Paint-it: Text-to-Texture Synthesis via Deep Convolutional Texture Map Optimization and Physically-Based Rendering

Kim Youwang[1,2,4*] Tae-Hyun Oh[4,5,6] Gerard Pons-Moll[1,2,3]

[1]University of Tübingen [2]Tübingen AI Center, Germany [3]Max Planck Institute for Informatics, Germany
[4]Dept. of Electrical Engineering, POSTECH [5]Grad. School of AI, POSTECH
[6]Institute for Convergence Research and Education in Advanced Technology, Yonsei University

Figure 1. ***Paint-it***. Given an untextured 3D mesh and the text description describing the desired appearance of the 3D mesh, *Paint-it* automatically synthesizes high-fidelity physically-based rendering (PBR) texture maps by neural re-parameterized texture map optimization.

Abstract

We present Paint-it, a text-driven high-fidelity texture map synthesis method for 3D meshes via neural re-parameterized texture optimization. Paint-it synthesizes texture maps from a text description by synthesis-through-optimization, exploiting the Score-Distillation Sampling (SDS). We observe that directly applying SDS yields undesirable texture quality due to its noisy gradients. We reveal the importance of texture parameterization when using SDS. Specifically, we propose Deep Convolutional Physically-Based Rendering (DC-PBR) parameterization, which re-parameterizes the physically-based rendering (PBR) texture maps with randomly initialized convolution-based neural kernels, instead of a standard pixel-based parameterization. We show that DC-PBR inherently schedules the optimization curriculum according to texture frequency and naturally filters out the noisy signals from SDS. In experiments, Paint-it obtains remarkable quality PBR texture maps within 15 min., given only a text description. We demonstrate the generalizability and practicality of Paint-it by synthesizing high-quality texture maps for large-scale mesh datasets and showing test-time applications such as relighting and material control using a popular graphics engine. Project page: https://kim-youwang.github.io/paint-it.

1. Introduction

Crafting realistic and diverse 3D assets is the key component in industrial fields such as movies, games, and AR/VR applications. Professional graphic designers strive to create realistic or creative virtual humans, animals, and objects. Still, the hand-designed generation of realistically textured 3D objects requires cumbersome and time-consuming efforts with intensive labor and the pain of creation.

To reduce such burdens, methods for generating diverse 3D assets have been extensively studied [8, 11, 17, 34, 40, 41, 58]. Notably, the recent progress in neural volumetric representations, *e.g.*, NeRF [38] and diffusion models [46, 47] have advanced the development of text-driven 3D asset generation [26, 28, 33, 42], which leverages a cheaper guidance, *i.e.*, text description. While these methods generate coarse

*Work done during a visiting research period at the University of Tübingen.

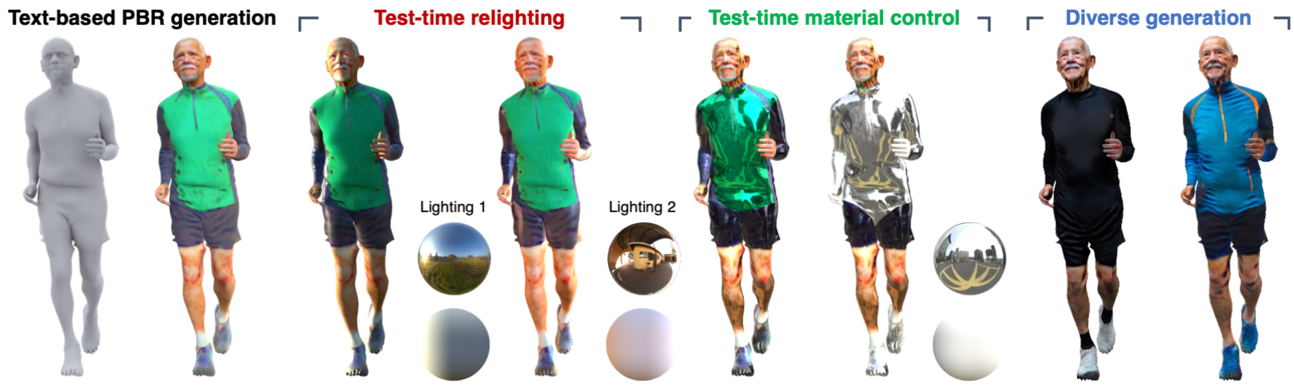

Figure 2. ***Paint-it*: Practical applications.** Using the synthesized PBR texture maps of *Paint-it* and commercial graphics engines, *e.g*., Blender, we can (1) relight the mesh by changing High-Dynamic Range (HDR) environmental lighting (see the balls) and (2) control the material properties at test-time. We can also simulate diverse appearance by synthesizing different PBR texture maps for the same mesh.

geometries and textures, the qualities are still unsatisfactory. Moreover, to use the generated assets in real graphics engines, *e.g*., Blender, one must convert the implicit volumetric geometries and textures into the explicit mesh and compatible texture maps, which makes them impractical. Manual extraction of mesh surfaces and unwrapping of textures could be performed, but it still has limitations. The unwrapped texture maps inevitably have heterogeneous texture mappings, so we cannot easily transfer or edit them, which is crucial for generating diverse 3D assets.

Recently, a line of work [6, 9, 35, 45] has been tackling the task of text-driven texture synthesis for practical use. Instead of generating an entire geometry and texture from scratch, the task aims at synthesizing diverse texture maps on top of the given mesh, conditioned on a text description. While many 3D meshes can be reused in the practical production pipeline, the texture maps should be diverse. For example, a single car mesh can be repeatedly used for making a game, but artists should create distinct texture maps to model different appearances. In this vein, text-driven texture map synthesis tries to revolutionize the current repetitive and exhaustive appearance modeling pipeline. However, existing methods [6, 9, 45] are limited in that they first conditionally generate latent or RGB images and back-project the colors onto the mesh. Although the back-projected colors may look plausible, they may introduce implausible textures since the back-projection cannot model material properties or the complicated reflections on the 3D surfaces.

To address these difficulties, we propose *Paint-it*, which synthesizes high-fidelity texture maps, given a mesh without texture and the text description via synthesis-through-optimization. The main contribution of our work is the analysis and investigation of a proper texture representation, which allows easier optimization with the Score-Distillation Sampling (SDS) [42]. When optimizing the texture maps, we introduce DC-PBR, the Deep Convolutional Physically-Based Rendering re-parameterization. We optimize the neural surrogate of the physically-based rendering (PBR) texture maps rather than directly optimizing the pixel values of the texture maps. DC-PBR formulates coupled optimization variables with diverse frequencies and serves as an implicit texture prior. In our analysis, we show that the DC-PBR, which uses randomly initialized convolution-based neural kernels, naturally imposes the frequency-scheduled learning, which helps filter out high-frequency noisy SDS signals during the optimization. Furthermore, since DC-PBR re-parameterizes the disentangled texture maps; diffuse, roughness & metalness, and normal maps, we simulate physical properties such as the bidirectional reflectance distribution function (BRDF), yielding photorealistic synthesis results. Overall, we observe that the proposed DC-PBR better interacts with the SDS loss than the diffuse-only texture representation.

