DOI: 10.1111/cgf.70191
Eurographics Symposium on Geometry Processing 2025
M. Attene and S. Sellán
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 5

Im2SurfTex: Surface Texture Generation via Neural Backprojection of Multi-View Images

Yiangos Georgiou[1,2], Marios Loizou[1,2,3], Melinos Averkiou[1,2] and Evangelos Kalogerakis[2,3]

[1]University of Cyprus [2]CYENS CoE [3]Technical University of Crete

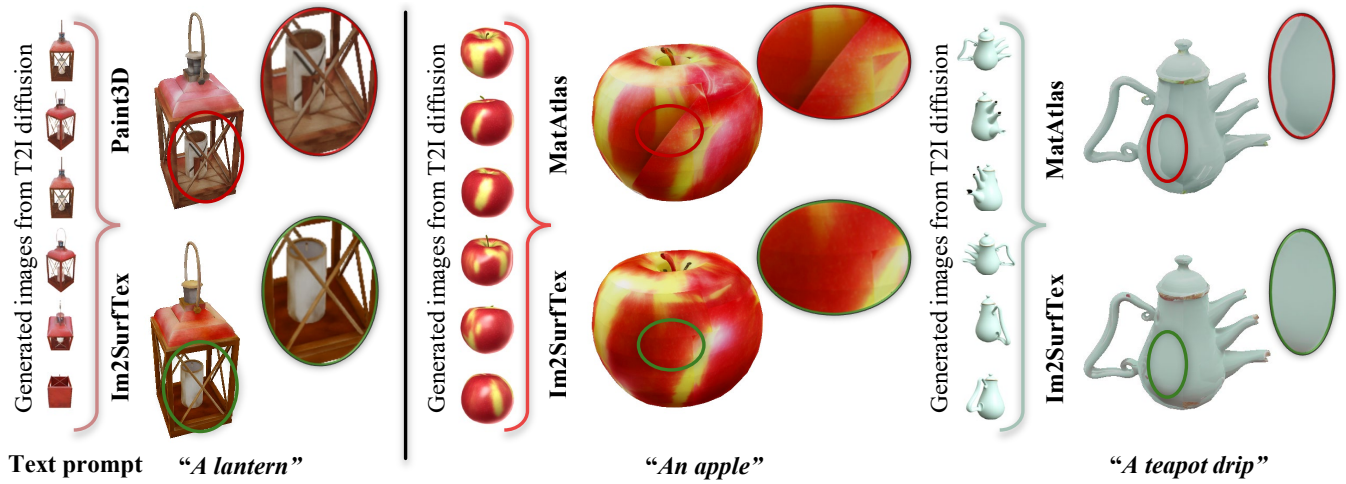

Figure 1: *Given a text prompt and an untextured 3D shape, Im2SurfTex generates a texture for it by learning to backproject images produced by text-to-image (T2I) diffusion models to the shape's texture space.* Left: *Im2SurfTex diminishes artifacts on surfaces with self-occlusions and complex geometry, preserving finer details where alternatives like Paint3D [ZCQ*24] struggle, resulting in backprojection issues, such as the guard grill's texture appearing on the candle inside the lantern.* Right: *Im2SurfTex prevents seam formation on high-curvature surfaces and seamlessly blends multiple views. In contrast, other approaches, such as MatAtlas [CDG*24], often introduce texture discontinuities, as seen on the apple, or fail to resolve multi-view inconsistencies, leading to visible artifacts, as in the teapot.*

Abstract

We present Im2SurfTex, a method that generates textures for input 3D shapes by learning to aggregate multi-view image outputs produced by 2D image diffusion models onto the shapes' texture space. Unlike existing texture generation techniques that use ad hoc backprojection and averaging schemes to blend multiview images into textures, often resulting in texture seams and artifacts, our approach employs a trained neural module to boost texture coherency. The key ingredient of our module is to leverage neural attention and appropriate positional encodings of image pixels based on their corresponding 3D point positions, normals, and surface-aware coordinates as encoded in geodesic distances within surface patches. These encodings capture texture correlations between neighboring surface points, ensuring better texture continuity. Experimental results show that our module improves texture quality, achieving superior performance in high-resolution texture generation.

CCS Concepts

• ***Computing methodologies*** → ***Texturing;*** *Neural networks;*

1. Introduction

Producing compelling 3D assets has become an increasingly active area of research in the field of generative AI. Despite the significant progress in training large-scale generative models of 3D geometry [LGT*23, LWVH*23, WLW*24, GSW*22, JN23, NJD*22, VWG*22, LXJ*24, LSC*24, SCZ*23, ZWZ*24], synthesizing compelling, seamless, and high-quality textures for 3D shapes and scenes still remains challenging. One major obstacle is the limited availability of training 3D asset datasets with high-quality textures. Several recent approaches [CSL*23, RMA*23, CKF*23, CDG*24,

Figure 2: *A gallery of 3D shapes from various categories, textured by Im2SurfTex.*

ZCQ*24,LXLW24,CMZ*24] have resorted to leveraging powerful generative 2D image models, such as denoising diffusion models pre-trained on massive 2D datasets, to guide surface texture generation. These approaches generate images across different views conditioned on text prompts and depth maps rendered from the given 3D shapes or scenes, then attempt to combine the generated images onto the surface.

Unfortunately, these methods suffer from a number of limitations. First, merely projecting the generated images back to the surface tends to generate texture distortions in areas of rapid depth changes or high curvature surface regions (Figure 1, left). Second, the images generated from different views are often combined with ad-hoc criteria to create a single surface texture. For example, many methods [CSL*23, RMA*23] simply transfer the colors from the most front-facing view to each texel (i.e., pixel in the texture map), causing visible seams in the texture maps especially in areas where colors of neighboring texels are copied from different views (Figure 1, middle & right). Other methods rely on simple averaging schemes of RGB colors [CDG*24] or image latents, causing significant color bleeding. Others resort to global texture optimization techniques [CKF*23, LXLW24], which are slow and can still fail to generate coherent textures since their initialization is still based on ad-hoc thresholds for view selection and color averaging.

Our approach, named Im2SurfTex, addresses the above limitations with the introduction of a novel, optimization-free module that can be easily integrated to existing texture generation approaches. The module is trained to combine color information from multiple viewpoints to textures through a cross-attention mechanism, where for each surface point, several candidate image neighborhoods across different views are examined and combined back in texture space depending on local surface geometry, as encoded in 3D positions, normals and geodesic distances within these patches. In this manner, the attention mechanism captures local context based on surface (geodesic) proximity rather than relying solely on 3D Euclidean proximity that might correlate surface region textures far from each other in geodesic sense. Our module yields coherent textures efficiently, without requiring any slow optimization procedures. Our experiments indicate significant improvements in generated texture quality, measured by different scores, including FID [HRU*17], KID [BSAG18], CLIP [SBV*22] metrics, when compared to alternatives.

In summary, our method introduces the following contributions:

- a cross-attention mechanism that learns how to wrap generated images from different views onto a single, coherent surface texture map.
- we integrate this modules with multiple alternative backbones based on texture map diffusion showing consistent improvements in synthesized texture quality.

2. Related Work

Early Works on Texture Synthesis. Classic texture generation methods mostly focused on example-based approaches [WLKT09], including region-growing techniques [EL99, WL00] and strategies leveraging local coherence [Ash01, TZL*02] to maintain consistency across synthesized textures. Patch-based methods [EF01, KSE*03] synthesized textures using patch patterns from reference

images, while other techniques [KEBK05, HZW*06] progressively refined synthesized texture based on optimization procedures. Another significant research direction involved directly generating textures on 3D surfaces [Tur01, ZMT06, FSDH07] by exploiting vector fields defined over the surface to seamlessly map textures onto complex geometries. All these example-based approaches were unable to capture texture variability, generate diverse textures, or handle diverse shapes.

Text-Guided Diffusion Models for Image Synthesis. Our method builds upon diffusion models [SDWMG15, HJA20, ZZZ*23], which have demonstrated superior performance compared to GANs [GPAM*14, ZPIE17] in image generation tasks [RBL*22, NDR*21, SCS*22, RDN*22, ZRA23]. Closely related to our approach are text-guided generation models for 3D object synthesis, where text-to-image diffusion models are used for distilling 3D objects as neural radiance fields [MST*20, KKLD23] via Score Distillation Sampling [PJBM22, WDL*23]. Following DreamFusion [PJBM22], several approaches have been proposed [LGT*23, MRP*23, CWWL24, WLW*24, TMT*24, SWY*24]. However, these methods do not specifically target the task of texture generation.

