interface and were allowed to practice on a sample model. During the tasks, participants generated texture brushes from selected reference views and were instructed to follow a hypothetical art-director brief by combining visual elements from multiple views. Participants were asked to generate at least two texture brushes and to paint approximately 70% of the mesh to gain sufficient hands-on experience with the system. Throughout the tasks, participants were encouraged to verbalize their thoughts and observations.

G.3 Questions

After completing each tool, participants answered the following questions:

- Q1: **Usefulness**: What features or capabilities of this tool did you find useful? Which aspects did not work well, and how could they be improved?
- Q2: **Controllability**:
  Rate from 1 to 5: I feel I have sufficient control over the final texture when using this tool.
- Q3: **Complementarity**:
  Rate from 1 to 5:
  a. I feel this tool complements the tools I use in my daily practice and I would love to use this feature as part of my texturing workflow.
  b. I would not like to use this system frequently.
  c. I feel I could work more efficiently if I used this system.

After completing both tools, participants were asked comparative questions:

- Q1: **Workflow Integration**: Which tool are you more likely to integrate into your texturing workflow, and why?
- Q2: **Texture Quality**: Which tool produced textures that were more faithful to the reference images? What differences did you observe between the two methods? Which tool better captured fine geometric details?
- Q3: **Creative Affordance**: Which tool better supports the creative process of texturing 3D objects, and why?

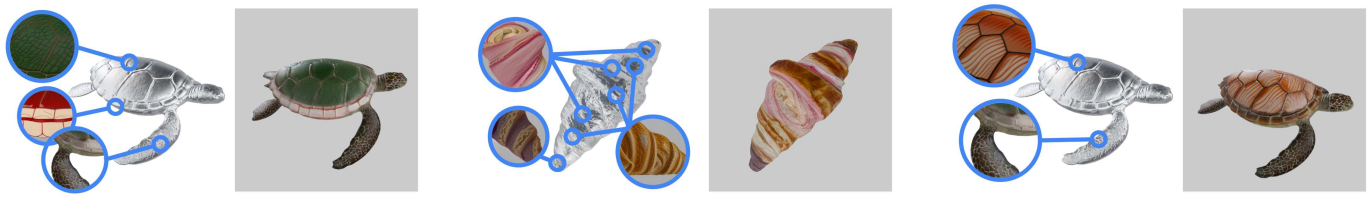
Finally, participants answered feature-specific questions related to the interactive system:

- Q1: **Texture Quality**: When using entire views or selected regions as reference brushes, does the auto-brush tool produce textures aligned with your intent?
- Q2: **Controllability**: The interface supports undoing, regenerating, and clearing textures. Do you find this iterative process valuable compared to fully automatic texturing?
- Q3: **Creative Affordance**: Does the system enable experimentation with diverse ideas and combinations, and does this benefit your creative workflow?