In experiments, we demonstrate that *Paint-it* produces high-quality texture maps for general 3D meshes, *e.g*., objects, humans, and animals (Fig. 1). Also, *Paint-it* synthesizes a remarkable quality of texture maps compared to competing methods. As a favorable byproduct, the synthesized PBR texture maps are compatible with the popular graphics engine and can be seamlessly integrated into relighting and material control pipelines (Fig. 2). We summarize our main contributions as follows.

- We propose *Paint-it*, a text-driven synthesis of high-fidelity PBR texture maps, which support practical test-time applications compatible with graphics engines.
- We identify the difficulties of synthesizing PBR texture maps in pixel-based parameterization.
- We introduce DC-PBR, a deep convolutional PBR texture map re-parameterization, and empirical analysis of DC-PBR's benefit when optimizing with the noisy signal, *e.g*., Score-Distillation Sampling (SDS).

2. Related Work

Text-driven 3D Asset Generation. Recently, a few impressive works have proposed remarkable 3D asset generation methods that require only simple text prompts [9, 10, 12, 24, 26–28, 33, 37, 42, 62]. Due to the absence of large-scale $\{$*text*, *3D asset*$\}$-paired datasets, most methods exploit indirect supervision signals by rendering the current estimate of the 3D asset into 2D images in multiple views and measuring similarity losses between the rendered images and the given input text. For measuring the similarity as losses, the vision-language joint embedding space, *e.g.*, CLIP [43], or text-conditional generative models, *e.g.*, text-to-image diffusion model [46, 47], are used. This enables per-instance generation by optimization without any paired fully supervised data, *i.e.*, synthesis-through-optimization.

With the synthesis-through-optimization framework, text-driven 3D asset generation methods are categorized into volume- and mesh-based methods. Volume-based methods [7, 24–28, 35, 42, 55, 57] optimize the characteristics, *e.g.*, occupancy, signed distance function (SDF), and color, of the points in a 3D space. Mesh-based methods [9, 37, 62] model geometry with explicit meshes and generate vertex textures or texture maps. Using meshes allows rasterization for faster and more efficient rendering, in contrast to volumetric rendering used in the volume-based ones. Also, meshes are well-compatible with graphics engines and suitable for texture transfer and animation. This contrasts the volume representation that requires separate post-processing to extract mesh and unwrap a texture map by dedicated methods. Thus, 3D designers prefer mesh representation due to its practicality. Recently, hybrid methods [10, 33] were suggested, but they eventually perform re-meshing and texture unwrapping after the 3D volume optimization, which introduces substantial texture seams and loses editability.

Our work chooses mesh representation for synthesizing realistic or aesthetic 3D assets in high fidelity with practical compatibility. Specifically, we focus on texture map synthesis, where we can obtain photorealistic renderings with fast and stable optimization.

Text-driven Texture Map Synthesis. While texture maps are the most commonly used for the graphics pipeline, there are only a few works that generate high-quality texture maps [9, 10, 35, 45, 50]. Text2Tex [9] and TEXTure [45] are analogous, where they generate a RGB image using a pre-trained text and depth-conditioned diffusion model. Since they use color back-projection from the image onto the texture map, their texture maps are limited in diffuse RGB domain. Also, they need additional masking methods to carefully distinguish which part to update. Latent-Paint [35] and TexFusion [6] propose optimizing a latent feature texture map. However, they can also decode RGB colors only due to the dependency of the pre-trained model that can only produce RGB, yielding limited photorealism. Fantasia3D [10] optimizes higher-dimensional physically-based rendering (PBR) materials and generates photorealistic text-driven 3D assets. They estimate per-point PBR material, rather than the spatially structured texture maps we use, and yield non-smooth textures. Our *Paint-it* optimizes the neural re-parameterized PBR material maps and obtains smooth and photorealistic 3D assets. Moreover, instead of generating an image and inpainting the texture map with low-dimensional colors, we directly synthesize the DC-PBR texture map; thus, we do not perform re-meshing or texture unwrapping for each mesh, so it is naturally compatible with applications, *e.g.*, texture transfer or mesh animation.

3. *Paint-it*: Text-Driven PBR Texture Synthesis via Neural Re-parameterized Optimization

3.1. Preliminary: Score-Distillation Sampling

The Score-Distillation Sampling (SDS) [42] iteratively samples the 3D representation θ to generate an image that conforms to the input text description y. Suppose there is a 3D representation, *e.g.*, NeRF [38], parameterized as θ, and we can render it into an image $\mathbf{x}$ using a differentiable renderer, $g(\cdot)$, *i.e.*, $\mathbf{x} = g(\theta)$. To perform SDS, we first perturb the rendered image $\mathbf{x} = g(\theta)$ to make the noisy image $\mathbf{x}_t$ by sampling a noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ and a noising timestep $t \sim \mathcal{U}(0, 1)$. Initially, the rendered image $\mathbf{x}$ would not look like an object described in the text prompt y. Thus, given a pre-trained text-conditional noise estimator ϵ_ϕ, where ϕ denotes the parameters of the pre-trained diffusion model, the error between the added noise ϵ and the text-conditioned estimated noise $\hat{\epsilon}_\phi(\mathbf{x}_t; y, t)$, *i.e.*, $\|\hat{\epsilon}_\phi(\mathbf{x}_t; y, t) - \epsilon\|_2^2$, would be large. On the contrary, if θ is well generated, and its rendering $\mathbf{x}$ conforms to the text prompt and in the distribution of the training image, the error would be minimized.

Poole *et al.* [42] formulate this intuition into an optimization problem, $\theta^* = \arg\min_\theta L_{\text{diff}}(\phi, \mathbf{x}{=}g(\theta))$, where $L_{\text{diff}}(\phi, \mathbf{x}{=}g(\theta)) = \mathbb{E}_{t,\epsilon}[m(t)\|\hat{\epsilon}_\phi(\mathbf{x}_t; y, t) - \epsilon\|_2^2]$. Thus, the update gradient for the 3D representation θ is written as:

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\phi, \mathbf{x}) = \mathbb{E}_{t,\epsilon}\left[m(t)(\hat{\epsilon}_\phi(\mathbf{x}_t; y, t) - \epsilon)\frac{\partial \mathbf{x}}{\partial \theta}\right], \quad (1)$$

where $m(t)$ denotes a weighting function conditioned on the diffusion noise timestep t. This enables obtaining 3D from a text through 2D rendering even without any $\{$text, 3D$\}$-paired dataset. We will use this gradient estimate to optimize the texture maps in a text-conditioned manner.