Texture Generation via T2I Diffusion Models. Initial efforts in texture generation via text-to-image diffusion models, such as Text2Tex [CSL*23] and TEXTure [RMA*23], employed depth-conditioned diffusion models [RBL*22, ZRA23] to iteratively inpaint and refine the textures of 3D objects. Both methods start with a preset viewpoint, generating texture updates for corresponding regions of the 3D object by back-projecting depth-guided views. In Text2Tex, a coarse texture is progressively created by iterating over multiple viewpoints and refining the texture map based on high- surface-coverage viewpoints. This refinement applies a denoising diffusion process of moderate strength to preserve the texture's original appearance while enhancing details. Similarly, TEXTure divides the texture map into distinct regions labeled as *keep*, *refine*, or *generate*, enabling selective refinement or generation of textures. Despite these efforts to achieve global consistency, these methods employ ad hoc thresholds to define the different shape regions and hand-engineered strategies for backprojection, often leading to seams between texture regions synthesized from different viewpoints.

To address these issues, other methods such as TexFusion [CKF*23] leverage *latent* diffusion to interlace diffusion and back-projection steps, producing 3D-aware latent images that are subsequently decoded and merged into a texture map. Similarly, SyncMVD [LXLW24] employs a latent texture map where all views are encoded at each denoising step, further enhancing consistency in geometry and appearance. TexGen [HGZ*24] introduced a multi-view sampling and resampling framework that updates a UV texture map iteratively during denoising, aiming to reduce view discrepancies. Still, these methods rely on ad hoc blending masks or heuristics for aggregating texture information from different views, such as mere averaging or using the most front-facing view information for each texel. In contrast, our approach learns to aggregate color information from multiple views based on both geometry and texture information, promoting the generation of more coherent and seamless surface texture maps.

A more recent approach, Paint3D [ZCQ*24], achieves impressive texture generation results, by adopting a two-stage texture generation strategy. In the first stage, a coarse texture is created by backprojecting views to texture space via the heuristic of using the most front-facing view information for each texel, as in previous methods. The second stage involves a refinement and inpainting process that utilizes a diffusion model in UV texture space, conditioned on a UV position map encoding the 3D adjacency information of texels. While this method directly encodes 3D geometric information into the texture map, it can still result in misaligned textures due to its employed heuristic during its coarse stage. The subsequent texture refinement steps often fail to fix the artifacts of the coarse stage, as demonstrated in our experiments. Another related method, MatAtlas [CDG*24], incorporates a three-step denoising process with sequential operations and line conditions to preserve geometry and style consistency. However, MatAtlas employs an averaging heuristic for blending the final texture from the generated views, leading to inconsistencies or overly smooth surfaces, as also demonstrated in our experiments. TEXGen [YYG*24] takes a different approach by directly training a large-scale diffusion model in the UV texture space, and integrating convolution operations in UV space with 3D-aware attention layers in their denoising network to achieve high-resolution texture synthesis. However, their method still faces challenges in maintaining cross-view consistency since the generated textures are conditioned on single-view images which are merely backprojected to the UV space to derive the initial partial texture maps used in their diffusion.

In a concurrent work, MVPaint [CMZ*24] introduces a multi-stage texture generation framework. In the initial stage, a latent texture map is employed during multi-view projection to create a synchronized texture across multiple views, similar to SyncMVD [LXLW24]. This is followed by an inpainting stage, where uncovered texture regions are filled using a dense colored point cloud extracted from the generated texture map. Colors are propagated to empty texels in a spatially aware manner using inverse distance weighting and normal similarity between neighboring points. Finally, a refinement stage upscales the texture map and smooths out seams through weighted color averaging among k-nearest neighbors in 3D space. Still, MVPaint relies on an averaging scheme to aggregate view information into texture space. In contract, our approach learns this aggregation by encoding both geometric and appearance information from multiple views to produce textures with greater consistency.

Expanding beyond single object texture generation, InstanceTex [YGC*24] focuses on texture generation for 3D scenes, employing a local synchronized multi-view diffusion strategy to improve local texture consistency across multiple objects. 3D Paintbrush [DLAH24] specializes in localized stylization of single objects, using cascaded score distillation to refine textures within specific object regions. These approaches differ from our method in scope: InstanceTex is tailored for stylistic consistency in large environments, while 3D Paintbrush targets localized edits.

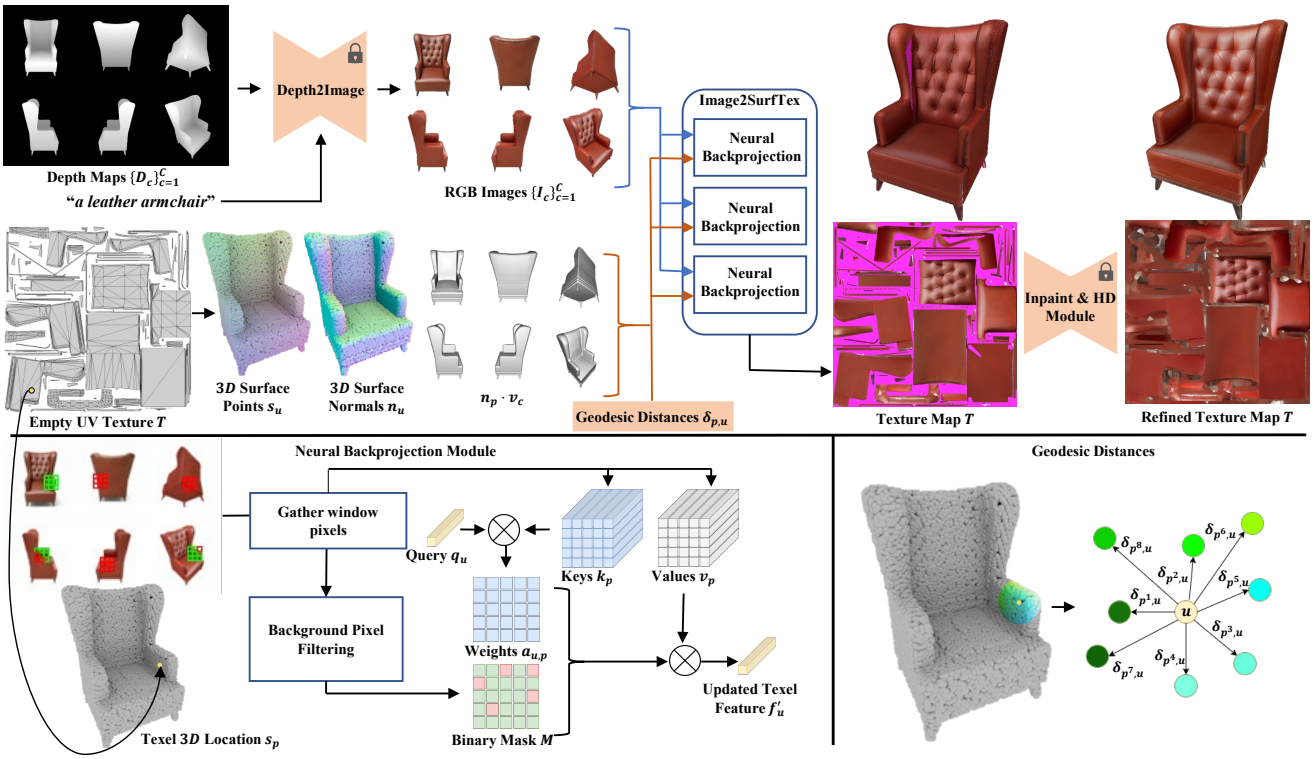

Figure 3: *(Top) The Im2SurfTex, pipeline utilizes depth images and a text prompt to generate a number of candidate views (RGB images) for a given shape. The views are aggregated through a learned backprojection module that incorporates geometric information, such as 3D location, normals, angles between normals, and view vectors, as well as geodesic neighborhood information (bottom right) of shape points corresponding to pixels of the generated RGB images. The backprojection module integrates several cross-attention blocks (bottom left) used to infer texel features and colors from the appearance and geometric information gathered from relevant, non-background pixels across all available views. As some texels may remain uncolored, an inpainting and high-definition (HD) module is applied to refine the texture map following Paint3D [ZCQ*24].*

3. Method

Given an untextured 3D shape **S**, represented as a polygon mesh, along with its surface parametrization in terms of UV coordinates and a text prompt **t** describing its intended texture, the goal of our method is to generate an albedo texture map i.e., the base RGB color of the object. The texture map **T** is stored as a high-res $H \times W \times 3$ atlas in UV space ($H = W = 1024$ in our implementation). Our overall pipeline is illustrated in Figure 3. Its stages involve: (a) rendering depth maps for the input shape from a set of viewpoints, (b) generating RGB views for these viewpoints through a diffusion model conditioned on the input depth maps, (c) backprojecting the RGB images to the shape's texture space, (d) inpainting and upsampling the map in UV space. In the following sections, we discuss the steps of our pipeline, and in particular the learned backprojection stage, which is our main contribution.