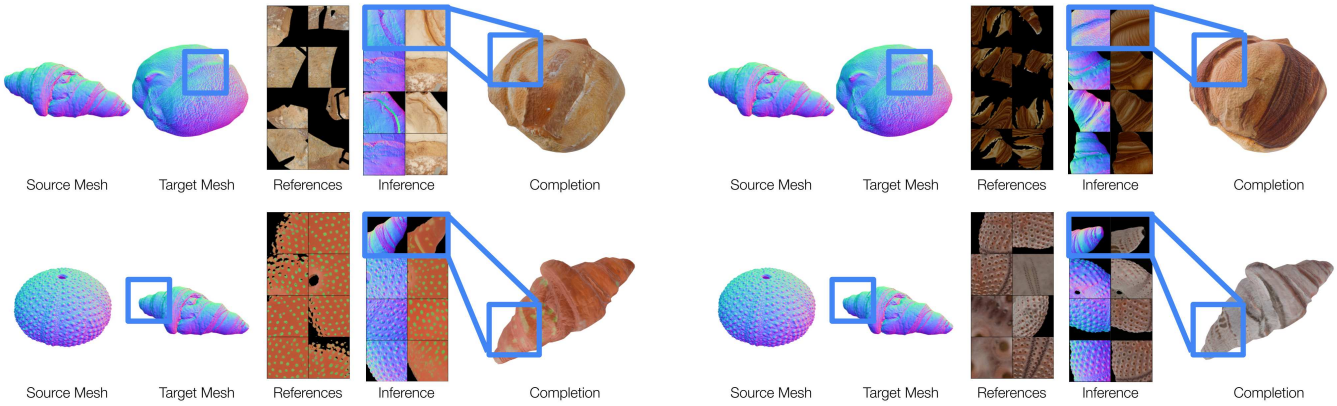
G.4 Suggestions for Improvement

Despite an overall positive response to our interactive texturing prototype, participants identified several limitations, with suggestions for implementation improvements and future work. Key areas for improvement include finer control over inpainting strength, real-time texture updates during brush strokes and painting with full material channels. Due to the fixed patch-based update scheme, participants found it difficult to make small, localized edits, as the

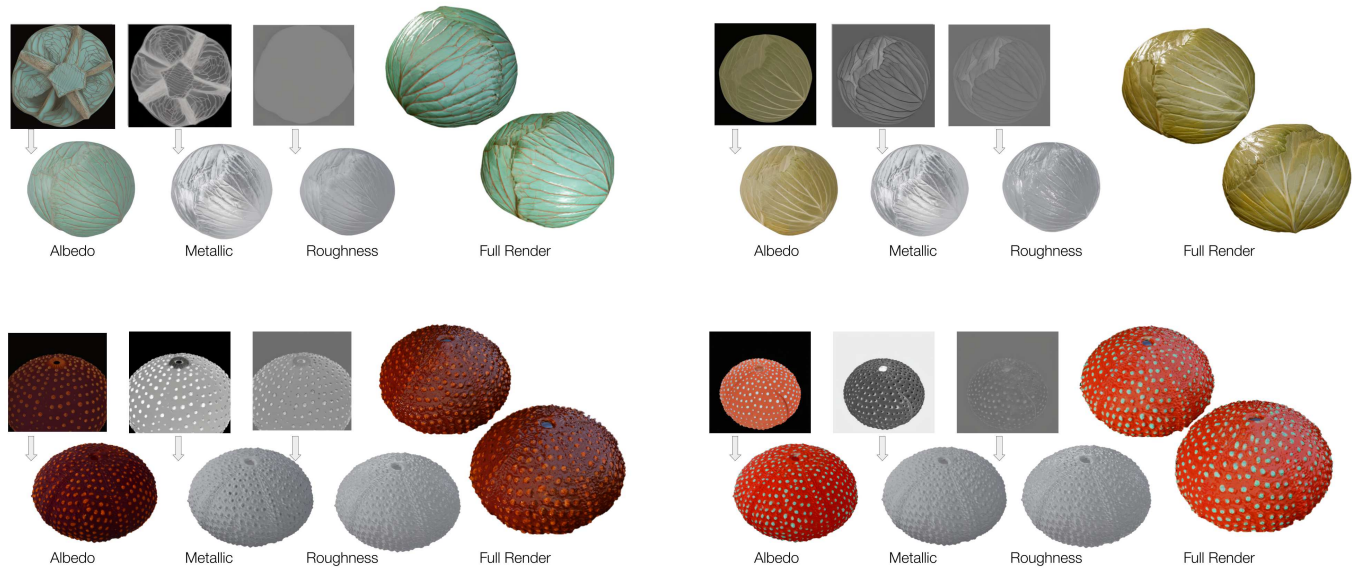
inpainting model tends to overly smooth transitions and lose sharp details. In addition, the model occasionally struggles to produce high-fidelity results from certain camera angles, which can reduce perceived controllability and visual quality. With continued research and development, we aim to further advance interactive texturing techniques for high-quality, production-ready 3D assets.



(a) More blending results. We blend multiple textures seamlessly on the same target mesh.



(b) More Results on Texture Transfer. Once we trained a GLOSS model on a source mesh, the same model can be used to texture new shapes. The transfer solely depends on the local patch-wise geometry and albedo between the target and the references, and generalizes to new geometry to a certain extent. Notice how we can transfer correlations between normal and urchin shell bump texture to the creases on a croissant.