3.2. Goal of *Paint-it*

Paint-it aims to synthesize high-fidelity physically-based rendering (PBR) texture maps for a given mesh and a text description so that the resulting texture maps visually conform to the text description. Given a 3D mesh without texture, $\mathbf{M}$,

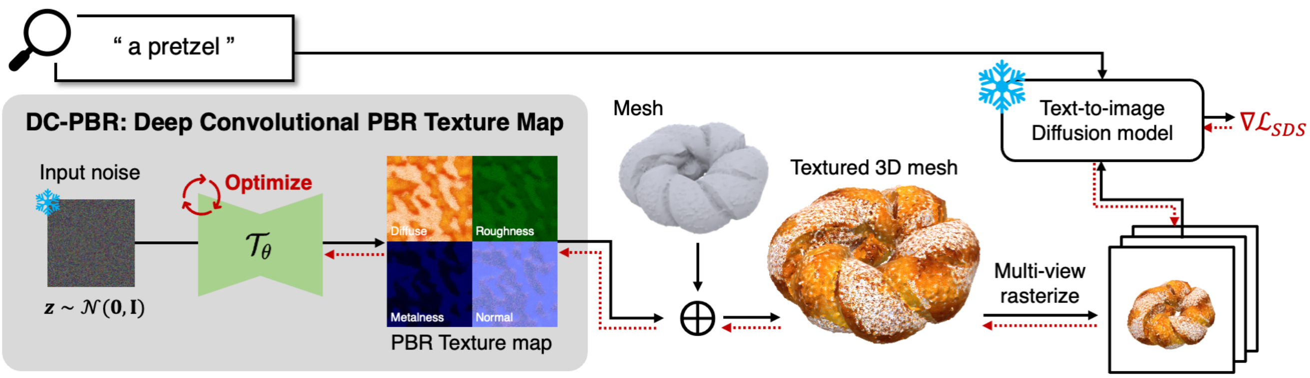

Figure 3. ***Paint-it*: Overall pipeline**. Given a 3D object mesh without a texture and a text describing the desired appearance of the mesh, *Paint-it* synthesizes realistic PBR texture maps via synthesis-through-optimization. We introduce DC-PBR, which parameterizes the PBR texture map into randomly initialized U-Net convolutional neural kernels. By performing texture mapping to texturize the given mesh, we differentiably rasterize the textured mesh and obtain multi-view images, then compute the diffusion-guided loss. Note that *Paint-it* optimizes the neural parameters of the U-Net rather than directly optimizing the pixel values of the texture map.

and a text description y describing the desired appearance of the mesh, our goal is to synthesize the PBR texture maps consisting of diffuse $\mathbf{K}^{\mathrm{d}}$, roughness & metalness $\mathbf{K}^{\mathrm{rm}}$, and detail surface normal $\mathbf{K}^{\mathrm{n}}$ representations. After synthesizing the PBR material texture maps, we can perform texture mapping to obtain a text-conforming textured mesh.

3.3. DC-PBR: Deep Convolutional PBR Texture Map Re-parameterization

We propose the deep convolutional re-parameterization of PBR texture maps, DC-PBR, $\mathcal{T}_\theta$. Instead of the pixel value parameterization of texture maps, using DC-PBR helps to sidestep the optimization difficulty of pixel-based representation, which will be discussed later in Sec. 4. We use a *randomly initialized* convolutional U-Net with skip connections for $\mathcal{T}_\theta$ and use the randomly sampled code $\mathbf{z}{\sim}\mathcal{N}(0, \mathbf{I}) \in \mathbb{R}^{H\times W\times 3}$ as a fixed input, where H and W are the height and width of the target texture maps, respectively, and $\mathbf{z}$ is fixed during optimization. With this, we re-parameterize the pixel-wise PBR parameters of the texture maps with the neural convolution kernels of the $\mathcal{T}_\theta$, *i.e.*, $[\mathbf{K}^{\mathrm{d}}_\theta, \mathbf{K}^{\mathrm{rm}}_\theta, \mathbf{K}^{\mathrm{n}}_\theta] = \mathcal{T}_\theta(\mathbf{z})$, where $\mathbf{K}^{\mathrm{d}}_\theta, \mathbf{K}^{\mathrm{n}}_\theta \in \mathbb{R}^{H\times W\times 3}$, $\mathbf{K}^{\mathrm{rm}}_\theta \in \mathbb{R}^{H\times W\times 2}$, and thus $\mathcal{T}_\theta(\mathbf{z}) \in \mathbb{R}^{H\times W\times(3+2+3)}$.

3.4. Text-driven DC-PBR Optimization

Given a randomly initialized DC-PBR $\mathcal{T}_\theta$ of the PBR texture maps, we perform an iterative optimization aid by the pre-trained text-to-image diffusion model.

Overall Pipeline. We visualize the *Paint-it* optimization pipeline in Fig. 3. At each iteration, we first feed the fixed noise $\mathbf{z}$ to $\mathcal{T}_\theta$ and obtain the predictions of diffuse, roughness & metalness and normal maps; $\mathbf{K}^{\mathrm{d}}_\theta, \mathbf{K}^{\mathrm{rm}}_\theta$, and $\mathbf{K}^{\mathrm{n}}_\theta$. We then rasterize the given mesh by texturing with the obtained texture maps. After rendering multi-view images of the textured mesh, we use the text-guided diffusion model to compute the update direction, $\nabla\mathcal{L}_{\mathrm{SDS}}$, for the neural parameter θ.

Rendering Mesh with PBR Texture Maps. Given the output PBR texture maps, *i.e.*, $\mathbf{K}^{\mathrm{d}}_\theta, \mathbf{K}^{\mathrm{rm}}_\theta, \mathbf{K}^{\mathrm{n}}_\theta$, we texture the given mesh and perform differentiable rasterization to obtain rendered mesh images. To render mesh surfaces, the diffuse $\mathbf{k}^{\mathrm{d}}_\theta \in \mathbb{R}^3$, roughness $\mathrm{k}^{\mathrm{r}}_\theta \in \mathbb{R}$, metalness $\mathrm{k}^{\mathrm{m}}_\theta \in \mathbb{R}$, and perturbing normal direction $\mathbf{k}^{\mathrm{n}}_\theta \in \mathbb{R}^3$ of a 3D surface point $\mathbf{p}$ can be indexed from $\mathbf{K}^{\mathrm{d}}_\theta, \mathbf{K}^{\mathrm{rm}}_\theta$, and $\mathbf{K}^{\mathrm{n}}_\theta$, using the *uv* coordinates. We can use the pre-defined *uv* coordinates of the given mesh or perform unwrapping to generate the *uv* coordinates. We compute the specularity $\mathbf{k}^{\mathrm{s}}_\theta \in \mathbb{R}^3$ as: $\mathbf{k}^{\mathrm{s}}_\theta = 0.04\cdot(1-\mathrm{k}^{\mathrm{m}}_\theta) + \mathrm{k}^{\mathrm{m}}_\theta \cdot \mathbf{k}^{\mathrm{d}}_\theta$. The rendered color L of the mesh surface point $\mathbf{p}$, seen from the view direction $\boldsymbol{\omega}$, can be computed using the rendering equation as:

$$L_\theta(\mathbf{p}, \boldsymbol{\omega}) = \int_\Omega L_i(\mathbf{p}, \boldsymbol{\omega}_i) f_\theta(\mathbf{p}, \boldsymbol{\omega}_i, \boldsymbol{\omega})\,(\boldsymbol{\omega}_i \cdot \mathbf{n}_\theta)\,d\boldsymbol{\omega}_i, \quad (2)$$

where $\boldsymbol{\omega}_i$ denotes the incident light direction, Ω is a hemisphere around the perturbed surface normal $\mathbf{n}_\theta$, and L_i is the incident light from an off-the-shelf environment map. Also, $f_\theta(\mathbf{p}, \boldsymbol{\omega}_i, \boldsymbol{\omega})$ is the bidirectional reflectance distribution function (BRDF) of the material at 3D surface point $\mathbf{p}$. We model the BRDF according to the PBR representation, $\mathbf{k}^{\mathrm{d}}_\theta$, $\mathbf{k}^{\mathrm{s}}_\theta$, and $\mathbf{k}^{\mathrm{n}}_\theta$, which is parameterized by our DC-PBR.