3.1. Depth/edge map rendering & viewpoint selection

As done in several recent texture generation approaches [ZCQ*24, LXLW24, CDG*24, CMZ*24], the first step in our pipeline is to render the mesh into a set of depth maps $\{\mathbf{D}_c\}_{c=1}^{C}$ from various viewpoints, where C is the total number of viewpoints. These maps are used as conditioning to guide the diffusion process to generate images consistent with the depth cues. There have been various strategies for viewpoint selection and diffusion model conditioning – in our paper, we experimented with two backbones: one based on Paint3D [ZCQ*24], and another based on MatAtlas [CDG*24], briefly described below.

Paint3D backbone. Paint3D follows an iterative strategy of viewpoint selection, image generation, and backprojection of the generated images to texture space. First, a couple of 1024×512 depth images are generated from the frontal and rear views of the shape and are concatenated in a 2×1 grid. The grid is passed as input to a diffusion process that generates a corresponding 2×1 grid of RGB images. The use of both views as input to the diffusion model helps with the view consistency [ZCQ*24]. The generated images are backprojected to the shape texture through a simple inverse UV mapping strategy – we discuss backprojection strategies, including ours, in Section 3.3. The next iteration proceeds with two side-wise viewpoints, from which both depth images and partially colored RGB images are rendered from the partially textured mesh. These are provided as a grid to another diffusion process, whose generated images are again backprojected to UV space. The process repeats

for one more step where two other top- and bottom-wise viewpoints are used. In total, three iterations (total $C = 6$ views), with two viewpoints processed at a time, yielded the best results in Paint3D. Our experiments with this backbone follow the same iterative procedure and viewpoints – we only modify the backprojection.

MatAtlas backbone. MatAtlas [CDG*24] follows a different viewpoint selection, diffusion conditioning, and view generation strategy. Initially, a set 400×400 depth maps are rendered from viewpoints uniformly sampled from the viewing sphere, and are arranged in a 4×4 grid. In addition, 16 edge maps are created using the shape's occluding and suggestive contours and are placed also in a grid. These two grids are used as input to the diffusion process that generates a 4×4 grid of RGB images. These are backprojected and blended into the shape's texture space (discussed in Section 3.3). The resulting partially textured shape is rendered from the same viewpoints, and the rendered RGB images along with added partial noise, are passed to a second diffusion process yielding an updated set of RGB views. These are backprojected to the shape's texture space, yielding a sharper texture [CDG*24]. In a third step, additional viewpoints are selected accessing shape regions not textured yet. The textured shape is rendered from these viewpoints and the rendered images are arranged in a grid processed through another diffusion process, which generates another set of RGB images. These are again backprojected to the final shape's texture. In our implementation, we use 6 initial viewpoints to render depth maps at resolution 512×512, arranged in a 3×2 grid (we do not make use of edge maps). The used viewpoints are the same as the ones used in Paint3D for more fair comparisons across the two backbones, and also because we observed that texture details are better preserved from the higher resolution depth maps. We replace the MatAtlas backprojection with ours – the rest of the pipeline follows MatAtlas.

3.2. View generation

Both backbones use a text-to-image stable diffusion model [RBL*22] to generate candidate RGB images based on the input grids. The stable diffusion model gradually denoises a random normal noise image in latent space $\mathbf{z} \in \mathbb{R}^{h \times w \times l}$, where $h = w = 64$ and $l = 4$ are the stable diffusion's latent space dimensions. The outputs of the diffusion model blocks are modulated by a ControlNet network branch [ZRA23], which is conditioned on the encoded text, depth map grid, and, depending on the specific backbone and iteration, on the rendered maps derived from partially textured shapes. The denoised latent is decoded into a grid of images $\{\mathbf{I}_c\}_{c=1}^{C}$ for the selected viewpoints:

$$\{\mathbf{I}_c\}_{c=1}^{C} = \mathcal{D}(\mathbf{z}, \mathbf{t}, \{\mathbf{D}_c\}_{c=1}^{C}, \{\mathbf{G}_c\}_{c=1}^{C}; \tau_t, \tau_d, \tau_g) \tag{1}$$

where $\mathbf{z}$ are noisy latents, $\mathbf{t}$ is the input text, $\{\mathbf{D}_c\}_{c=1}^{C}$ are depth maps, $\{\mathbf{G}_c\}_{c=1}^{C}$ are rendered images from the partially textured shape (used in Paint3D and MatAtlas after the first iteration), τ_t, τ_d, τ_g are encoder networks that produced text, depth, and image representations used as control guidance for the diffusion process.

3.3. Backprojection

The goal of the backprojection is to transfer the generated image colors from all used viewpoints back to the shape's texture map. We first describe how backprojection has been implemented in previous methods, then we discuss our neural approach.

3.3.1. Traditional backprojection

Inverse UV mapping. Previous methods use an inverse UV mapping procedure for backprojection. Specifically, given each texel in the texture map $\mathbf{u} = (u, v) \in \mathbf{T}$, its corresponding 3D surface point $\mathbf{s_u} = (x_\mathbf{u}, y_\mathbf{u}, z_\mathbf{u}) \in \mathbf{S}$ is first estimated. Practically, this can be implemented by rendering a flattened version of the input polygon mesh $\mathbf{S}$ with its vertex coordinates replaced with its texture coordinates. Then for each rendered pixel, its barycentric coordinates are calculated within the flattened triangle it belongs to. These are used to interpolate the 3D vertex positions of this triangle in the original mesh to acquire the corresponding 3D point $\mathbf{s_u}$ for that texel. The procedure assumes that each texture coordinate maps to a single 3D face – if a texture coordinate is re-used by multiple faces, the texture can be unwrapped to avoid this [LPRM02].

Backprojection via most front-facing view. Most previous methods, such as Paint3D [ZCQ*24], Text2Tex [CSL*23], TEXTure [RMA*23], find the view where the 3D point appears to be the most front-facing i.e., the dot product between its normal $\mathbf{n_u}$ and the view vector $\mathbf{v}_c$ is maximized, and simply copy the color from the generated image pixel where the 3D point is projected onto under that view:

$$\mathbf{T}[\mathbf{u}] = \mathbf{I}_{c'}[\mathcal{R}_{c'}(\mathbf{s_u})], \text{ where } c' = argmax_c\left(\mathbf{n_u} \cdot \mathbf{v}_c\right) \tag{2}$$

where $\mathcal{R}_{c'}$ returns the 2D pixel coordinates of the point $\mathbf{s_u}$ rendered onto the image $\mathbf{I}_{c'}$ under the most front-facing viewpoint c' for this point. It is also common to employ a hand-tuned threshold $\mathbf{n_u} \cdot \mathbf{v}_c > thr$ to avoid copying colors from obscure views. We note some texels may not acquire any color, if their corresponding points are not accessible by any acceptable views – texture inpainting is used to fill such texels with color [ZCQ*24]. Unfortunately, this strategy can easily lead to inconsistencies e.g., texels of neighboring 3D points might acquire colors from different views that may not blend well together.

Backprojection via blending views. An alternative strategy, followed by MatAtlas [CDG*24] in its first diffusion iteration, is to average colors from the pixels of all views accessing the texel's corresponding point to blends any small inconsistencies:

$$\mathbf{T}[\mathbf{u}] = \operatorname*{avg}_c \mathbf{I}_c[\mathcal{R}_c(\mathbf{s_u})], \tag{3}$$

Other approaches [ZPZ*24, CMZ*24] implement a weighted averaging scheme, where the weights are the dot product between the 3D point normals $\mathbf{n_u}$ and view vectors $\mathbf{v}_c$. Unfortunately, averaging schemes can yield blurry texture results, as also noted in [CDG*24].

3.3.2. Neural backprojection

Instead of relying on ad hoc, hand-tuned schemes for backprojecting and blending colors from the generated views, we instead propose a learned backprojection scheme. We utilize a neural module based on attention [VSP*17] to assign appropriate colors to each texel by comparing its features with those of pixels gathered from image neighborhoods related to this texel across all views. The texel and pixels features are learned based on positional encodings of the underlying 3D points corresponding to these texels and pixels as well as their underlying appearance (color). The positional encodings incorporate information about their 3D position, normals, angles between normals and view vectors, and surface coordinates encoded in geodesic distances – the reason for using all this information is that the texel color should not be determined by a pixel from a single view, or by merely averaging pixels, but instead by considering broader pixel neighborhoods across all views to maximize view consistency, and by considering texture correlations in local surface neighborhoods according to the underlying 3D geometry to promote texture consistency.