(c) More Results on PBR Completion. Trained on albedo, GLOSS generalizes to completion in metallic and roughness channels. We show renders of albedo channel, metallic only (basecolor grey), roughness only (basecolor grey) as well as the full material channel renderings.

Fig. 13. Additional result on texturing applications.

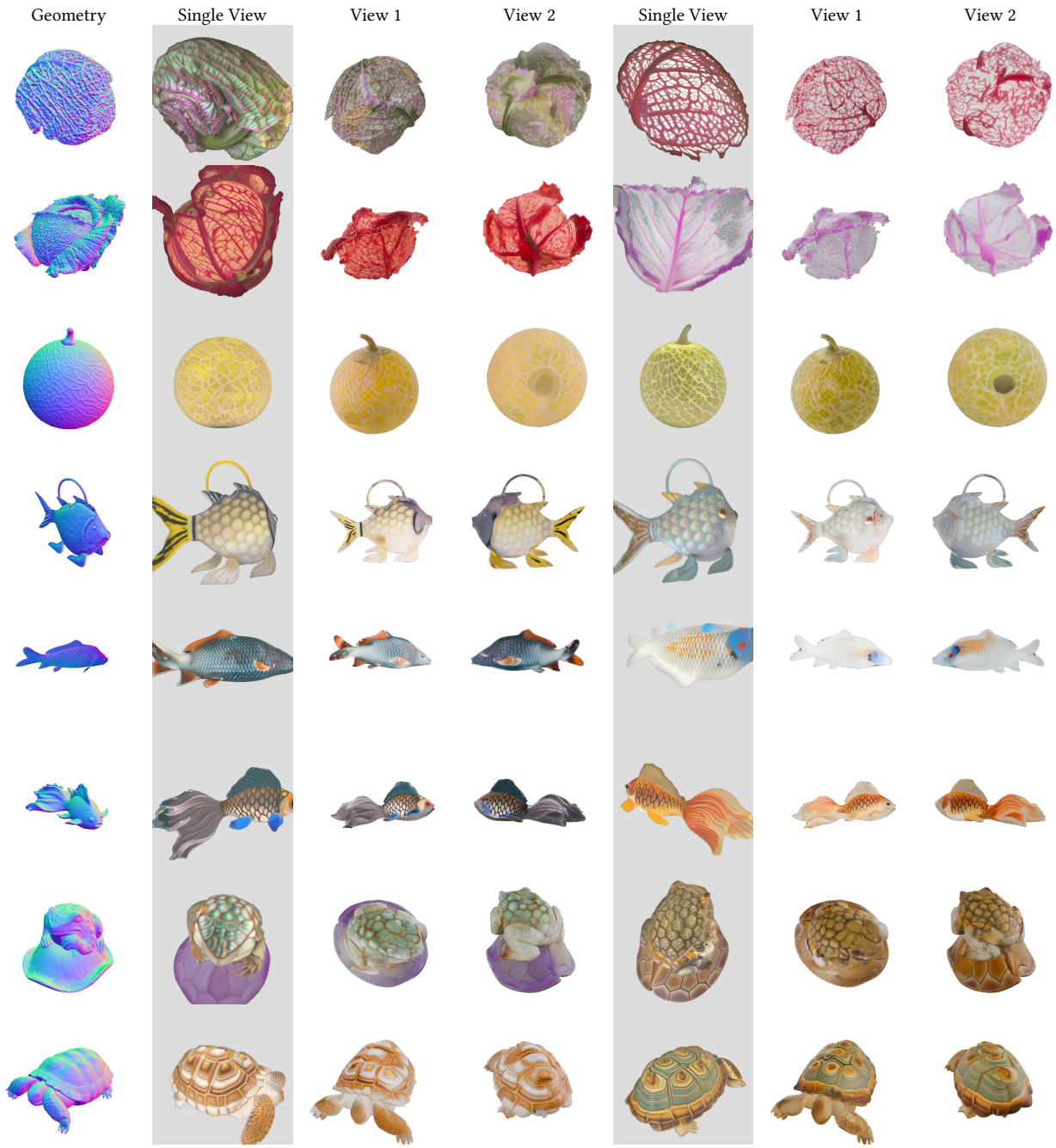


Fig. 14. Additional finetuning completion results.