Using the renowned Cook-Torrance microfacet specular shading model [14], we can decompose Eq. (2) into the diffuse term $L_{\mathrm{d}_\theta}(\mathbf{p})$ and the specular term $L_{\mathrm{s}_\theta}(\mathbf{p}, \boldsymbol{\omega})$ as:

$$L_\theta(\mathbf{p}, \boldsymbol{\omega}) = L_{\mathrm{d}_\theta}(\mathbf{p}) + L_{\mathrm{s}_\theta}(\mathbf{p}, \boldsymbol{\omega}),$$

$$L_{\mathrm{d}_\theta}(\mathbf{p}) = \mathbf{k}^{\mathrm{d}}_\theta(1 - k^{\mathrm{m}}_\theta)\int_\Omega L_i(\mathbf{p}, \boldsymbol{\omega}_i)\,(\boldsymbol{\omega}_i \cdot \mathbf{n}_\theta)d\boldsymbol{\omega}_i,$$

$$L_{\mathrm{s}_\theta}(\mathbf{p}, \boldsymbol{\omega}) = \int_\Omega \frac{D_\theta F_\theta G_\theta}{4(\boldsymbol{\omega}\cdot\mathbf{n}_\theta)(\boldsymbol{\omega}_i \cdot \mathbf{n}_\theta)} L_i(\mathbf{p}, \boldsymbol{\omega}_i)(\boldsymbol{\omega}_i \cdot \mathbf{n}_\theta)d\boldsymbol{\omega}_i,$$

where D_θ, F_θ, and G_θ denote the microfacet distribution, Fresnel term, and geometric attenuation function, respectively. Note that D_θ, and G_θ are the functions of the generated $\mathrm{k}^{\mathrm{r}}_\theta$, and F_θ is the function of the specularity, $\mathbf{k}^{\mathrm{s}}_\theta$.

Iterating all the surface points, we obtain the image of the rendered mesh, $\mathbf{I}_\theta$. For simplicity, we denote the aforementioned rendering process for mesh $\mathbf{M}$ as $\mathbf{I}_\theta = \mathcal{R}^{\mathbf{M}}(\mathbf{K}^{\mathrm{d}}_\theta, \mathbf{K}^{\mathrm{rm}}_\theta, \mathbf{K}^{\mathrm{n}}_\theta)$, where $\mathcal{R}^{\mathbf{M}}(\cdot)$ denotes the differentiable mesh rendering function, which we use NVDiffRast [31].

Diffusion-guided DC-PBR Optimization. We obtain noisy PBR texture maps for the initial iteration of the optimization since the DC-PBR $\mathcal{T}_\theta$ is randomly initialized. We use the Score-Distillation Sampling (Eq. (1)) to iteratively update the neural re-parameterized PBR texture maps, $\mathcal{T}_\theta$. Our optimization problem can be written as follows:

$$\theta^* = \arg\min_\theta \mathbb{E}_{t,\epsilon}\left[\|\hat{\epsilon}_\phi(\mathcal{R}^{\mathbf{M}}_t(\mathbf{K}^{\mathrm{d}}_\theta, \mathbf{K}^{\mathrm{rm}}_\theta, \mathbf{K}^{\mathrm{n}}_\theta); y, t) - \epsilon\|^2_2\right], \quad (3)$$

where ϕ denotes the parameters of the pre-trained diffusion model, $\mathcal{R}^{\mathbf{M}}_t(\mathbf{K}^{\mathrm{d}}_\theta, \mathbf{K}^{\mathrm{rm}}_\theta, \mathbf{K}^{\mathrm{n}}_\theta)$ denotes the noisy image perturbed with forward diffusion process, respectively. We omit the t-dependent weighting function $m(t)$ for notation simplicity. Given an image $\mathbf{I}_\theta$ rendered from the textured mesh, we compute the SDS update gradient for updating the neural re-parameterized texture maps as follows:

$$\nabla_\theta \mathcal{L}_{\mathrm{SDS}}(\phi, \mathbf{I}_\theta) = \mathbb{E}_{t,\epsilon}\left[(\hat{\epsilon}_\phi(\mathbf{I}_{\theta,t}; y, t) - \epsilon)\frac{\partial \mathbf{I}_\theta}{\partial \theta}\right]$$
$$= \mathbb{E}_{t,\epsilon}\left[\{\hat{\epsilon}_\phi(\mathcal{R}^{\mathbf{M}}_t(\mathbf{K}^{\mathrm{d}}_\theta, \mathbf{K}^{\mathrm{rm}}_\theta, \mathbf{K}^{\mathrm{n}}_\theta); y, t) - \epsilon\}\frac{\partial \mathbf{I}_\theta}{\partial \theta}\right].$$

The iterative update of DC-PBR $\mathcal{T}_\theta$ with $\nabla_\theta \mathcal{L}_{\mathrm{SDS}}$ finally yields a solution θ^*, and we obtain high-quality PBR texture maps as: $[\mathbf{K}^{\mathrm{d}}_{\theta^*}, \mathbf{K}^{\mathrm{rm}}_{\theta^*}, \mathbf{K}^{\mathrm{n}}_{\theta^*}] = \mathcal{T}_{\theta^*}(\mathbf{z})$.

4. Analysis: Effect of the Deep Convolutional Re-parameterization for PBR Texture Maps

4.1. Analysis of Fitting Behavior

We observe that the SDS loss is noisy, including notable randomness. To analyze, we first design a simple experiment focusing on the parameterization by excluding the influence of the randomness induced by the diffusion model.

Methods. We compare the optimizations on pixel values and neural parameters as: 1) *Pixel Optimization*: The most direct way to fit an initial texture map $T \in \mathbb{R}^{H\times W\times 3}$ to the ground truth $\tilde{T}$ would be to optimize the pixel value of T to minimize the error, *e.g.*, L1 loss, as : $T^* = \arg\min_T |T - \tilde{T}|$. 2) *Neural Re-parameterized Optimization*: Our method to fit a texture T is to re-parameterize it with the neural parameters, *i.e.*, $T = \mathcal{T}_\theta(\mathbf{z})$, where $\mathcal{T}_\theta(\cdot)$ is a randomly initialized convolutional U-Net with skip connections, and $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \in \mathbb{R}^{H\times W\times 3}$, which is fixed during the optimization. Thus, the optimization problem is as follows: $\theta^* = \arg\min_\theta |\mathcal{T}_\theta(\mathbf{z}) - \tilde{T}|$.

