GenesisTex: Adapting Image Denoising Diffusion to Texture Space

Chenjian Gao[1,2†#] Boyan Jiang[2] Xinghui Li[2] Yingpeng Zhang[2#] Qian Yu[1*]
[1]School of Software, Beihang University
[2]R&D Efficiency and Capability Department, Tencent IEG
https://cjeen.github.io/GenesisTexPaper/

Abstract

We present GenesisTex, a novel method for synthesizing textures for 3D geometries from text descriptions. GenesisTex adapts the pretrained image diffusion model to texture space by texture space sampling. Specifically, we maintain a latent texture map for each viewpoint, which is updated with predicted noise on the rendering of the corresponding viewpoint. The sampled latent texture maps are then decoded into a final texture map. During the sampling process, we focus on both global and local consistency across multiple viewpoints: global consistency is achieved through the integration of style consistency mechanisms within the noise prediction network, and low-level consistency is achieved by dynamically aligning latent textures. Finally, we apply reference-based inpainting and img2img on denser views for texture refinement. Our approach overcomes the limitations of slow optimization in distillation-based methods and instability in inpainting-based methods. Experiments on meshes from various sources demonstrate that our method surpasses the baseline methods quantitatively and qualitatively.

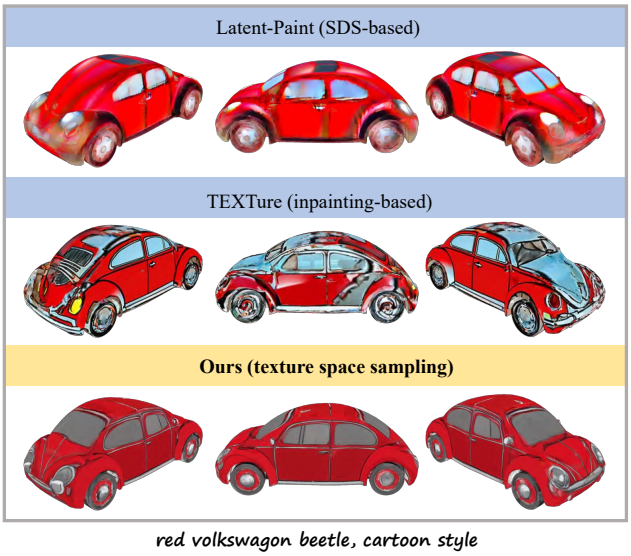

Figure 1. Texturing results of different methods. Score Distillation Sampling (SDS) based method produces blurred and oversaturated textures. Inpainting-based approach results in artifacts at the boundaries of inpainting masks. Texture space sampling concurrently generating content from multiple viewpoints, produces clean, clear and natural colored textures.

1. Introduction

In recent years, with the development of deep learning, 3D content generation technology has made significant progress. The applications of 3D content generation are diverse, ranging from AR/VR to gaming and filmmaking. While there has been considerable research on deep learning-based geometric asset generation[40], there has been a notable industry demand for generating realistic textures for given geometries.

Recently, text-conditioned image diffusion models [34] have achieved impressive results in image generation. Some works [7, 19, 29] have leveraged text-conditioned image diffusion models to generate textured 3D assets by generating content from multiple viewpoints, achieving notable performance. In this work, our focus is on generating high-quality textures for a given geometry, leveraging the priors provided by pre-trained text-to-image diffusion models. This task poses several challenges, including: 1) View consistency: ensuring cross-view constraints for maintaining low-level consistency; 2) High efficiency: generating textures for a model within a few minutes, enabling practical applications; 3) Zero-shot learning: achieving texture generation without requiring additional training or finetuning. Addressing these challenges is crucial for successful application of image diffusion models to the texture domain.

Currently, the most prominent method for text-to-3D generation is Score Distillation Sampling (SDS) [7, 19, 29]. SDS utilizes the prior knowledge from image diffusion

[†]Work was done during an internship at Tencent IEG.
[#]Equal contribution.
[*]Corresponding author.

models to iteratively generate gradients for rendering images of 3D objects. However, SDS has limitations in terms of both efficiency and quality. It often takes several hours to generate a single textured 3D object, and the resulting textures can suffer from over-saturation due to the adoption of a large CFG (classifier-free guidance) scale, which reduces randomness during generation. Recent works [2, 6, 33] have introduced inpainting-based methods for multi-view generation. These methods establish a predetermined order of views, using the content of previously generated views as a condition for subsequent views. While inpainting-based approaches are faster and produce realistic colors, they are sensitive to the predefined view order. Additionally, they require texture segmentation to determine the source view for each texture pixel, which often results in artifacts around the borders of the segmented areas.

To address the aforementioned issues, we propose GenesisTex, a novel approach that introduces texture space sampling. Specifically, we maintain multiple-view latent texture maps throughout the sampling process and perform denoising to improve their quality. Subsequently, the content is decoded from the latent space to obtain RGB textures. Our approach focuses on achieving two aspects of consistency during the sampling process. Firstly, we ensure global style consistency across multiple views by incorporating style consistency in the noise prediction network. This helps maintain a coherent style throughout the generated textures. Secondly, we employ dynamic alignment of latent textures to ensure low-level multi-view consistency, enhancing the overall quality of the generated textures. Due to memory limitations, texture space sampling operates on sparse views. However, to further enhance the quality, we leverage reference-based inpainting and Img2Img techniques on denser views for texture map refinement. GenesisTex can generate detailed, clean, and naturally colored textures for a given geometry within a few minutes. Unlike inpainting-based methods, our proposed texture space sampling concurrently generates content for multiple views without relying on a predefined order. This adaptability makes GenesisTex suitable for various geometries and results in fewer artifacts in the generated textures.