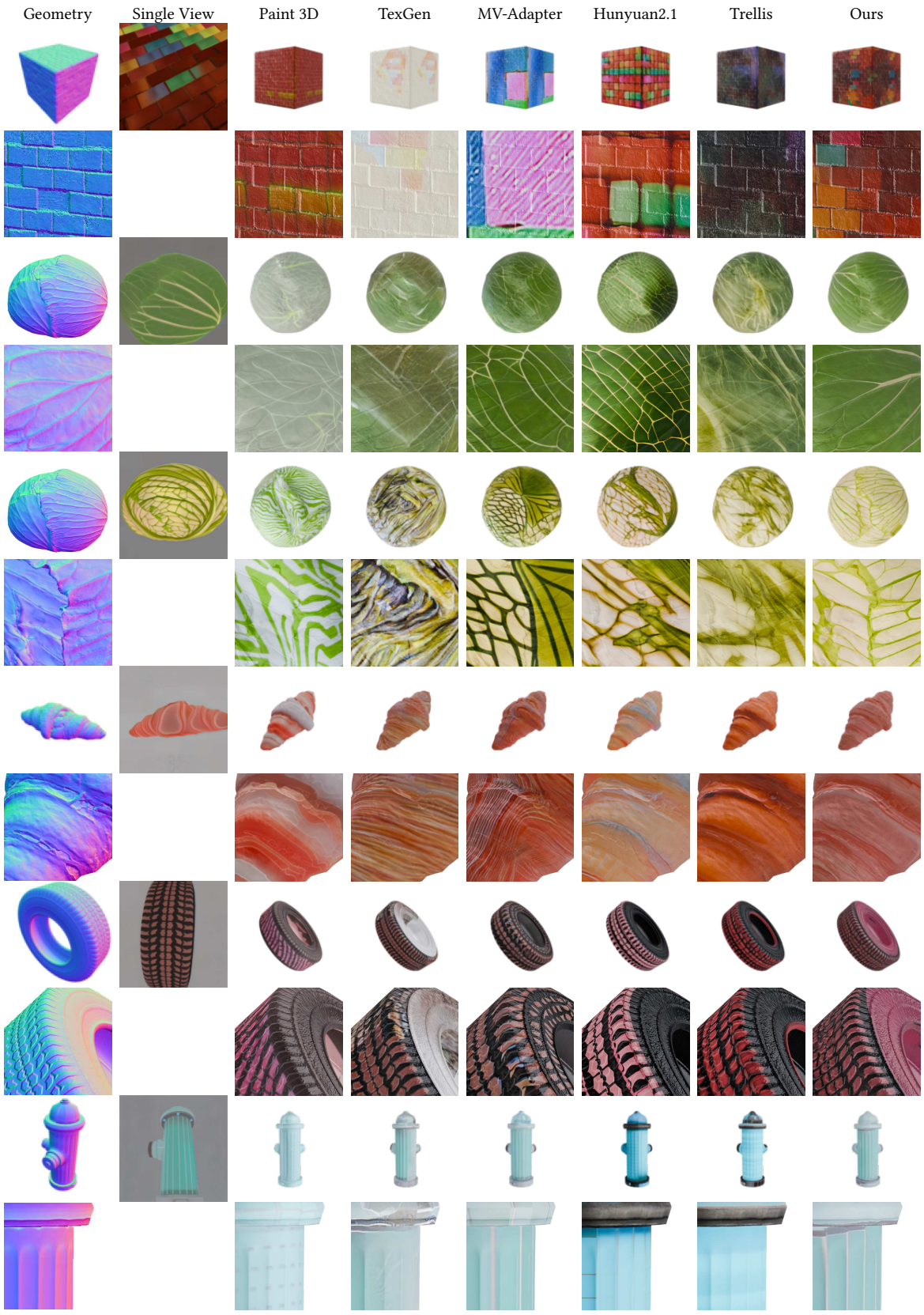


Fig. 15. Additional results on the baseline comparison, both global view and zoomed-in views.