Frequency Band Energy Analysis. By comparing both methods, we see the characteristic differences of two representations: pixel parameters and deep convolutional re-parameterization. In this analysis, we investigate the energies of the frequency components in the texture maps. Given each iteration's texture map, we conduct the spatial texture frequency (Fourier) analysis and compute the energy components in five non-overlapping frequency bands from the lowest to highest frequencies. See supplementary for details.

Figures 4a and 4b show the energy-iteration plot of both methods. While the pixel value optimization fits all the frequency bands simultaneously (Fig. 4a), the neural re-parameterized optimization fits the low-frequency components faster and defers to fit the high-frequency components later (Fig. 4b), *i.e.*, schedules the frequency. Considering that lower-frequency bands mostly contain the content of the image while highest-frequency bands mainly correlate with noises in the image, we hypothesize that the scheduled frequency of neural re-parameterization helps the optimization focus more on the content of the texture map in the earlier iterations. The texture map visualizations show the neural re-parameterized optimization fits the overall texture and skin tones, *i.e.*, low-frequency, first in earlier iterations and details later. A similar observation in the natural image domain is reported in [49, 54], and we further show that the consistent result also holds for the PBR representation. On the other hand, the pixel optimization learns low-to-high frequencies simultaneously, which fits noise and texture signals jointly. This yields undesirable optimization paths that may be harmful for sensitive losses like the SDS loss, which is further investigated as follows.

4.2. Analysis of Optimization with the SDS Loss

We investigate whether the observed frequency scheduling effect of our neural re-parameterization occurs in more complicated *Paint-it* optimization with the SDS loss (Eq. (3)). Note that the SDS loss is much noisier than the L1 loss in Sec. 4.1. The randomness in the sampled perturbation noise ϵ, diffusion timestep t, and multi-view camera positions yield incoherent gradients in every optimization iteration.

Similar to the pixel optimization in Sec. 4.1, we design the baseline pixel optimization for synthesizing PBR texture maps with the SDS loss as follows:

$$[\mathbf{K}^{\mathrm{d}*}, \mathbf{K}^{\mathrm{rm}*}, \mathbf{K}^{\mathrm{n}*}] = \arg\min_{\mathbf{K}^{\mathrm{d}}, \mathbf{K}^{\mathrm{rm}}, \mathbf{K}^{\mathrm{n}}}$$
$$\mathbb{E}_{t,\epsilon}\left[\|\hat{\epsilon}_\phi(\mathcal{R}^{\mathbf{M}}_t(\mathbf{K}^{\mathrm{d}}, \mathbf{K}^{\mathrm{rm}}, \mathbf{K}^{\mathrm{n}}); y, t) - \epsilon\|^2_2\right] + \mathcal{L}_{\mathrm{TV}}, \quad (4)$$

where $\mathbf{K}^{\mathrm{d}}, \mathbf{K}^{\mathrm{rm}}, \mathbf{K}^{\mathrm{n}}$ denote the diffuse, roughness & metalness, and normal maps, respectively. We also use the total

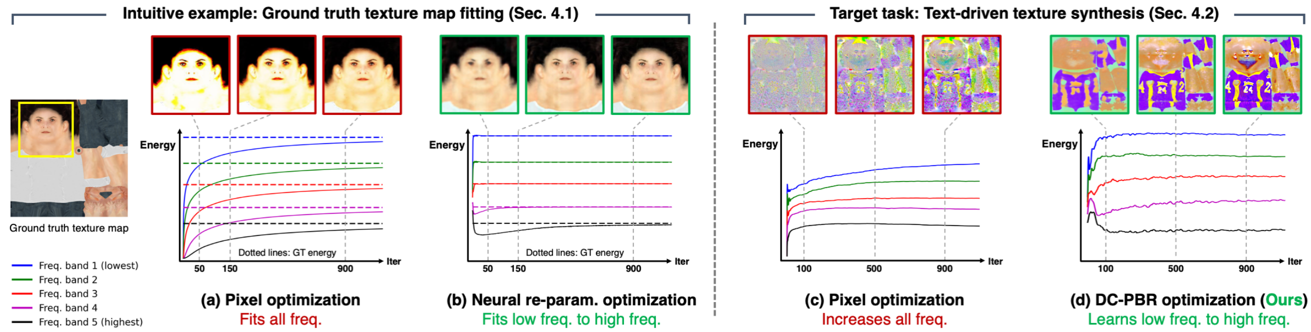

Figure 4. **Frequency scheduling of neural re-parameterized texture optimization**. For each iteration, we investigate the energies of the frequency components of the reconstructed (a,b) / synthesized (c,d) texture maps. The pixel optimization (a,c) fits and increases all frequency bands and suffers from fitting high-frequency texture contents from the initial stages, yielding degraded quality texture maps. In contrast, our proposed neural re-parameterization (b,d) naturally schedules which frequencies to focus on, thus obtaining coarse-to-fine texture synthesis with robustness to noisy supervision, *e.g.*, SDS loss, and yielding high-quality texture maps.

variation $\mathcal{L}_{\text{TV}}$ for $\mathbf{K}^{\text{d}}$ as a regularization to guide the smoothness of the local diffuse texture. This compensates for the lack of inductive bias in the pixel parameterization so that we can derive a stronger baseline to be compared.

Figures 4c and 4d compare the baseline pixel optimization (Eq. (4)) and our proposed DC-PBR optimization (Eq. (3)) by plotting the frequency band energies of the PBR texture maps obtained in each iteration. In Fig. 4c, the baseline pixel optimization increases all frequency bands. It fits noisy details from the SDS loss and yields significantly degraded texture maps. On the contrary, in Fig. 4d, our proposed neural re-parameterized optimization shows behaviors similar to those of Fig. 4b. The neural re-parameterization of DC-PBR guides the optimization to learn low-frequency bands faster than high-frequency noise, and later, mid-frequency bands follow. As a result, interestingly, the texture maps are spontaneously synthesized in a coarse-to-fine manner perceptually, where the overall structure and colors are learned first and the details, such as eyes and letters on the body, later.

Our neural re-parameterized optimization robustly filters out the high-frequency noise gradients from the SDS loss by its favorable frequency scheduling property. We postulate that this favorable property is induced by the architecture of the convolutional U-Net $\mathcal{T}_\theta$, consisting of a diverse composition of convolution kernels. The convolution kernel itself tends to learn favorable expressive local texture prior [19, 22, 54], including smoothness. Also, the stacked convolution mechanism that is repeatedly applied across the spatial domain with diverse compositions is analogous to other prior structures leveraging pattern recurrences of natural images, *e.g.*, [4, 15, 36, 56].