Our contributions can be summarized as follows:

- First, we present a novel method for texture generation, where the core is texture space sampling. This sampling technique allows for the concurrent denoising of latent textures associated with multiple viewpoints.
- Second, we introduce Style Consistency and Dynamic Alignment in texture space sampling for multi-view consistency.
- Third, we conduct a comprehensive study involving numerous 3D objects from various sources. The experimental results demonstrate the superiority of our method over baseline methods.

2. Related Work

Texture Synthesis. Generating textures over 3D surfaces is a challenging problem, as it requires attention to both colors and geometry. Earlier works like AUV-Net [8] and Texturify [38] embed the geometric prior into a UV map or mesh parameterization. Different from them, EG3D [3] and GET3D [11] directly train 3D StyleGANs [15] to generate geometries and textures jointly, where the textures are implicit texture fields. However, these methods either only work on a single category, or demand textured 3D shapes for training without text conditioning, which limits their broad applicability. Recently, [2, 6, 33] use the priors provided by the image diffusion model to synthesize textures. Text2Tex [6] and TEXTure [33] performs inpainting on multiview renderings. TexFusion [2] propose a sequential texture sampling method. Our method predicts multi-view noise concurrently, avoiding the following issues of sequential noise prediction: 1) Each iteration requires adding forward noise to visited regions to match unvisited areas, potentially leading to detail loss. 2) Consistency is only constrained between adjacent viewpoints, with no direct consistency for long-range viewpoints.

Text-to-Image Diffusion Models. Over the past years, the development of several large-scale diffusion models [28, 32, 34, 35] has enabled the production of highly detailed and visually impressive images. These models generate images based on input text prompts. Specifically, Stable Diffusion is trained on a substantial text-image dataset [36] and incorporates a text encoder from CLIP [31] to understand the input prompts. Beyond the basic text conditioning, ControlNet [51] enables the model to condition its denoising network on additional input modalities, such as depth maps. In this work, we utilize ControlNet and Stable Diffusion to provide geometrically-conditioned image priors.

Text-to-3D using 2D Image Diffusion Models. Early works [14, 26, 27] utilize the pretrained CLIP model to maximize the similarity between rendered images and text prompt. However, pioneering works DreamFusion [29] and SJC [46], on the other hand, propose to distill a 2D text-to-image generation model to generate 3D shapes from texts, and many follow-up works [19–21, 23, 25, 30, 47] follow such per-shape optimization scheme. Recently, several methods [4, 12, 13, 22, 24, 37, 42–44, 48] have been proposed to generate consistent multi-view images by using diffusion models. MVDream [37] and SyncDreamer [22] share similar ideas, generating consistent multi-view images via attention layers. However, existing approaches either endure slow optimization processes or depend on separately trained 3D priors, rendering them unsuitable for direct application in texture synthesis. In contrast, our method does not require additional training and can generate results within several minutes.

3. Methodology

Our objective is to generate a texture map $\mathcal{T}$ for a given 3D mesh $\mathcal{M}$, using the provided text as a descriptive condition. Our approach leverages the Stable Diffusion model as the image diffusion model. We begin by introducing some basic concept about Stable Diffusion and texture representation in Sec. 3.1. Then, we introduce a sampling algorithm in texture space for Stable Diffusion in Sec. 3.2. Furthermore, we propose multi-view consistency strategy in Sec. 3.3. Finally we perform refinement on the texture map, which includes Inpainting and Img2Img, as described in Sec. 3.4.

3.1. Preliminary

Stable Diffusion. The diffusion model belongs to a class of generative models that generate data through iterative denoising from random noise. In the sampling process, $\mathbf{z}_{i-1}$ can be sampled from $\mathbf{z}_i$ using a DDIM sampler [39]:

$$\begin{aligned}\hat{\mathbf{z}}_{i\to 0} &= (\mathbf{z}_i - \sqrt{1-\alpha_i}\epsilon_\theta(\mathbf{z}_i, t_i, \mathbf{c}_{text}))/\sqrt{\alpha_i},\\ \mathbf{z}_{i-1} &= \sqrt{\alpha_{i-1}}\hat{\mathbf{z}}_{i\to 0} + \sqrt{1-\alpha_{i-1}-\sigma_i^2}\epsilon_\theta(\mathbf{z}_i, t_i, \mathbf{c}_{text})\\ &\quad + \sigma_i\varepsilon,\end{aligned} \tag{1}$$

where ϵ_θ is a pretrained U-Net for noise prediction and the initial $\mathbf{z}_T \sim \mathcal{N}(0, \mathbf{I})$. In this paper, we specifically employ Stable Diffusion [34]. Stable Diffusion performs denoising in the latent space and employs an autoencoder $\mathcal{D}(\mathcal{E}(\cdot))$ for the conversion between image and latent representations. So $\mathbf{z}_0$ generated by the diffusion process is decoded to the image $\mathcal{D}(\mathbf{z}_0)$ finally.

ControlNet. ControlNet [51] injects low-level control during the denoising process of Stable Diffusion. Our approach employs depth $\mathbf{d}$ as the geometric control condition. The predicted noise of the U-Net with ControlNet is represented as $\epsilon_\theta(\mathbf{z}_i, t_i, \mathbf{c}_{text}, \mathbf{d})$.

Texture Representation. In this paper, we utilize mesh as the 3D representation. Texture map is associated with an UV parameterization of the mesh $\mathcal{M}$. We use *xatlas* [50] to generate the UV parameterization.