Pixel neighborhoods. For each texel $\mathbf{u}$ and each input view, we collect the $K \times K$ pixel neighborhood centered around the pixel $\mathcal{R}_c(\mathbf{s_u})$, where the texel's corresponding point $\mathbf{s_u}$ is projected onto. We discard any pixels that lie outside the shape's silhouette, i.e., those in the background. The remaining pixels from neighborhoods across all views are then gathered to form a set of pixels $\mathcal{N}(\mathbf{s_u})$. The features from these pixels are used as input to our neural module, which learns to determine the texel's color by identifying relevant pixels from this set. We discuss the choice of K in our experiments. While one could theoretically use a very large K (even the entire image), this would be inefficient and degrade performance. Limiting K to 1, which only includes pixels where the 3D point projects, results in less view-consistent textures in our experiments. We found that smaller neighborhoods ($K = 3$) yield the most consistent textures.

Positional encodings. For each pixel $\mathbf{p} \in \mathcal{N}(\mathbf{s}_u)$, we determine the corresponding 3D surface point projected onto this pixel based on the view the pixel originated from. We then compute a feature vector that encodes the 3D position $\mathbf{s_p}$ and normal $\mathbf{n_p}$ of this surface point relative to the texel's corresponding surface point. Pixels whose 3D locations are closer to the texel's point, or have more similar normals, are expected to have a stronger influence on its color. Additionally, we encode the geodesic distance $\delta_{\mathbf{p},\mathbf{u}}$ between the pixel's surface point and the texel's 3D point. Geodesic distances refine pixel contributions by accounting for true surface proximity unlike Euclidean distances, which may misleadingly suggest closeness e.g., in regions with folds and high-curvature regions (Figure 8). Geodesic distances are computed using the method in [MR12]. The encoding is obtained via a trained MLP using the following features:

$$\mathbf{h_p} = MLP(\mathbf{s_p} - \mathbf{s_u}, \mathbf{n_p} - \mathbf{n_u}, \mathbf{n_p} \cdot \mathbf{v_c}, \delta_{\mathbf{p},\mathbf{u}}) \tag{4}$$

The texel's encoding $\mathbf{h_u}$ is also computed using the same MLP. Since we encode relative positions and normals, the texel itself is represented by zero vectors for position and normal differences, and a geodesic distance of zero. We note that absolute 3D positions and normals are not included in our encodings, as they were found to degrade performance.

Appearance encodings. The texel color should be determined as a function of the pixel color in the extracted neighborhoods, thus we also encode color features used as input to our backprojection module. For each pixel $\mathbf{p} \in \mathcal{N}(\mathbf{s}_u)$, we use an MLP to encode its RGB color into a feature vector $\mathbf{f_p}$. The same MLP is used to encode the texel's current color into $\mathbf{f_u}$, provided it has been initialized from a previous backprojection step. If the texel is empty, we use black color as the input to the MLP.

Cross attention. To compute the texel color, our module employs a cross-attention mechanism that compares the texel's position and color encoding with those of neighboring pixels to determine their contribution towards the texel color. Specifically, we treat the texel as the query and each pixel as a key, applying the following query-key-value transformations:

$$\mathbf{q_u} = \mathbf{Q} \cdot (\mathbf{f_u} + \mathbf{h_u}) \tag{5}$$

$$\mathbf{k_p} = \mathbf{K} \cdot (\mathbf{f_p} + \mathbf{h_p}) \tag{6}$$

$$\mathbf{v_p} = \mathbf{V} \cdot \mathbf{f_p} \tag{7}$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are learned transformations. Note that the positional encodings are added to the rest of the features, as also done in [VSP*17]. Note that in our case, the value transformation involves only the color encodings, as our end goal is to transform pixel colors (rather than position) to texel colors. Based on the query and key transformations, we compute the attention weights, which represent the importance of each pixel in contributing to the texel's color:

$$a_{\mathbf{u},\mathbf{p}} = softmax(\mathbf{q_u} \cdot \mathbf{k_p} / \sqrt{D}) \tag{8}$$

where D is the dimensionality of the feature vectors ($D = 64$ in our implementation). Finally texel features are updated based on the computed attention weights and a residual block:

$$\mathbf{f'_u} = \sum_{\mathbf{p}} a_{\mathbf{u},\mathbf{p}} \mathbf{v_p} + \mathbf{f_u} \tag{9}$$

The computed texel features serve as input to a subsequent cross-attention block – our module applies a total of three attention blocks. The final texel features are then decoded into RGB colors using a trained MLP. Each texel with a non-empty pixel neighborhood is processed through this pipeline. Texels without detected pixel neighborhoods, corresponding to regions inaccessible from any view, remain empty (non-colored); we discuss inpainting for these cases in the next section.

3.4. Texture inpainting and refinement

After backprojection and the final iteration of either backbone, some texels may still remain empty. For texture inpainting, we follow Paint3D's approach: a trained diffusion model fills any texture holes within the UV plane. Additionally, Paint3D's high-definition (HD) diffusion model is subsequently used to further enhances the visual quality of the texture map in UV space. We refer readers to Paint3D [ZCQ*24] for more details, and the authors' implementation for these trained modules. We note that we apply the same

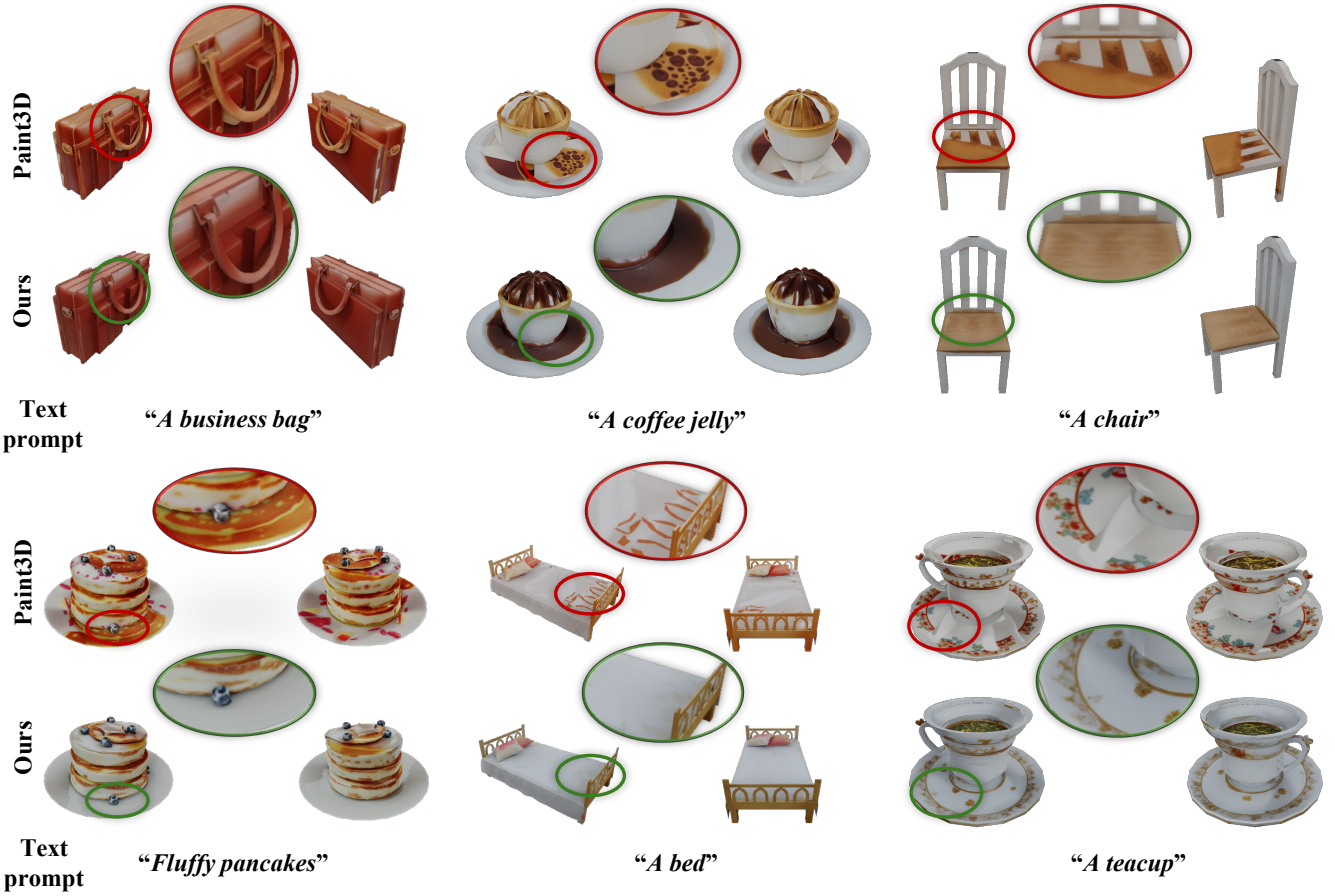

Figure 4: *Comparisons between our method, Im2SurfTex, and Paint3D [ZCQ*24]. Paint3D suffers from view projection artifacts when there are steep depth changes or occluded regions in the input views, as its heuristic best view selection strategy leads to texture discontinuities and inconsistencies. In contrast, our approach generates more seamless and coherent textures.*

inpainting and HD processing for both backbone implementations. Unfortunately, as shown in our experiments, these post-processing modules often fail to correct the artifacts introduced by traditional backprojection.