5. Experiments

5.1. Qualitative Results

In Fig. 5, we visualize the rendered meshes using *Paint-it*'s synthesized PBR texture maps for a given text prompt. To show the generalizability of the *Paint-it* synthesis method, we take the subsets of the large-scale mesh datasets: Objaverse [16] and RenderPeople [3] for general objects and clothed humans. For animals, we obtained the template meshes from the quadruped animal linear mesh model [65]. *Paint-it* can generate photorealistic and vivid textures with material properties such as a mushroom's matte surface and a teapot's metallic surface. By leveraging the strong generative prior from the pre-trained text-to-image diffusion model, *Paint-it* faithfully distinguishes texture parts for skin and cloth. Interestingly, *Paint-it* can generate pseudo-stereoscopic effects, even though the given mesh surface is flat, *e.g.*, the jewels and gems on a crown. We postulate this effect stems from our DC-PBR, where we synthesize the disentangled material properties along with perturbing tangent space normals. *Paint-it* also supports the material control or texture transfer for the same input mesh and the relighting using different environment maps, thanks to the synthesized PBR texture maps (Fig. 2).

5.2. Comparison with Competing Methods

In Fig. 6, we evaluate *Paint-it* with recent text-driven mesh texture synthesis methods, Latent-Paint [35], Fantasia3D [10], Text2Tex [9], and TEXTure [45]. Texture maps are generated from each method on identical 3D meshes and text. *Paint-it* synthesizes more vivid, realistic, and consistent textures, compared to texture inpainting methods, Text2Tex and TEXTure. Specifically, they suffer from multi-view texture inconsistencies and the baked lighting effects. The back-projection of the generated RGB image onto the mesh and the limited diffuse texture representation could be the reason. Latent-Paint synthesizes blurry texture and is also limited in diffuse texture. Fantasia3D learns a coordinate-based MLP to predict the per-point PBR materials, whereas we parameterize the full texture map globally. When backpropagating gradients, *Paint-it* has a global effect over the full texture, while Fantasia3D is much more local. Given that SDS is an

Figure 5. **Qualitative results**. We take diverse 3D meshes from Objaverse [16], RenderPeople [3], and SMAL [65], then synthesize texture maps with our manual text prompts. We visualize the original and rendered meshes with our synthesized PBR texture maps. *Paint-it* can model diverse material properties, *e.g.*, the metallic surface of a crown, the rough surface of a mushroom, realistic human skin tones, front-to-back appearance consistency, and complicated patterns of the animal's appearance. See supplementary material for more results.

Figure 6. **Qualitative comparison**. We compare *Paint-it* with recent competing methods [9, 10, 35, 45]. We script each method to synthesize textures for the subset of Objaverse [16] meshes and compare the rendered quality of the textured meshes. Deep convolutional re-parameterization of the PBR texture maps helps *Paint-it* synthesize a photorealistic and vivid appearance compared to other methods.

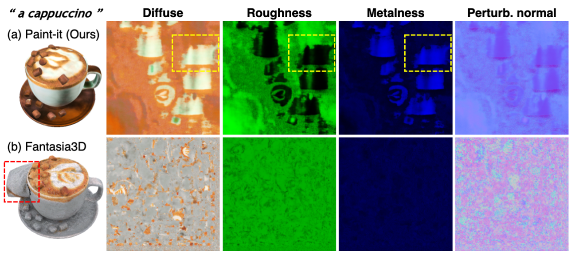

Figure 7. **PBR disentanglement results.** *Paint-it* vs. Fantasia3D.

	Latent-Paint	Fantasia3D	Text2Tex	TEXTure	Ours
FID (↓)	41.11	58.79	37.89	38.40	**34.46**
User score (↑)	3.22	2.71	3.34	3.04	**4.37**

Table 1. **Quantitative results on Objaverse subset.** We evaluate the realism of the synthesized texture maps by measuring FID and user study. *Paint-it* outperforms the recent competing methods.

ambiguous and noisy signal, the global gradient update of *Paint-it* helps get a higher-quality, coherent appearance. In Fig. 7, *Paint-it* obtain better-disentangled materials and specular properties, while Fantasia3D fails to generate the mug's metallic (smooth) surface. Also, Fantasia3D re-meshes the input mesh, destroying the geometry and obtaining implausible *uv* mapping with substantial seams.

Following the protocol from Text2Tex [9], we report the Fréchet Inception Distance (FID) [23]. Given untextured meshes from Objaverse [16], we scripted each method to synthesize texture maps from the same text prompt. Then, we render the textured meshes in multi-views and compute the FID score. Please refer to Text2Tex for details. We also conduct a user study, requesting users to rate the realism of samples synthesized with *Paint-it* and competing methods. We got responses from 30 users. Table 1 shows that Paint-it outperforms recent competing methods in terms of FID and user scores. Only *Paint-it* surpasses the score four (realistic), showing our superior realism and synthesis quality.

5.3. Ablation studies

Effects of PBR Texture Representation. First, we optimize only the diffuse texture map $\mathbf{K}^d$ as other recent methods [9, 45]. Simplifying the texture representation to model only the diffuse texture still generates a decent visual quality. However, compared to our full method, it is less realistic since it cannot model the reflection on the surface or stereoscopic effects. We highlight the notable difference in the visual qualities of our full method and the diffuse-only optimization in Fig. 8a, *w/o* PBR and in Fig. 8b.

Effects of Texture Neural Re-parameterization. As discussed in Sec. 4, DC-PBR, *i.e.*, neural re-parameterized optimization, naturally embodies the frequency scheduling for synthesizing textures. While baseline optimization (Eq. (4)) adopts a regularization term to avoid synthesizing noisy textures with high frequencies, it still introduces severely jittered textures (see Fig. 8a, *w/o* Re-param., & Fig. 8c).

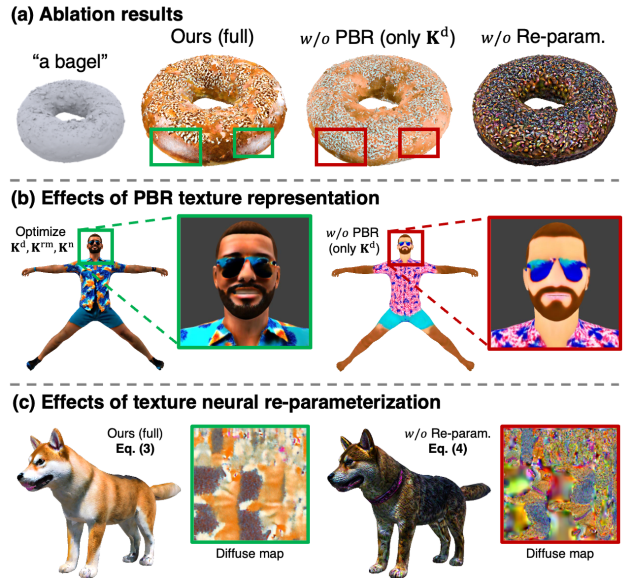

Figure 8. **Ablation study.** The proposed neural re-parameterization of PBR textures significantly enhances the visual qualities of the meshes, *e.g.*, stereoscopic effects, realism, and texture consistency.