Rendering. Given a mesh $\mathcal{M}$, a texture map $\mathcal{T}$ and a viewpoint C, we can use the rendering function $\mathcal{R}$ to get the rendered image $\mathbf{x}^{(img)} = \mathcal{R}(\mathcal{T}; \mathcal{M}, C)$. Similar to [2], we do not consider any lighting and rendering involves solely sampling colors from the texture map. The inverse rendering function $\mathcal{R}^{-1}$ can inverse render the image $\mathbf{x}^{(img)}$ back to a texture map $\mathcal{T}' = \mathcal{R}^{-1}(\mathbf{x}^{(img)}; \mathcal{M}, C)$.

3.2. Sampling in Texture Space

The most straightforward method for sampling on the texture map is to fine-tune the Stable Diffusion model directly on some texture maps, making its distribution more like the distribution of texture maps. However, the distribution of texture maps is much more complex than that of rendering images since different UV parameterizations of the same 3D object correspond to different texture maps. Instead of fine-tuning existing image diffusion models, we adapt them to the texture space for texture map generation.

Algorithm 1 Texture Space Sampling

Input: mesh $\mathcal{M}$, cameras $\{C_1, \dots, C_N\}$
Parameters: Denoising time schedule $\{t_i\}_{i=T}^{0}$, DDIM noise schedule $\{\sigma_i\}_{i=T}^{0}$
$\{\mathbf{z}_{T,n}^{(tex)}\}_{n=1}^{N} \leftarrow \{\mathbf{0}\}$
$\{\mathbf{z}_n^{(bg)}\}_{n=1}^{N} \sim \{\mathcal{N}(\mathbf{0}, \mathbf{I})\}$
$\{\hat{\epsilon}_{T+1,n}\}_{n=1}^{N} \sim \{\mathcal{N}(\mathbf{0}, \mathbf{I})\}$
▷ **denoising stage:**
for $i \in \{T \dots 1\}$ **do**
  for $n \in \{1 \dots N\}$ **do**
    $\varepsilon_n \sim \mathcal{N}(0, \boldsymbol{I})$
    $\mathbf{z}_{i,n}^{(img)} \leftarrow \mathbf{M}_n(\sqrt{\alpha_i}\mathcal{R}(\mathbf{z}_{i,n}^{(tex)}) + \sqrt{1-\alpha_i-\sigma_{i+1}^2}\hat{\epsilon}_{i+1,n})$
      $+(1-\mathbf{M}_n)\mathbf{z}_n^{(bg)} + \sigma_{i+1}\varepsilon_n$
    $\hat{\epsilon}_{i,n} \leftarrow \epsilon_\theta(\mathbf{z}_{i,n}^{(img)}, t_i, \mathbf{c}, \mathbf{d}_n)$
    $\hat{\mathbf{z}}_{0,n}^{(img)} \leftarrow (\mathbf{z}_{i,n}^{(img)} - \sqrt{1-\alpha_i}\hat{\epsilon}_{i,n})/\sqrt{\alpha_i}$
    $\mathbf{z}_{i-1,n}^{(tex)} \leftarrow \mathcal{R}^{-1}(\hat{\mathbf{z}}_{0,n}^{(img)})$
    $\mathbf{z}_n^{(bg)} \leftarrow \sqrt{\alpha_{i-1}}\hat{\mathbf{z}}_{0,n}^{(img)} + \sqrt{1-\alpha_{i-1}-\sigma_i^2}\cdot\hat{\epsilon}_{i,n}$
  end for
  $\{\mathbf{z}_{i-1,n}^{(tex)}\}_{n=1}^{N} = dynamic\_align(\{\mathbf{z}_{i-1,n}^{(tex)}\}_{n=1}^{N})$
end for
▷ **decoding stage:**
for $n \in \{1 \dots N\}$ **do**
  $\mathbf{x}_n^{(img)} = \mathcal{D}(\hat{\mathbf{z}}_{0,n}^{(img)})$
  $\mathbf{x}_n^{(tex)} = R^{-1}(\mathbf{x}_n^{(img)})$
end for
$\mathcal{T} = merge(\{\mathbf{x}_n^{(tex)}\})_{n=1}^{N}$
return Texture map $\mathcal{T}_{sampled}$

Algorithm 1 presents our proposed texture space sampling algorithm. Firstly we define a set of $\mathcal{C}^{(sampling)} = \{C_1, .., C_N\}$ camera views and render the corresponding depth map $\mathbf{d}_n$ in each view. Since we employ a latent diffusion model, our texture space sampling method first perform denoising in the latent texture space and then decode the latent to texture image. In the denoising stage, a dedicated diffusion denoising algorithm is applied to a set of latent texture maps $\{\mathbf{z}_n^{(tex)}\}_{n=1}^{N}$. In the decoding stage, we recover the texture map $\mathcal{T}_{sampled}$ from the latent space.