3.5. Training

We train the parameters of our MLPs and cross-attention module based on supervision from Objaverse [DSS*23]. We use the training split from Paint3D, a subset of the Objaverse dataset containing approximately 100K shapes, each paired with a target texture image. We preprocess the data by computing geodesic neighborhoods for each object, storing the resulting tensors as additional shape-specific information. The network renders input views, which are then used to reconstruct the target texture during training. Our network is trained with a batch size of 4 for 10 epochs on four NVIDIA A6000 GPUs, taking approximately five days. To make our model more robust to any view inconsistencies, we employ a mixed batch approach where some renders are re-generated using a pretrained Stable Diffusion 1.5 model [RBL*22] with partial noise levels ranging from 0.2 to 0.7. Samples with 0.2 noise introduce minor variations, while those with 0.7 noise introduce significant deviations from the target texture. For training, we optimize the model's weights using an L1 loss function between the generated and target texture images.

3.6. Implementation details

During inference, our approach follows either backbone described in Section 3.1, yet incorporating the learned backprojection module instead of their heuristic backprojection. Since some texels remain unfilled after backprojection, they are subsequently inpainted and refined using pretrained Stable Diffusion 1.5 and the Paint3D's 3D-aware ControlNet module. The final output textures have a resolution of 1024×1024. The entire texturing process takes one to two minutes on a single NVIDIA A6000 GPU to texture an input shape. Each iteration of view generation and neural backprojection takes approximately 35 seconds, while the inpainting and high-definition (HD) modules require around 20 seconds each. For preprocessing, our approach employs a one-time procedure to compute geodesic information metadata, which takes approximately 30 minutes when processing a new object for the first time. This part can be significantly accelerated with more efficient techniques for computation of geodesics [CWW13, ZHA*23]. We also refer readers to our project page with source code for more details. †

† *Project page (with code): ygeorg01.github.io/Im2SurfTex*

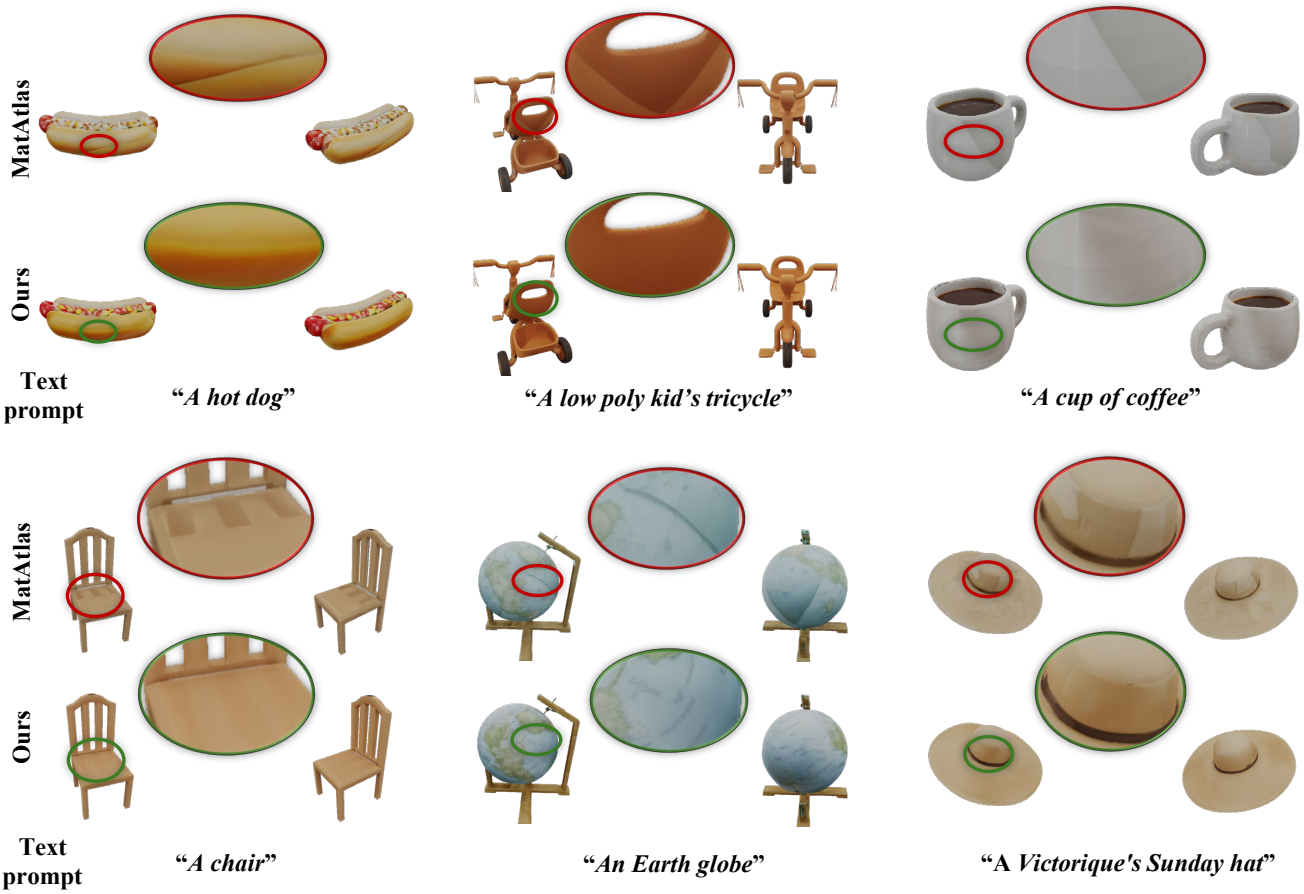

Figure 5: *Comparisons between our method and MatAtlas [CDG*24]. MatAtlas struggles with inconsistencies in the output texture, particularly in regions with high curvature, where misalignments become more apparent. In contrast, as shown in the figure, Im2SurfTex tends to produce more coherent textures.*

4. Evaluation

We evaluate Im2SurfTex on text-to-texture generation both quantitatively and qualitatively. In the following sections, we explain the experimental setup for evaluating our approach, including the evaluation dataset and metrics (Section 4.1). We then compare Im2SurfTex against competing text-to-texture generation methods (Section 4.2). We also analyze the impact of pixel neighborhood sizes, geodesic distances and number of input views on performance in an ablation study (Section 4.3).

4.1. Experimental setup

Test dataset. We evaluate Im2SurfTex on the test split provided by Text2Tex [CSL*23]. The split includes 410 textured meshes from Objaverse [DSS*23] across 225 categories. All competing methods are trained on the same training split, as discussed in Section 3.5, and evaluated on the same above test Objaverse split. We also note that all competing methods use the same UV maps and surface parametrization.

Metrics. For quantitative evaluation, we use standard image quality metrics for generative image models. Specifically, we report the Fréchet Inception Distance (**FID**) [HRU*17] and Kernel Inception Distance (**KID**) [BSAG18]. The FID compares the mean and standard deviation of the deepest layer features in the Inception v3 network between the set of real and generated images. The KID calculates the maximum mean discrepancy between the real and generated images. In practice, the MMD is calculated over a number of subsets to obtain a mean and standard deviation measurement. Additionally, we measure alignment, or similarity, of the generated images with the input text prompt using the **CLIP score** [RKH*21]. To compute these metrics, following [ZCQ*24], we render each mesh with the generated textures from 20 fixed viewpoints at a resolution of 512×512. The reference distribution consists of renders of the same meshes using the textures found in the Objaverse dataset, under identical lighting settings.