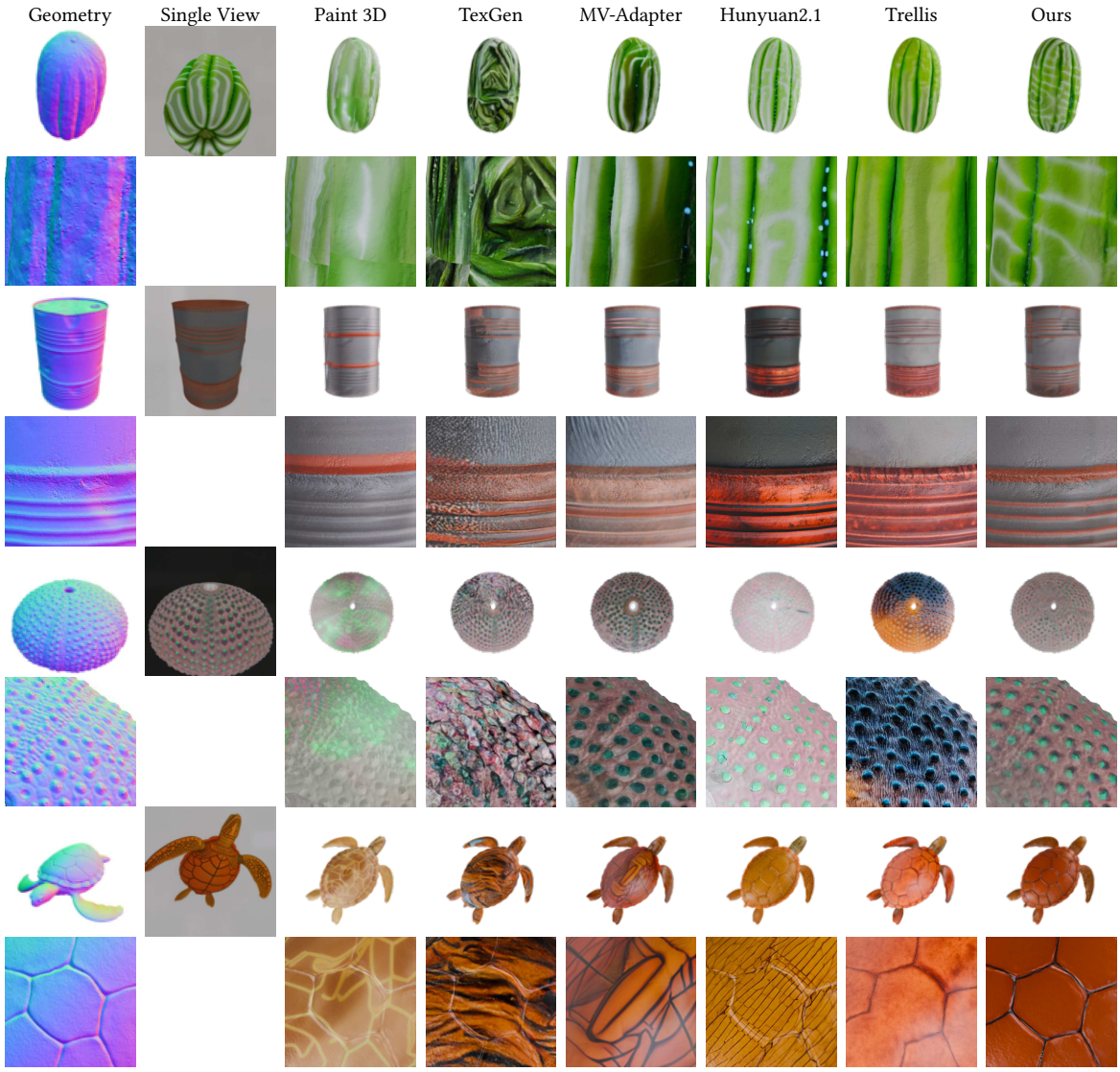
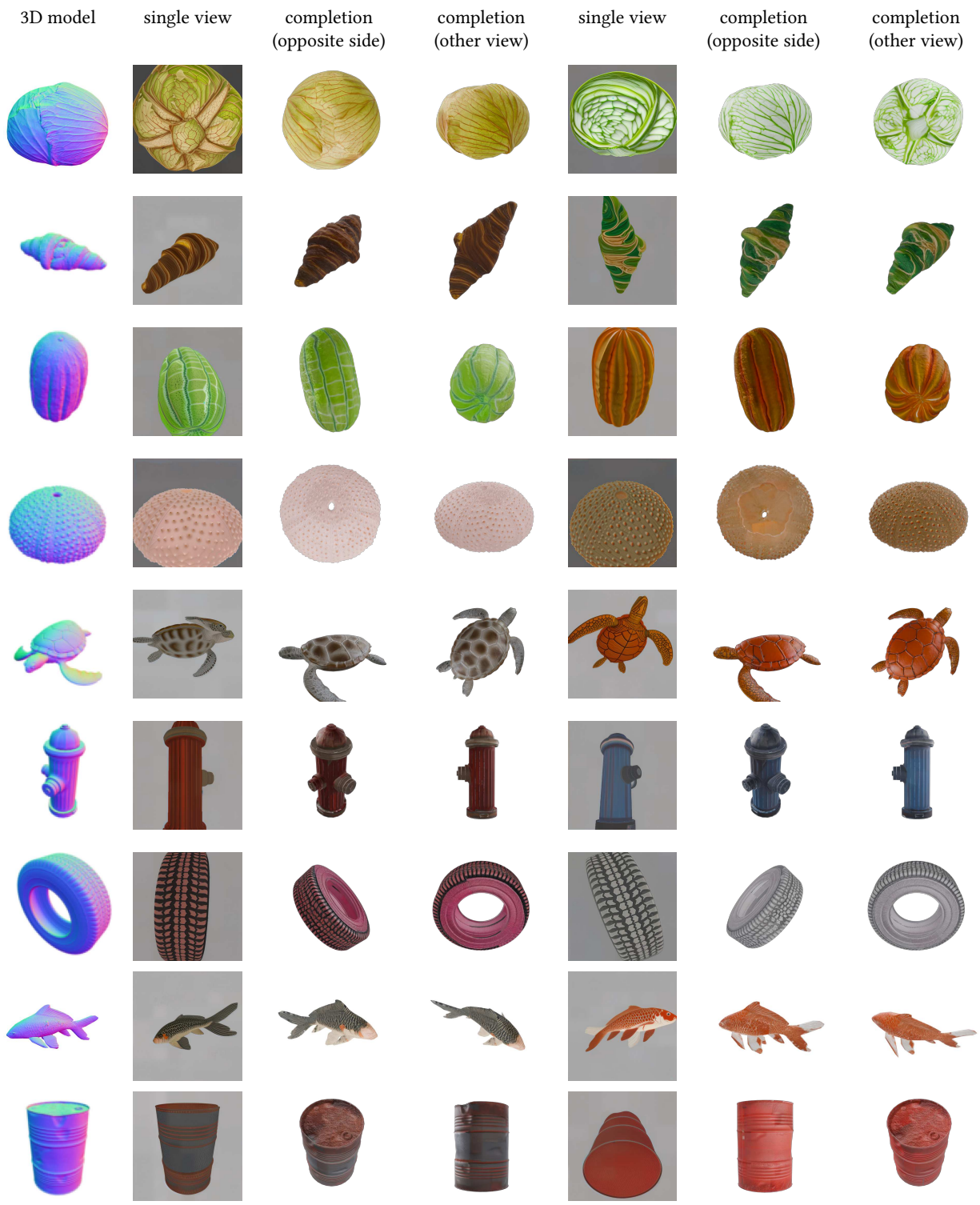


Fig. 16. Additional results on the baseline comparison, both global view and zoomed-in views.



(a) Texture Completion

Fig. 17. We show more automatic texture completion results with GLOSS. We show views of the 3D model that are opposite to the conditional view. We also show the textured 3D model from a view other than these two.