Denoising Stage. In the denoising stage, we adapt the DDIM from the rendering space to the texture space and utilize the same parameters as DDIM, including denoising time schedule $\{t_i\}_{i=T}^{0}$, DDIM noise schedule $\{\sigma_i\}_{i=T}^{0}$. For each viewpoint, we maintain a separate latent texture which dynamically interact during the denoising process. We begin with $\{\mathbf{z}_{T,n}^{(tex)}\}_{n=1}^{N} = \{\mathbf{0}\}$, the zero latent texture map

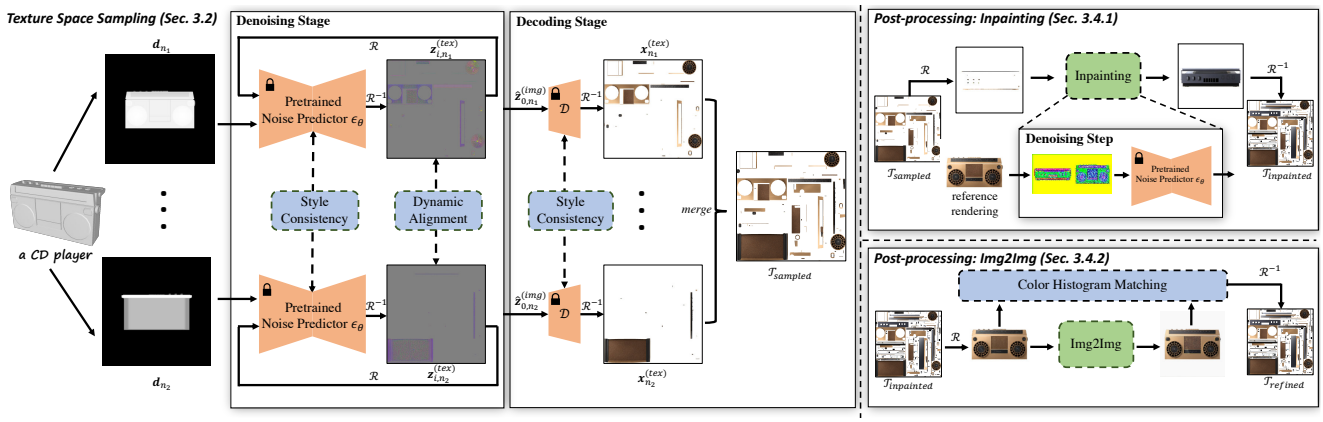

Figure 2. Overview of GenesisTex. GenesisTex generates a texture map for a given mesh $\mathcal{M}$, based on a prompt. Texture Space Sampling samples a texture map using Stable Diffusion, introducing style consistency and dynamic alignment across multiple viewpoints. Furthermore, Inpainting and Img2Img are applied to fill in the blank regions and enhance the quality of texture map details, respectively.

corresponding to N viewpoints. At each denoising step, our goal is to predict $\mathbf{z}_{i-1,n}^{(tex)}$ from $\mathbf{z}_{i,n}^{(tex)}$. We first render the latent texture map $\mathbf{z}_{i,n}^{(tex)}$ to the rendering space using the rendering function $\mathcal{R}$, and then calculate the corresponding latent image $\mathbf{z}_{i,n}^{(img)}$:

$$\begin{aligned}\mathbf{z}_{i,n}^{(img)} &= \mathbf{M}_n(\sqrt{\alpha_i}\mathcal{R}(\mathbf{z}_{i,n}^{(tex)}) + \sqrt{1-\alpha_i-\sigma_{i+1}^2}\hat{\epsilon}_{i+1,n}) \\ &\quad + (1-\mathbf{M}_n)\mathbf{z}_n^{(bg)} + \sigma_{i+1}\varepsilon_n\end{aligned} \tag{2}$$

Here, $\mathbf{M}_n$ represents the foreground mask for viewpoint C_n (downsampled to the same resolution as $\mathbf{z}^{(img)}$), where foreground pixels have a value of 1, and background pixels have a value of 0. $\hat{\epsilon}_{i+1,n}$ represents the noise predicted by the noise prediction network at step $i+1 \rightarrow i$ (notably, we define $\hat{\epsilon}_{T+1,n} \sim \mathcal{N}(0,\mathbf{I})$). $\mathbf{z}_n^{(bg)}$ is the background of the latent image, which is set to Gaussian noise at initialization. $\varepsilon_n \sim \mathcal{N}(0,\mathbf{I})$ is a random Gaussian noise. Subsequently, we predict the corresponding noise $\hat{\epsilon}_{i,n} = \epsilon_\theta(\mathbf{z}_{i,n}^{(img)}, t_i, \mathbf{c}, \mathbf{d}_n)$ for the latent image $\mathbf{z}_{i,n}^{(img)}$ conditioned on the depth map $\mathbf{d}_n$. With the predicted noise, we can obtain the predicted $\hat{\mathbf{z}}_{0,n}^{(img)}$:

$$\hat{\mathbf{z}}_{0,n}^{(img)} = (\mathbf{z}_{i,n}^{(img)} - \sqrt{1-\alpha_i}\hat{\epsilon}_{i,n})/\sqrt{\alpha_i}, \tag{3}$$

Finally, we inverse render $\hat{\mathbf{z}}_{0,n}^{(img)}$ to the texture space using $\mathcal{R}^{-1}$ and update $\mathbf{z}_n^{(bg)}$:

$$\begin{aligned}\mathbf{z}_{i-1,n}^{(tex)} &= \mathcal{R}^{-1}(\hat{\mathbf{z}}_{0,n}^{(img)}), \\ \mathbf{z}_n^{(bg)} &= \sqrt{\alpha_{i-1}}\hat{\mathbf{z}}_{0,n}^{(img)} + \sqrt{1-\alpha_{i-1}-\sigma_i^2}\cdot\hat{\epsilon}_{i,n}\end{aligned} \tag{4}$$

In the denoising stage, we repeat the above process $T-1$ times. After each denoising step, we impose consistency constraints on latent texture $\mathbf{z}_{i-1,n}^{(tex)}$, which will be introduced in Sec. 3.3. Notably, differing from DDIM where the latent includes noise, the latent texture in our method is an estimate of the noise-free version. This is because $\mathcal{R}$ and $\mathcal{R}^{-1}$ in the denoising process require interpolation of the latent, which may distort the noise.