4.2. Comparisons

Our main finding – replacing traditional backprojection with our neural module – is numerically examined in Table 1. Our neural module improves both the original Paint3D backbone as well as our implemented MatAtlas backbone. The improvements are consistent across all three evaluation metrics (FID, KID, and CLIP score). Our

Model	FID ↓	KID ↓	CLIP Score ↑
Paint3D	29.13	2.62 ± 0.3	29.45
$\text{Im2SurfTex}_{paint3d}$	27.34	2.12 ± 0.2	29.63
MatAtlas	28.68	2.16 ± 0.2	29.65
$\text{Im2SurfTex}_{matatlas}$	**26.68**	$\mathbf{1.53 \pm 0.2}$	**29.76**

Table 1: *Evaluation using different backbones for viewpoint selection and image generation. Note that the KID metric includes a mean and standard deviation measurement.*

Model	FID ↓	KID ↓	CLIP Score ↑
Text2Tex	34.89	4.82 ± 0.3	29.65
Paint3D	29.13	2.62 ± 0.3	29.45
MatAtlas	28.68	2.16 ± 0.2	29.65
TEXGen	27.41	2.42 ± 0.2	29.23
Im2SurfTex	**26.68**	$\mathbf{1.53 \pm 0.2}$	**29.76**

Table 2: *Comparisons with other text-to-texture methods.*

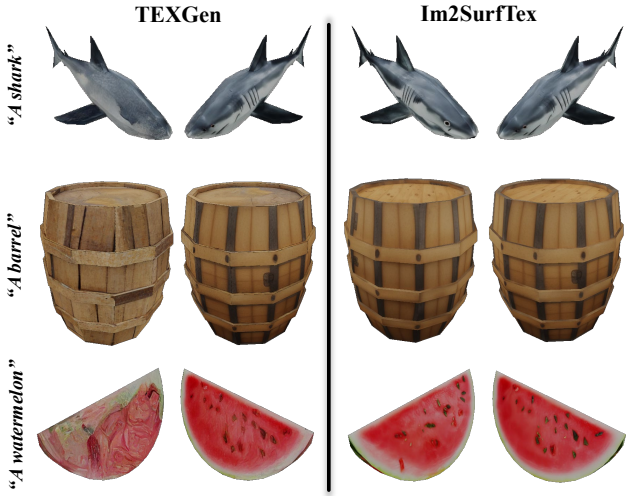

Figure 6: *Comparison between Im2SurfTex and TEXGen [YYG*24]. We show two different views of the textured objects. Our method produces more coherent and view-consistent textures.*

neural backprojection improves the FID distance by 6.1% for the Paint3D backbone, and 6.9% for the MatAtlas backbone. The improvements are more prominent in terms of the KID score (19.1% relative reduction for the Paint3D backbone, and 29.2% relative reduction for the MatAtlas backbone). The KID score is more sensitive to fine-grained texture variations due to the use of Maximum Mean Discrepancy (MMD) with a polynomial kernel when comparing distributions. As a result, when a model improvement primarily reduces local inconsistencies – such as texture artifacts and fine details, KID tends to exhibit a more substantial improvement than FID. In terms of CLIP score, all methods seem to generate images that are similarly aligned with the text prompt, yet our module still maintains a small edge over traditional backprojection.

Figure 4 and 5 provide comparisons of our module against the Paint3D and MatAtlas respectively. Overall, we observe that our texture results have less artifacts and seams, while preserving a similar level of texture detail. We also refer readers to the supplementary material for more results.

In Table 2, we include quantitative comparisons of the best variant of our method (based on the MatAtlas backbone) with other state-of-the-art models for text-to-texture generation. Here we also include a comparison with the recent method of TEXGen [YYG*24]. According to all the evaluation metrics, our method provides the best performance in terms of FID & KID distances as well as CLIP score.

In Figure 6, we show qualiitative comparisons with TEXGen's released implementation [YYG*24]. We observe that TEXGen often leads to global texture inconsistencies on the output shapes, while our method is more view-consistent.

4.3. Ablation

We provide an ablation study where we vary the input pixel neighborhood size extracted from the generated images for each texel. Results are shown in Table 3 for neighborhoods 1×1 , 3×3 , 5×5, and 7×7. Best performance is achieved under the 3×3 neighborhood setting.

Table 4 provides another ablation where we compare using absolute versus relative coordinates in the positional encodings of the Eq. 4, and also examine whether using geodesic distance as additional feature in the positional encodings helps. Relative coordinates enhance performance compared to absolute coordinates, as they provide a more effective encoding for processing the local interactions between neighboring points, regardless of their actual 3D locations. With repect to the use of geodesic distances, we observe rather minor improvements in terms of the numerical scores. We suspect that the small differences are due to the fact that the improvements happen only in small image regions for the shapes of our dataset, where the surface changes rapidly (e.g., folds, handles, high curvature regions), as shown in Figure 8. These small regions seem to have a relatively small effect on the established image quality metrics. Figure 8 demonstrates that adding geodesic distances as features in our module leads to fewer texture artifacts and diminished color bleeding in these regions e.g., see the color bleeding between the bed mattress and wooden frame, or the green leaf and the apple when geodedic distances are not used.

Figure 7 demonstrates a visual comparison between reference textured meshes from our dataset, and reconstructed textures by our method, when we pass as input the rendered images from the original textures. This comparison aims to show whether our method causes any significant color shifting or bleeding while aggregating information from different views. We see that demonstrating our method does not introduce any such discrepancies during neural backprojection.

Table 5 presents the impact of using different number of views on our evaluation metrics for both Paint3D and our method (using the Paint3D backbone). Increasing the number of views from 4 to 6 views results in improvements for the FID, KID, and CLIP cores. Yet, for the maximum number of views (8) in this experiments,

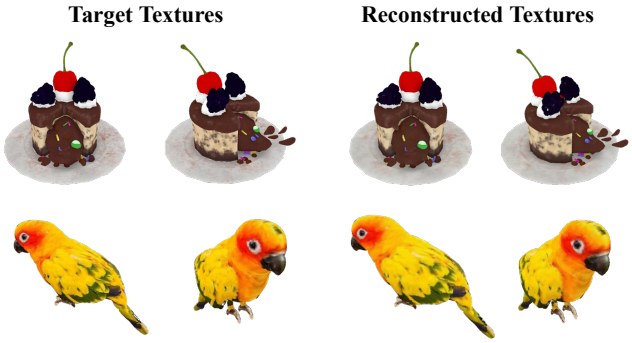

Figure 7: *Our neural backprojection can closely reconstruct challenging textures of target objects in our dataset without causing noticeable color shifting or discrepancies between target and decoded textures.*

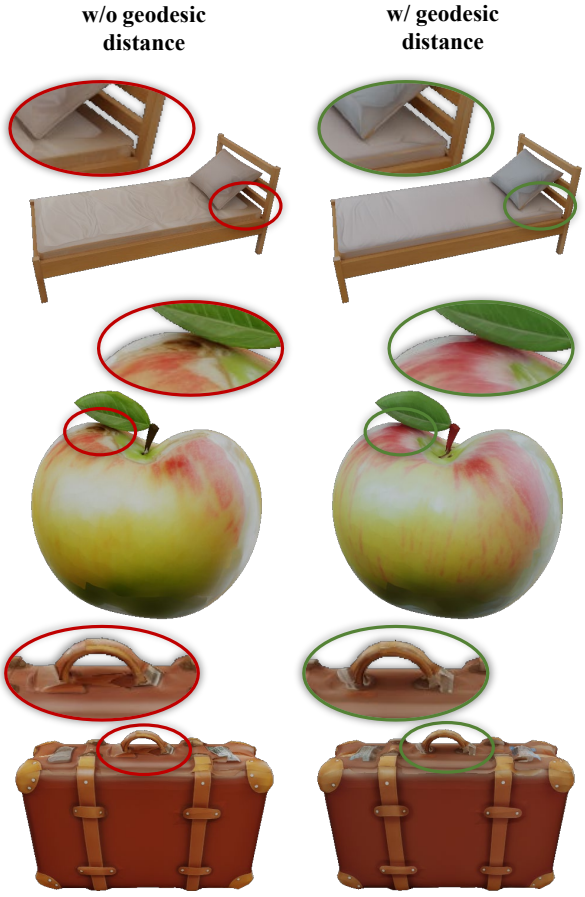

Figure 8: *Using geodesic distances in the positional encodings of texels promote texture consistency. On the left, Im2SurfTex operates without geodesic information, resulting in less coherent textures in areas with rapidly changing local geometry (e.g., surface regions with folds, handles, or high curvature). On the right, incorporating geodesic information improves texture quality in these regions.*

we see that the FID and KID scores do not further improve. As shown in Figure 9, we observe more artifacts appearing in Paint3D,

Window size	FID ↓	KID ↓	CLIPscore ↑
1×1	27.65	2.31 ± 0.2	29.59
3×3	**27.35**	$\mathbf{2.15 \pm 0.2}$	**29.61**
5×5	27.43	2.26 ± 0.2	29.60
7×7	28.12	2.36 ± 0.3	29.54

Table 3: *Ablation study results wrt texel neighborhood size (no geodesic distances are used in this experiment)*

Rel Coords.	Geod. Distances	FID ↓	KID ↓	CLIPscore ↑
-	-	27.86	2.32 ± 0.2	29.62
✓	-	27.35	2.15 ± 0.2	29.61
✓	✓	**27.34**	$\mathbf{2.12 \pm 0.3}$	**29.63**

Table 4: *Ablation study results wrt using geodesic distances or not in the cross-attention operation of our backprojection module. Note that this experiment uses 3×3 pixel neighborhoods.*

# of Views	Method	FID ↓	KID ↓	CLIPscore ↑
4	Paint3D	29.41	2.71 ± 0.3	29.40
	Im2SurfTex	**28.51**	$\mathbf{2.42 \pm 0.3}$	**29.62**
6	Paint3D	29.13	2.62 ± 0.2	29.45
	Im2SurfTex	**27.34**	$\mathbf{2.12 \pm 0.2}$	**29.63**
8	Paint3D	29.20	2.75 ± 0.3	29.48
	Im2SurfTex	**27.86**	$\mathbf{2.32 \pm 0.2}$	**29.62**

Table 5: *Ablation study results wrt using different number of views.*

probably due to its use of mere backprojection which often leads to more seams when more views are backprojected. Our method scores with 8 views are affected less; our neural backprojection produces smoother results, yet we do notice a bit more oversmoothing in our case, which is a limitation our our method.