6. Discussion, Limitation, and Conclusion

We present *Paint-it*, a text-based synthesis of physically-based rendering (PBR) texture maps for meshes. We propose the deep convolutional re-parameterization of PBR texture maps, which inherently eases and robustifies the optimization with the Score-Distillation Sampling. We show the performance and potential of the proposed method by synthesizing high-fidelity PBR texture maps for large-scale mesh datasets, including general objects, humans, and animals.

We expect *Paint-it* can revolutionize the heuristic graphics pipelines, *e.g.*, editing, relighting textures, and generating unlimited realistic 3D assets for production. The current limitation of *Paint-it* is the optimization time, which takes approximately 15∼30 minutes per mesh. To further accelerate *Paint-it*, an efficient loss using the Consistency models [51] would be helpful. Also, based on our synthesized texture maps for large-scale mesh datasets, curating a PBR texture map dataset and using it to train a feed-forward generative model would be a promising future direction.

Acknowledgment. We thank the members of AMILab [1] and RVH group [2] for their helpful discussions and proofreading. The project was made possible by funding from the Carl Zeiss Foundation. This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 409792180 (Emmy Noether Programme, project: Real Virtual Humans), and the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A. Gerard Pons-Moll is a Professor at the University of Tübingen endowed by the Carl Zeiss Foundation, at the Department of Computer Science and a member of the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645. Kim Youwang and Tae-Hyun Oh were supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2023-00225630, Development of Artificial Intelligence for Text-based 3D Movie Generation; No.2022-0-00290, Visual Intelligence for Space-Time Understanding and Generation based on Multi-layered Visual Common Sense; No.2021-0-02068, Artificial Intelligence Innovation Hub).

References

[1] https://ami.postech.ac.kr/members. 8

[2] http://virtualhumans.mpi-inf.mpg.de/people.html. 8

[3] https://renderpeople.com/, 2023. 6, 7, 15, 17, 19

[4] Connelly Barnes, Eli Shechtman, Dan B Goldman, and Adam Finkelstein. The generalized PatchMatch correspondence algorithm. In *European Conference on Computer Vision (ECCV)*, 2010. 6

[5] Benjamin Biggs, Oliver Boyne, James Charles, Andrew Fitzgibbon, and Roberto Cipolla. Who left the dogs out?: 3D animal reconstruction with expectation maximization in the loop. In *European Conference on Computer Vision (ECCV)*, 2020. 15

[6] Tianshi Cao, Karsten Kreis, Sanja Fidler, Nicholas Sharp, and KangXue Yin. Texfusion: Synthesizing 3d textures with text-guided image diffusion models. In *IEEE International Conference on Computer Vision (ICCV)*, 2023. 2, 3

[7] Yukang Cao, Yan-Pei Cao, Kai Han, Ying Shan, and Kwan-Yee K. Wong. Dreamavatar: Text-and-shape guided 3d human avatar generation via diffusion models. *arXiv preprint, arxiv:2304.00916*, 2023. 3

[8] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[9] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. In *IEEE International Conference on Computer Vision (ICCV)*, 2023. 2, 3, 6, 7, 8, 13, 15

[10] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *IEEE International Conference on Computer Vision (ICCV)*, 2023. 3, 6, 7, 12, 13, 15

[11] Xu Chen, Tianjian Jiang, Jie Song, Jinlong Yang, Michael Black, Andreas Geiger, and Otmar Hilliges. gdna: Towards generative detailed neural avatars. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[12] Yongwei Chen, Rui Chen, Jiabao Lei, Yabin Zhang, and Kui Jia. Tango: Text-driven photorealistic and robust 3d stylization via lighting decomposition. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3

[13] Junhyeong Cho, Kim Youwang, and Tae-Hyun Oh. Cross-attention of disentangled modalities for 3d human mesh recovery with transformers. In *European Conference on Computer Vision (ECCV)*, 2022. 15

[14] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics (SIGGRAPH)*, 1(1), 1982. 4

[15] Antonio Criminisi, Patrick Perez, and Kentaro Toyama. Object removal by exemplar-based inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003. 6

[16] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 6, 7, 8, 13, 15, 16, 18

[17] Zijian Dong, Xu Chen, Michael J.Black Jinlong Yang, Otmar Hilliges, and Andreas Geiger. AG3D: Learning to generate 3D avatars from 2D image collections. In *IEEE International Conference on Computer Vision (ICCV)*, 2023. 1

[18] Yao Feng, Haiwen Feng, Michael J. Black, and Timo Bolkart. Learning an animatable detailed 3D face model from in-the-wild images. *ACM Transactions on Graphics (SIGGRAPH)*, 40(8), 2021. 15

[19] Leon Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 6

[20] Shubham Goel, Georgios Pavlakos, Jathushan Rajasegaran, Angjoo Kanazawa*, and Jitendra Malik*. Humans in 4D: Reconstructing and tracking humans with transformers. In *IEEE International Conference on Computer Vision (ICCV)*, 2023. 15

[21] Vladimir Guzov, Aymen Mir, Torsten Sattler, and Gerard Pons-Moll. Human poseitioning system (hps): 3d human pose estimation and self-localization in large scenes from body-mounted sensors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 15

[22] Reinhard Heckel and Mahdi Soltanolkotabi. Denoising and regularization via exploiting the structural bias of convolutional generators. In *International Conference on Learning Representations (ICLR)*, 2020. 6

[23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 8

[24] Fangzhou Hong, Mingyuan Zhang, Liang Pan, Zhongang Cai, Lei Yang, and Ziwei Liu. Avatarclip: Zero-shot text-driven generation and animation of 3d avatars. *ACM Transactions on Graphics (SIGGRAPH)*, 41(4):1–19, 2022. 3

[25] Yukun Huang, Jianan Wang, Yukai Shi, Xianbiao Qi, Zheng-Jun Zha, and Lei Zhang. Dreamtime: An improved optimization strategy for text-to-3d content creation. *arXiv preprint, arxiv:2306.12422*, 2023.

[26] Yukun Huang, Jianan Wang, Ailing Zeng, He Cao, Xianbiao Qi, Yukai Shi, Zheng-Jun Zha, and Lei Zhang. Dreamwaltz: Make a scene with complex 3d animatable avatars. *arXiv preprint, arxiv:2305.12529*, 2023. 1, 3

[27] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[28] Ruixiang Jiang, Can Wang, Jingbo Zhang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Avatarcraft: Transforming text into neural human avatars with parameterized shape and pose control. In *IEEE International Conference on Computer Vision (ICCV)*, 2023. 1, 3

[29] GeonU Kim, Kim Youwang, and Tae-Hyun Oh. FPRF: Feed-forward photorealistic style transfer of large-scale 3D neural radiance fields. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2024. 14

[30] Jihoon Kim, Jiseob Kim, and Sungjoon Choi. Flame: Free-form language-based motion synthesis & editing. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022. 15

[31] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (SIGGRAPH)*, 39(6), 2020. 5