Decoding Stage. To recover the texture map $\mathcal{T}_{sampled}$, we first decode each viewpoint's latent separately to obtain multi-view rendering. Subsequently, we inverse render the renderings into the texture space. After obtaining multi-view texture maps, we merge them into a single texture map as the output. Similar to [2, 6, 33], we aim for each pixel on each texture map to come from the viewpoint where the corresponding point on the mesh is observed most directly. Therefore, our merge is defined as follows:

$$\mathcal{T}_{sampled} = \sum_{n=1}^{N} Softmax(\mathcal{R}^{-1}(\mathbf{N_n})) \times \mathbf{x}_n^{(tex)} \tag{5}$$

$\mathbf{N_n}$ is the similarity mask at viewpoint C_n, where each pixel represents the cosine similarity between the normal vectors of the visible faces and the reversed view direction. We use *Softmax* instead of *Max* because *Softmax* can result in more natural transition in the texture maps.

3.3. Consistency between Latent Texture Maps

3.3.1 Dynamic Alignment

During sampling in texture space, we maintain N latent texture maps, now we introduce an alignment approach to ensure their local consistency. Firstly, we directly reduce these latent texture maps to obtain a uniform latent texture map:

$$\mathbf{z}_{i-1}^{(unitex)} = \sum_{n=1}^{N} Softmax(\mathcal{R}^{-1}(\mathbf{N}_n)) \times \mathbf{z}_{i-1,n}^{(tex)} \tag{6}$$

Next, we blend each viewpoint's latent texture map $\mathbf{z}_{i-1,n}^{(tex)}$ with the uniform latent texture map $\mathbf{z}_{i-1}^{(unitex)}$. We do not enforce strict alignment among the N latent texture maps at every step. Since a latent pixel gains full meaning only when combined with the surrounding context, which is viewpoint-dependent, the latent texture map from each viewpoint should contain some independent information. Moreover, the content generated at different timestamps varies, so the alignment constraints should adapt flexibly. We introduce a signal $c(t)$ for dynamically regulating the variations in consistency during the sampling process:

$$\mathbf{z}_{i-1,n}^{(consistex)} = c(t_i) \times \mathbf{z}_{i-1}^{(unitex)} + (1 - c(t_i)) \times \mathbf{z}_{i-1,n}^{(tex)} \quad (7)$$

We empirically find that maintaining strong local consistency in the mid-term of denoising, while keeping weak local consistency in the early and late stages, leads to higher quality generation results.

3.3.2 Style Consistency

In texture space sampling, we use the noise estimation network ϵ_θ to estimate noise for the multi-view latents, which is responsible for content generation. So we adapt ϵ_θ to ensure style consistency of estimated noise across different views. Inspired by video diffusion models [16, 49] and multi-view diffusion model [37], we modify all the self-attention layers and group normalization layers in the noise prediction network to align the style of multi-view content.

Adapted Self-Attention. In Stable Diffusion, self-attention facilitates long-range interactions among features within an image. We transform self-attention into cross-view attention to establish style consistency across multiple viewpoints, by using the key $\mathbf{K}'$ and value $\mathbf{V}'$ from all viewpoints,

$$\begin{aligned} \mathbf{K}' &= \mathbf{W}^K[\mathbf{z}_{i,1}^{(img)}; ...; \mathbf{z}_{i,n}^{(img)}], \\ \mathbf{V}' &= \mathbf{W}^V[\mathbf{z}_{i,1}^{(img)}; ...; \mathbf{z}_{i,n}^{(img)}]. \end{aligned} \quad (8)$$

where $\mathbf{W}^K$, $\mathbf{W}^V$ are pretrained parameters.

Adapted Group Normalization. Stable diffusion adopt group normalization as the normalization layer. Group normalization divides the channels into groups and computes within each group the mean and variance for normalization. We convert 2D group normalization to 3D by connecting all different views in the group normalization layer, *i.e.*, the mean and variance are calculated within the channel group of all pixels across all viewpoints.

By adjusting self-attention layer and group normalization layer, we achieve multi-view style consistency without any additional training. In addition to the noise prediction network ϵ_θ, we apply the same modifications to the decoder $\mathcal{D}$ to further ensure style consistency in the multi-view decoded RGB images.

3.4. Texture Map Refinement

3.4.1 Post-processing: Inpainting

Since the memory cost of texture space sampling increases with the number of viewpoints of $\mathcal{C}_{sampling}$, the number of viewpoints N is limited. Some regions in the texture map $\mathcal{T}_{sampled}$ may not be observed in any of the viewpoints in $\mathcal{C}_{sampling}$. Therefore, after texture space sampling, we introduce an inpainting epoch to fill texture in areas not observed in $\mathcal{C}$.