5. Conclusion & Future Work

In conclusion, Im2SurfTex presents a novel backprojection approach to generating high-quality, coherent textures for 3D shapes from multiview image outputs from pretrained 2D diffusion models. Unlike conventional methods that rely on heuristic and averaging backprojection strategies that introduce texture artifacts and seams, our approach enhances texture continuity and coherence. Experimental results validate the effectiveness of our method.

Limitations and future work. In our current implementation, texture generation is limited by predefined viewpoints that may be suboptimal. Instead, a better approach would be to dynamically adapt to the shape's intrinsic structure. Future work can focus on integrating richer geometric information and utilizing specialized 3D networks to encode complex features such as curvature and occluded regions, which remain challenging for current approaches. By enabling geometry-aware processing in the diffusion process, future methods may further mitigate view-dependent biases.

Acknowledgements. This project has received funding from the European Research Council (ERC) under the Horizon Research

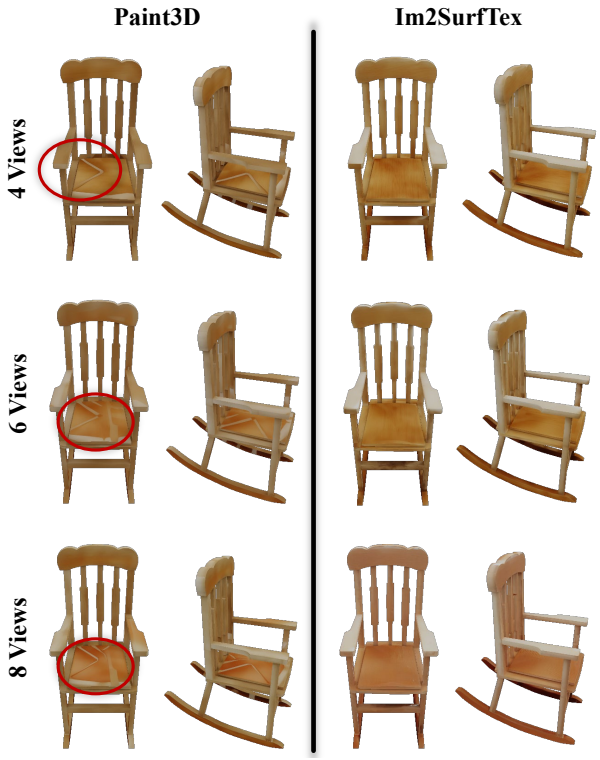

Figure 9: *Results for Paint3D and our method for* 4*,* 6*, and* 8 *input views. Im2SurfTex generates smoother surfaces.*

and Innovation Programme (Grant agreement No. 101124742). Additionally, it has been supported from the EU H2020 Research and Innovation Programme and the Republic of Cyprus through the Deputy Ministry of Research, Innovation and Digital Policy (Grant agreement No. 739578).

References

[Ash01] ASHIKHMIN M.: Synthesizing natural textures. In *Proc. I3D* (2001). 2

[BSAG18] BIŃKOWSKI M., SUTHERLAND D. J., ARBEL M., GRETTON A.: Demystifying MMD GANs. In *Proc. ICLR* (2018). 2, 8

[CDG*24] CEYLAN D., DESCHAINTRE V., GROUEIX T., MARTIN R., HUANG C.-H., ROUFFET R., KIM V., LASSAGNE G.: MatAtlas: Text-driven Consistent Geometry Texturing and Material Assignment. *arXiv preprint arXiv:2404.02899* (2024). 1, 2, 3, 4, 5, 8

[CKF*23] CAO T., KREIS K., FIDLER S., SHARP N., YIN K.: TexFusion: Synthesizing 3D textures with text-guided image diffusion models. In *Proc. ICCV* (2023). 1, 2, 3

[CMZ*24] CHENG W., MU J., ZENG X., CHEN X., PANG A., ZHANG C., WANG Z., FU B., YU G., LIU Z., PAN L.: MVPaint: Synchronized Multi-View Diffusion for Painting Anything 3D. *arXiv preprint arXiv:2411.02336* (2024). 1, 3, 4, 5

[CSL*23] CHEN D. Z., SIDDIQUI Y., LEE H.-Y., TULYAKOV S., NIESSNER M.: Text2Tex: Text-driven Texture Synthesis via Diffusion Models. In *Proc. ICCV* (2023). 1, 2, 3, 5, 8

[CWW13] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics 32*, 5 (2013). 7

[CWWL24] CHEN Z., WANG F., WANG Y., LIU H.: Text-to-3d using gaussian splatting. In *Proc. CVPR* (2024). 3

[DLAH24] DECATUR D., LANG I., ABERMAN K., HANOCKA R.: 3D Paintbrush: Local stylization of 3d shapes with cascaded score distillation. In *Proc. CVPR* (2024). 3

[DSS*23] DEITKE M., SCHWENK D., SALVADOR J., WEIHS L., MICHEL O., VANDERBILT E., SCHMIDT L., EHSANI K., KEMBHAVI A., FARHADI A.: Objaverse: A universe of annotated 3d objects. In *Proc. CVPR* (2023). 7, 8

[EF01] EFROS A. A., FREEMAN W. T.: Image quilting for texture synthesis and transfer. In *Proc. SIGGRAPH* (2001). 2

[EL99] EFROS A. A., LEUNG T. K.: Texture synthesis by non-parametric sampling. In *Proc. ICCV* (1999). 2

[FSDH07] FISHER M., SCHRÖDER P., DESBRUN M., HOPPE H.: Design of tangent vector fields. *ACM Transactions on Graphics (also in the Proc. of SIGGRAPH) 26*, 3 (2007). 3

[GPAM*14] GOODFELLOW I., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative adversarial nets. In *Proc. NeurIPS* (2014). 3

[GSW*22] GAO J., SHEN T., WANG Z., CHEN W., YIN K., LI D., LITANY O., GOJCIC Z., FIDLER S.: Get3d: A generative model of high quality 3d textured shapes learned from images. In *Proc. NeurIPS* (2022). 1

[HGZ*24] HUO D., GUO Z., ZUO X., SHI Z., LU J., DAI P., XU S., CHENG L., YANG Y.-H.: TexGen: Text-Guided 3D Texture Generation with Multi-view Sampling and Resampling. In *Proc. ECCV* (2024). 3

[HJA20] HO J., JAIN A., ABBEEL P.: Denoising diffusion probabilistic models. In *Proc. NeurIPS* (2020). 3

[HRU*17] HEUSEL M., RAMSAUER H., UNTERTHINER T., NESSLER B., HOCHREITER S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proc. NeurIPS* (2017). 2, 8

[HZW*06] HAN J., ZHOU K., WEI L.-Y., GONG M., BAO H., ZHANG X., GUO B.: Fast example-based surface texture synthesis via discrete optimization. *The Visual Computer 22* (2006). 3

[JN23] JUN H., NICHOL A.: Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463* (2023). 1

[KEBK05] KWATRA V., ESSA I., BOBICK A., KWATRA N.: Texture optimization for example-based synthesis. *ACM Transactions on Graphics (also in the Proc. of SIGGRAPH) 24*, 3 (2005). 3

[KKLD23] KERBL B., KOPANAS G., LEIMKÜHLER T., DRETTAKIS G.: 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics (also in the Proc. of SIGGRAPH) 42*, 4 (2023). 3

[KSE*03] KWATRA V., SCHÖDL A., ESSA I., TURK G., BOBICK A.: Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics (also in the Proc. of SIGGRAPH) 22*, 3 (2003). 2