Firstly we define a denser set of viewpoints $\mathcal{C}^{(inpainting)} = \{C_1, .., C_{N_1}\}$ as compared to $\mathcal{C}^{(samping)}$, where $N_1 > N$. We use a mask $\mathbf{M}_C^{(blank)}$ to indicate the rendered areas with blank textures at the viewpoint C. The entire inpainting epoch comprises N_1 iterations. In each iteration, we first compute the viewpoint $C_n = \arg\max_C \mathbf{M}_C^{(blank)}$ with the largest blank area. Then, we perform inpainting on the rendering $\mathbf{x}_n^{(img)}$ using the pretrained depth conditioned Stable Diffusion model as described in [6, 33]. During inpainting, Stable Diffusion utilizes regions with textures as conditions to fill in blank areas without textures. To enhance the robustness of inpainting, we render an image using a viewpoint C_0 from the set $\mathcal{C}^{(sampling)}$. We then concatenate this image to the image requiring inpainting as a additional condition for inpainting. Compared to naive inpainting, reference-based inpainting provides a more comprehensive condition. To ensure a natural and smooth transition between blank areas and textured regions, we apply Gaussian blur to the mask $\mathbf{M}_{C_n}^{(blank)}$. Finally the inpainted rendering $\mathbf{x}_n^{(img)\prime}$ is used to update the texture map:

$$\begin{aligned} \mathcal{T}_n = \mathcal{T}_{n-1} &\times (1 - \mathcal{R}^{-1}(\mathbf{M}^{(blank)}) \\ &+ \mathcal{R}^{-1}(\mathbf{x}_n^{(img)\prime}) \times \mathcal{R}^{-1}(\mathbf{M}^{(blank)}) \end{aligned} \quad (9)$$

We start with$\mathcal{T}_0 = \mathcal{T}_{sampled}$ and iterate for N_1 times. Finally we get $\mathcal{T}_{inpainted} = \mathcal{T}_{N_1}$.

It's worth noting that, unlike [2, 6, 33], which relies heavily on inpainting to maintain multi-view consistency, in our method inpainting is performed only to patch small areas. Therefore, the inpainting's performance does not significantly affect the final quality of the generated texture.

3.4.2 Post-processing: Img2Img

So far our generated texture map $\mathcal{T}_{inpainted}$ may still contain unnatural transitions in some regions due to multi-view conflicts and inpainting artifacts. To further improve the generation quality, we introduce an Img2Img epoch.

In Img2Img epoch, we also define a set of viewpoints $C^{(img2img)}$ include N_2 viewpoints and iterate through viewpoint from $C^{(img2img)}$. In each iteration, we encode

the rendering image $\mathbf{x}_n^{(img)}$ to the latent space using the encoder $\mathcal{E}$ and add some noise to the latent rendering. Then we perform denoising using the pretrained depth conditioned Stable Diffusion model. And finally we decode the denoised latent back to image space using $\mathcal{D}$. However, $\mathcal{D}(\mathcal{E}(\cdot))$ is not an identity transformation because of the reconstruction distortion of the autoencoder. So we apply the color histogram matching to the foreground of the decoded image to alleviate the color distortion. Finally we update the texture map using the img2img result $\mathbf{x}_n^{(img)\prime}$ based on view similarity $\mathbf{N}_n$:

$$\begin{aligned}\mathcal{T}_n = \mathcal{T}_{n-1} \times (1 - \mathcal{R}^{-1}(\mathbf{N}_n)) \\ + \mathcal{R}^{-1}(\mathbf{x}_n^{(img)\prime}) \times \mathcal{R}^{-1}(\mathbf{N}_n)\end{aligned} \tag{10}$$

We start with$\mathcal{T}_0 = \mathcal{T}_{inpainted}$. After N_2 iterations we get $\mathcal{T}_{refined} = \mathcal{T}_{N_2}$.

4. Experiments

4.1. Setup

Implementation Details. Our experiments are conducted on an NVIDIA A10 GPU. We utilize the official Stable Diffusion-v1.5 model with the ControlNet-v1.1 (depth). For Inpainting epoch and Img2Img epoch, we use DDIM [39] as the sampler. For all samplers, we set the number of iterations to 20 steps and the CFG scale (classifier-free guidance scale) to 7.5. We implement the rendering function using nvdiffrast [17] and make modifications to nvdiffrast to support inverse rendering. Texture space sampling takes about 3 minutes to generate textures for a single object. For more detailed parameter settings, please refer to the supplementary materials.

Dataset. Similar to TexFusion [2], we utilize 35 meshes to test the texture generation performance, including 11 objects from Objaverse [10], 17 from ShapeNet [5], 1 from Text2Mesh [26], 2 from Turbosquid [1], 2 from Stanford 3D Scans [9, 45] and 2 from Three D Scans [18]. Each object has 1-4 text descriptions, resulting in a total of 80 (mesh, prompt) pairs in this collection.

Baselines. We conduct comparisons with state-of-the-art methods for generating textures from text: 1) **Text2Mesh** [26], a method that stylizes a 3D mesh by predicting color and local geometric details which conform to a target text prompt, harnessing the representational power of CLIP. 2) **Latent-Paint** [25], an approach leveraging SDS to obtain texture gradients from the Stable Diffusion model. 3) **Text2Tex** [6], a method that uses a depth-aware image inpainting diffusion model to incrementally produce partial textures from various viewpoints. 4) **TEXTure** [33], similar to Text2Tex, but utilizes a different region segmentation strategy. 5) **TexFusion** [2], a method sequentially inpainting latent texture. As TexFusion does not have publicly available source code, we extract some rendered images of generated results from their paper for comparison.

Method	FID (↓)	KID (↓) ($\times 10^{-3}$)	User study (%) Visual Quality (↑)	User study (%) Align with Prompt (↑)
Latent-Paint	110.14	10.64	5.15	4.28
Text2Mesh	121.61	15.13	3.15	4.86
Text2Tex	101.38	8.35	18.28	19.72
TEXTure	100.47	9.22	23.42	24.86
Ours	**74.58**	**2.89**	**50.00**	**46.28**

Table 1. Quantitative comparisons with baseline methods.

texture space sampling	Inpainting Round	Img2Img Round	FID(↓)	KID(↓) ($\times 10^{-3}$)
✔	✗	✗	86.25	3.79
✔	✗	✔	82.05	3.67
✔	✔	✗	76.30	3.29
✔	✔	✔	**74.58**	**2.89**

Table 2. Effectiveness of texture refinement.

4.2. Qualitative Analysis

Fig. 3 and Fig. 5 presents visual comparisons between our method and baseline approaches. Textures produced by Text2Mesh are notably lacking in detail. Latent-Paint yields textures with greater intricacy compared to Text2Mesh, but they still exhibit considerable blurriness due to SDS limitations. Inpainting-based approaches like Text2Tex and TEXTure introduce artifacts with unnatural transitions at inpainting mask boundaries, struggling to maintain consistency across multiple views because they depend heavily on inpainting. The output of TexFusion is compromised by blurry details, likely a consequence of repeated noise addition during sequential sampling. In contrast, our method stands out by producing textures that are not only clear but also remarkably coherent.

Fig. 4 presents more results of our method. Benefiting from the powerful prior of Stable Diffusion, our method can generate textures of diverse styles for different geometries.

4.3. Quantitative Comparisons

FID & KID. Similar to TexFusion, we sample from pretrained image diffusion model to create ground truth labels. For each mesh, we generate depth maps from the five most common canonical viewpoints: front, back, top, and both sides. These depth maps, along with textual descriptions, serve as inputs to condition the Stable Diffusion model. To ensure the focus remains on the texture of the objects, we modify the background of the ground truth images to be white. We visualize the textures created by various methods

Figure 3. Qualitative comparisons with Text2Mesh [26], Latent-Paint [25], Text2Tex [6] and TEXTure [33]. In comparison with the baselines, our GenesisTex exhibits richer details and fewer artifacts.

Figure 4. Texturing results with GenesisTex.

Figure 5. Qualitative Comparisons with TexFusion [2]. Images are extracted from their original paper.

from the same five viewpoints and evaluate their quality by calculating the Fréchet Inception Distance (FID) and Kernel Inception Distance (KID). FID and KID measure the feature dissimilarity between two image collections, with feature extraction performed using the Inception V3 [41]. The results in Tab.1 demonstrates that the textures generated by GenesisTex are closer to the ground truth compared to those produced by baselines, indicating a superior synthesis quality.

User Study. We conduct a user study to compare the quality of textures generated by our method versus those produced by baseline methods. We render all textured mesh results into videos to facilitate the presentation to users. Each

video contains a rotating textured mesh along with the corresponding prompt. For each questionnaire, we randomly show the users 10 groups of rendered videos. Each group displays results for a specific (mesh, prompt) pair, comparing outputs from baseline methods and our approach. Each volunteer is required to answer two questions for a single group, including which one has the highest visual quality and which one aligns best with the prompt. Additional information about the questionnaire is available in the supplementary. Finally, we received 35 valid responses from the questionnaires, which included the results of comparisons from 350 groups. The results of the user study are shown in Tab. 1. In terms of both visual quality and alignment with the text prompt, our method is preferred by the most participants, with percentages reaching 50.00% and 46.28%, respectively. Furthermore, we conduct a pairwise comparison study and the results are included in the supplementary.

4.4. Ablation Studies

Consistency in Texture Space Sampling. Our GenesisTex employs style consistency and dynamic alignment for maintaining multi-view consistency during texture space sampling. To investigate the impact of these two consistency strategies, we visualize the decoded multi-view images. We set $\mathcal{C}(elevation, azimuth) = \{(0°, 0°), (0°, 15°), (0°, 35°), (0°, 45°)\}$ to ensure that multiple viewpoints share sufficient content. Fig. 6 illustrates an example with the prompt *penguin toy*. We can observe that without any consistency constraints, the color of the toy varies significant across different viewpoints. After introducing style consistency, consistency for the toy color improves, but there are still inconsistencies in the hair and ears. Further we use dynamic alignment and we can see the consistency of details has significantly improved. We include more results in the supplementary to further demonstrate the effectiveness of these two consistencies.

Figure 6. Ablation results on consistency in texture space sampling. `Style Consistency + Dynamic Alignment` (bottom row) achieves the best multi-view consistency.

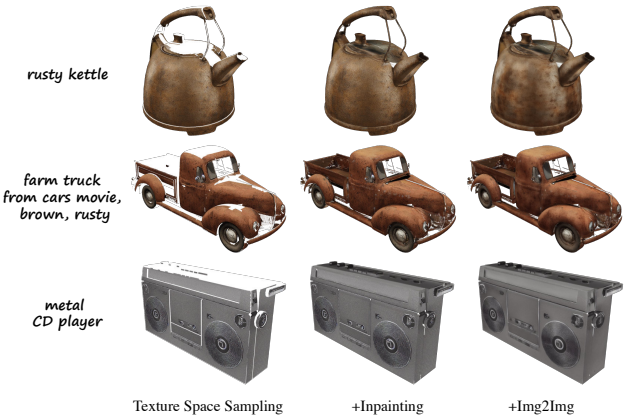

Figure 7. Ablation results on texture refinement. Inpainting fills the blank regions, while Img2Img enhances the quality of details.

Effectiveness of Texture Refinement. After texture space sampling, we conduct refinement on the texture, including Inpainting and Img2Img. We validate the effectiveness of these two modules and the results are presented in Fig. 7 and Tab. 2. Inpainting fills in areas lacking texture obtained during texture space sampling, contributing significantly to the improvement in visual quality. Img2Img further repairs artifacts and enhances details, leading to a certain improvement in visual quality as well.