[LGT*23] LIN C.-H., GAO J., TANG L., TAKIKAWA T., ZENG X., HUANG X., KREIS K., FIDLER S., LIU M.-Y., LIN T.-Y.: Magic3D: High-resolution text-to-3d content creation. In *Proc. CVPR* (2023). 1, 3

[LPRM02] LEVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics 21*, 3 (2002). 5

[LSC*24] LIU M., SHI R., CHEN L., ZHANG Z., XU C., WEI X., CHEN H., ZENG C., GU J., SU H.: One-2-3-45++: Fast Single Image to 3D Objects with Consistent Multi-View Generation and 3D Diffusion. In *Proc. CVPR* (2024). 1

[LWVH*23] LIU R., WU R., VAN HOORICK B., TOKMAKOV P., ZAKHAROV S., VONDRICK C.: Zero-1-to-3: Zero-shot one image to 3d object. In *Proc. CVPR* (2023). 1

[LXJ*24] LIU M., XU C., JIN H., CHEN L., VARMA T M., XU Z., SU H.: One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. In *Proc. NeurIPS* (2024). 1

[LXLW24] LIU Y., XIE M., LIU H., WONG T.-T.: Text-Guided Texturing by Synchronized Multi-View Diffusion. In *Proc. SIGGRAPH Asia* (2024). 1, 2, 3, 4

[MR12] MELVÆR E. L., REIMERS M.: Geodesic Polar Coordinates on Polygonal Meshes. *Computer Graphics Forum 31*, 8 (2012). 6

[MRP*23] METZER G., RICHARDSON E., PATASHNIK O., GIRYES R., COHEN-OR D.: Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proc. CVPR* (2023). 3

[MST*20] MILDENHALL B., SRINIVASAN P. P., TANCIK M., BARRON J. T., RAMAMOORTHI R., NG R.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *Proc. ECCV* (2020). 3

[NDR*21] NICHOL A., DHARIWAL P., RAMESH A., SHYAM P., MISHKIN P., MCGREW B., SUTSKEVER I., CHEN M.: Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741* (2021). 3

[NJD*22] NICHOL A., JUN H., DHARIWAL P., MISHKIN P., CHEN M.: Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751* (2022). 1

[PJBM22] POOLE B., JAIN A., BARRON J. T., MILDENHALL B.: Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988* (2022). 3

[RBL*22] ROMBACH R., BLATTMANN A., LORENZ D., ESSER P., OMMER B.: High-resolution image synthesis with latent diffusion models. In *Proc. CVPR* (2022). 3, 5, 7

[RDN*22] RAMESH A., DHARIWAL P., NICHOL A., CHU C., CHEN M.: Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125* (2022). 3

[RKH*21] RADFORD A., KIM J. W., HALLACY C., RAMESH A., GOH G., AGARWAL S., SASTRY G., ASKELL A., MISHKIN P., CLARK J., ET AL.: Learning transferable visual models from natural language supervision. In *Proc. ICML* (2021). 8

[RMA*23] RICHARDSON E., METZER G., ALALUF Y., GIRYES R., COHEN-OR D.: TEXTure: Text-Guided Texturing of 3D Shapes. In *Proc. SIGGRAPH* (2023). 1, 2, 3, 5

[SBV*22] SCHUHMANN C., BEAUMONT R., VENCU R., GORDON C., WIGHTMAN R., CHERTI M., COOMBES T., KATTA A., MULLIS C., WORTSMAN M., SCHRAMOWSKI P., KUNDURTHY S., CROWSON K., SCHMIDT L., KACZMARCZYK R., JITSEV J.: LAION-5B: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402* (2022). 2

[SCS*22] SAHARIA C., CHAN W., SAXENA S., LI L., WHANG J., DENTON E. L., GHASEMIPOUR K., GONTIJO LOPES R., KARAGOL AYAN B., SALIMANS T., ET AL.: Photorealistic text-to-image diffusion models with deep language understanding. In *Proc. NeurIPS* (2022). 3

[SCZ*23] SHI R., CHEN H., ZHANG Z., LIU M., XU C., WEI X., CHEN L., ZENG C., SU H.: Zero123++: a Single Image to Consistent Multi-view Diffusion Base Model. *arXiv preprint arXiv:2310.15110* (2023). 1

[SDWMG15] SOHL-DICKSTEIN J., WEISS E., MAHESWARANATHAN N., GANGULI S.: Deep unsupervised learning using nonequilibrium thermodynamics. In *Proc. ICML* (2015). 3

[SWY*24] SHI Y., WANG P., YE J., MAI L., LI K., YANG X.: MVDream: Multi-view Diffusion for 3D Generation. In *Proc. ICLR* (2024). 3

[TMT*24] TSALICOGLOU C., MANHARDT F., TONIONI A., NIEMEYER M., TOMBARI F.: Textmesh: Generation of realistic 3d meshes from text prompts. In *Proc. 3DV* (2024). 3

[Tur01] TURK G.: Texture synthesis on surfaces. In *Proc. SIGGRAPH* (2001). 3

[TZL*02] TONG X., ZHANG J., LIU L., WANG X., GUO B., SHUM H.-Y.: Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics (also in the Proc. of SIGGRAPH) 21*, 3 (2002). 2

[VSP*17] VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER L., POLOSUKHIN I.: Attention is All you Need. In *Proc. NeurIPS* (2017). 6

[VWG*22] VAHDAT A., WILLIAMS F., GOJCIC Z., LITANY O., FIDLER S., KREIS K., ET AL.: Lion: Latent point diffusion models for 3d shape generation. In *Proc. NeurIPS* (2022). 1

[WDL*23] WANG H., DU X., LI J., YEH R. A., SHAKHNAROVICH G.: Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *Proc. CVPR* (2023). 3

[WL00] WEI L.-Y., LEVOY M.: Fast texture synthesis using tree-structured vector quantization. In *Proc. SIGGRAPH* (2000). 2

[WLKT09] WEI L.-Y., LEFEBVRE S., KWATRA V., TURK G.: State of the Art in Example-based Texture Synthesis. In *Proc. Eurographics - STAR* (2009). 2

[WLW*24] WANG Z., LU C., WANG Y., BAO F., LI C., SU H., ZHU J.: Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *Proc. NeurIPS* (2024). 1, 3

[YGC*24] YANG M., GUO J., CHEN Y., CHEN L., LI P., CHENG Z., ZHANG X., HUANG H.: InstanceTex: Instance-level Controllable Texture Synthesis for 3D Scenes via Diffusion Priors. In *Proc. SIGGRAPH Asia* (2024). 3

[YYG*24] YU X., YUAN Z., GUO Y.-C., LIU Y.-T., LIU J., LI Y., CAO Y.-P., LIANG D., QI X.: TEXGen: a Generative Diffusion Model for Mesh Textures. *ACM Transactions on Graphics (also in the Proc. of SIGGRAPH Asia) 43*, 6 (2024). 3, 9

[ZCQ*24] ZENG X., CHEN X., QI Z., LIU W., ZHAO Z., WANG Z., FU B., LIU Y., YU G.: Paint3D: Paint anything 3D with lighting-less texture diffusion models. In *Proc. CVPR* (2024). 1, 3, 4, 5, 6, 7, 8

[ZHA*23] ZHANG Q., HOU J., ADIKUSUMA Y. Y., WANG W., HE Y.: Neurogf: A neural representation for fast geodesic distance and path queries. In *Proc. NeurIPS* (2023). 7

[ZMT06] ZHANG E., MISCHAIKOW K., TURK G.: Vector field design on surfaces. *ACM Transactions on Graphics (also in the Proc. of SIGGRAPH) 25*, 4 (2006). 3

[ZPIE17] ZHU J.-Y., PARK T., ISOLA P., EFROS A. A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. ICCV* (2017). 3

[ZPZ*24] ZHANG H., PAN Z., ZHANG C., ZHU L., GAO X.: Texpainter: Generative mesh texturing with multi-view consistency. In *ACM SIGGRAPH 2024 Conference Papers* (2024), pp. 1–11. 5

[ZRA23] ZHANG L., RAO A., AGRAWALA M.: Adding Conditional Control to Text-to-Image Diffusion Models. In *Proc. ICCV* (2023). 3, 5

[ZWZ*24] ZHANG L., WANG Z., ZHANG Q., QIU Q., PANG A., JIANG H., YANG W., XU L., YU J.: CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets. *ACM Transactions on Graphics (also in the Proc. of SIGGRAPH) 43*, 4 (2024). 1

[ZZZ*23] ZHANG C., ZHANG C., ZHANG M., KWEON I. S., KIM J.: Text-to-image Diffusion Models in Generative AI: A Survey. *arXiv preprint arXiv:2303.07909* (2023). 3