5. Conclusion and Limitations

In this work, we propose a novel method for text-based texture generation, named *GenesisTex*. GenesisTex leverages the prior of the pre-trained Stable Diffusion model by introducing texture space sampling. Texture space sampling concurrently generates multi-view content without relying on a predefined sequence of views. Our approach can generate high-quality textures for a given 3D model in several minutes. The primary limitation of our method is the significant memory cost associated with maintaining style consistency, caused by cross-view attention. This limitation restricts the number of viewpoints and necessitates post-processing steps, such as inpainting and img2img. Future work could investigate hierarchical style consistency approaches to reduce the computational costs of cross-view attention by iterating over a smaller set of viewpoints.

Acknowledgement: This work is supported by the Young Elite Scientists Sponsorship Program by CAST (China Association for Science and Technology) and IEG Moonshot Program by Tencent.

References

[1] Turbosquid by shutterstock. In *www.turbosquid.com*. 6

[2] Tianshi Cao, Karsten Kreis, Sanja Fidler, Nicholas Sharp, and Kangxue Yin. Texfusion: Synthesizing 3d textures with text-guided image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4169–4181, 2023. 2, 3, 4, 5, 6, 7

[3] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022. 2

[4] Eric R Chan, Koki Nagano, Matthew A Chan, Alexander W Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. Generative novel view synthesis with 3d-aware diffusion models. *arXiv preprint arXiv:2304.02602*, 2023. 2

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 6

[6] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. *arXiv preprint arXiv:2303.11396*, 2023. 2, 4, 5, 6, 7

[7] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv preprint arXiv:2303.13873*, 2023. 1

[8] Zhiqin Chen, Kangxue Yin, and Sanja Fidler. Auv-net: Learning aligned uv maps for texture transfer and synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1465–1474, 2022. 2

[9] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996. 6

[10] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 6

[11] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances In Neural Information Processing Systems*, 35:31841–31854, 2022. 2

[12] Jiatao Gu, Alex Trevithick, Kai-En Lin, Joshua M Susskind, Christian Theobalt, Lingjie Liu, and Ravi Ramamoorthi. Nerfdiff: Single-image view synthesis with nerf-guided distillation from 3d-aware diffusion. In *International Conference on Machine Learning*, pages 11808–11826. PMLR, 2023. 2

[13] Zehuan Huang, Hao Wen, Junting Dong, Yaohui Wang, Yangguang Li, Xinyuan Chen, Yan-Pei Cao, Ding Liang, Yu Qiao, Bo Dai, and Lu Sheng. Epidiff: Enhancing multi-view synthesis via localized epipolar-constrained diffusion, 2023. 2

[14] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 867–876, 2022. 2

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 2

[16] Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2video-zero: Text-to-image diffusion models are zero-shot video generators. *arXiv preprint arXiv:2303.13439*, 2023. 5

[17] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020. 6

[18] Oliver Laric. Three d scans. In *threedscans.com*, 2012. 6

[19] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023. 1, 2

[20] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Zexiang Xu, Hao Su, et al. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *arXiv preprint arXiv:2306.16928*, 2023.

[21] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309, 2023. 2

[22] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. *arXiv preprint arXiv:2309.03453*, 2023. 2

[23] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In *European Conference on Computer Vision*, pages 210–227. Springer, 2022. 2

[24] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. *arXiv preprint arXiv:2310.15008*, 2023. 2

[25] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12663–12673, 2023. 2, 6, 7

[26] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13492–13502, 2022. 2, 6, 7

[27] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 conference papers*, pages 1–8, 2022. 2

[28] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021. 2

[29] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 1, 2

[30] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, et al. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. *arXiv preprint arXiv:2306.17843*, 2023. 2

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2

[32] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1 (2):3, 2022. 2

[33] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. *arXiv preprint arXiv:2302.01721*, 2023. 2, 4, 5, 6, 7

[34] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1, 2, 3

[35] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 2

[36] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294, 2022. 2

[37] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 2, 5

[38] Yawar Siddiqui, Justus Thies, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Texturify: Generating textures on 3d shape surfaces. In *European Conference on Computer Vision*, pages 72–88. Springer, 2022. 2

[39] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 3, 6

[40] Jia-Mu Sun, Tong Wu, and Lin Gao. Recent advances in implicit representation-based 3d shape generation. *Visual Intelligence*, 2(1):9, 2024. 1

[41] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 7

[42] Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Viewset diffusion:(0-) image-conditioned 3d generative models from 2d data. *arXiv preprint arXiv:2306.07881*, 2023. 2

[43] Shitao Tang, Fuyang Zhang, Jiacheng Chen, Peng Wang, and Yasutaka Furukawa. Mvdiffusion: Enabling holistic multi-view image generation with correspondence-aware diffusion. *arXiv preprint arXiv:2307.01097*, 2023.

[44] Hung-Yu Tseng, Qinbo Li, Changil Kim, Suhib Alsisan, Jia-Bin Huang, and Johannes Kopf. Consistent view synthesis with pose-guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16773–16783, 2023. 2

[45] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318, 1994. 6

[46] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12619–12629, 2023. 2

[47] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023. 2

[48] Jianfeng Xiang, Jiaolong Yang, Binbin Huang, and Xin Tong. 3d-aware image generation using 2d diffusion models. *arXiv preprint arXiv:2303.17905*, 2023. 2

[49] Shuai Yang, Yifan Zhou, Ziwei Liu, and Chen Change Loy. Rerender a video: Zero-shot text-guided video-to-video translation. *arXiv preprint arXiv:2306.07954*, 2023. 5

[50] Jonathan Young. xatlas. In *github.com/jpcy/xatlas*, 2016. 3

[51] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 2, 3