[32] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 14

[33] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 3

[34] Qianli Ma, Jinlong Yang, Anurag Ranjan, Sergi Pujades, Gerard Pons-Moll, Siyu Tang, and Michael Black. Learning to dress 3d people in generative clothing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 1

[35] Gal Metzer, Elad Richardson, Or Patashnik, Raja Giryes, and Daniel Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3, 6, 7, 13, 15

[36] Tomer Michaeli and Michal Irani. Nonparametric blind super-resolution. In *IEEE International Conference on Computer Vision (ICCV)*, 2013. 6

[37] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 3

[38] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 3

[39] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (SIGGRAPH)*, 41(4):102:1–102:15, 2022. 14

[40] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[41] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 15

[42] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *International Conference on Learning Representations (ICLR)*, 2022. 1, 2, 3, 12

[43] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021. 3

[44] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J. Guibas. Humor: 3d human motion model for robust pose estimation. In *IEEE International Conference on Computer Vision (ICCV)*, 2021. 15

[45] Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. Texture: Text-guided texturing of 3d shapes. *ACM Transactions on Graphics (SIGGRAPH)*, 2023. 2, 3, 6, 7, 8, 13, 15

[46] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 3

[47] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, Seyedeh Sara Mahdavi, Raphael Gontijo Lopes, Tim Salimans, Jonathan Ho, David Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 1, 3

[48] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 15

[49] Zenglin Shi, Pascal Mettes, Subhransu Maji, and Cees G M Snoek. On measuring and controlling the spectral bias of the deep image prior. *International Journal of Computer Vision*, 2022. 5

[50] 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 (ECCV)*, 2022. 3

[51] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning (ICML)*, 2023. 8

[52] Kim Sung-Bin, Lee Hyun, Da Hye Hong, Suekyeong Nam, Janghoon Ju, and Tae-Hyun Oh. Laughtalk: Expressive 3d talking head generation with laughter, 2024. 15

[53] Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. Human motion diffusion model. In *International Conference on Learning Representations (ICLR)*, 2023. 15

[54] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5, 6, 12, 14

[55] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[56] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[57] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. 3

[58] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 1

[59] Xianghui Xie, Bharat Lal Bhatnagar, and Gerard Pons-Moll. Visibility aware human-object interaction tracking from single rgb camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 15

[60] Yuxuan Xue, Bharat Lal Bhatnagar, Riccardo Marin, Nikolaos Sarafianos, Yuanlu Xu, Gerard Pons-Moll, and Tony Tung. Nsf: Neural surface fields for human modeling from monocular depth. In *IEEE International Conference on Computer Vision (ICCV)*, 2023. 15

[61] Kim Youwang, Kim Ji-Yeon, Kyungdon Joo, and Tae-Hyun Oh. Unified 3d mesh recovery of humans and animals by learning animal exercise. In *British Machine Vision Conference (BMVC)*, 2021. 15

[62] Kim Youwang, Kim Ji-Yeon, and Tae-Hyun Oh. CLIP-Actor: Text-driven recommendation and stylization for animating human meshes. In *European Conference on Computer Vision (ECCV)*, 2022. 3, 15

[63] Kim Youwang, Lee Hyun, Kim Sung-Bin, Suekyeong Nam, Janghoon Ju, and Tae-Hyun Oh. A large-scale 3D face mesh video dataset via neural re-parameterized optimization. *arXiv preprint, arXiv:2310.03205*, 2023. 15

[64] Wojciech Zielonka, Timo Bolkart, and Justus Thies. Towards metrical reconstruction of human faces. In *European Conference on Computer Vision (ECCV)*, 2022. 15

[65] Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 7

Paint-it: Text-to-Texture Synthesis via Deep Convolutional Texture Map Optimization and Physically-Based Rendering

— Supplementary Material —

In this supplementary material, we provide additional details and results that are not included in the main paper due to the space limit. The attached video includes a brief introduction and interesting qualitative results of *Paint-it*.

A. Details of *Paint-it*

A.1. DC-PBR: Network Design

The main contribution of our work is the proposed DC-PBR parameterization for optimizing the physically-based rendering (PBR) texture maps. Instead of pixel-based parameterization of the PBR texture maps, we introduce the fixed random noise input $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I}) \in \mathbb{R}^{H\times W\times 3}$, and a randomly initialized U-Net with skip connections, $\mathcal{T}_\theta$. We obtain the PBR texture map $[\mathbf{K}^{\text{d}}_\theta, \mathbf{K}^{\text{rm}}_\theta, \mathbf{K}^{\text{n}}_\theta] = \mathcal{T}_\theta(\mathbf{z}) \in \mathbb{R}^{H\times W\times(3+2+3)}$, for every iteration of the synthesis optimization.

Our design choice of DC-PBR is inspired by the Deep Image Prior [54], and we extended it to re-parameterize the PBR texture maps for the text-driven texture map synthesis task. We use an encoder-decoder ("hourglass") architecture with skip connections between encoder and decoder features for our neural re-parameterization, DC-PBR $\mathcal{T}_\theta$. For the network hyperparameters, we used the *default architecture* of the Deep Image Prior, *i.e.*, five levels of downsampling and upsampling layers for the encoder and decoder. We encourage readers to refer to the details in Fig. 21 of Deep Image Prior. We empirically set the learning rate as $5 \cdot 10^{-4}$ and the total iteration for PBR texture synthesis as 1000.

A.2. Details of SDS Loss

Recall that we optimize the DC-PBR given the text with the Score-Distillation Sampling (SDS).

SDS Loss for Multi-view Mesh Images. We adopt some engineering to obtain high-fidelity and multi-view consistent PBR texture maps. When computing the SDS loss, we need the rendered image of the textured mesh. We randomly sample camera poses in multi-view and render N view images. We sample the elevation angle as $\varphi_{\text{elev}} \sim \mathcal{U}(-\frac{\pi}{3}, \frac{\pi}{3})$, and the azimuth angle as $\varphi_{\text{azim}} \sim \mathcal{U}(0, 2\pi)$. We set $N = 4$ for most cases. When optimizing DC-PBR for humans and animals, we increase the generation quality of the face regions by additionally rendering the face-focused images. We translate the mesh so that the head can be the center of the world coordinate and render it with $\varphi_{\text{elev}} \sim \mathcal{U}(-\frac{\pi}{6}, \frac{\pi}{3})$ and $\varphi_{\text{azim}} \sim \mathcal{U}(0, 2\pi)$. For human and animal cases, we use a total $N = 8$ views for computing SDS loss, where four views are for the full body, and the others are for face regions. We also use the directional text prompt engineering as in prior arts [10, 42] to mitigate the "Janus problem".

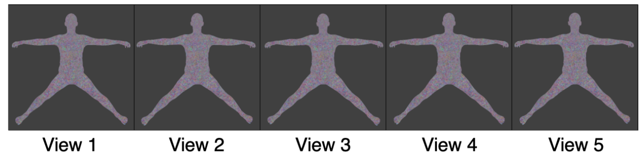

Figure S1. (For SDS analysis) Textured mesh image rendered from adjacent camera views $-2^\circ < \varphi_{\text{elev}} < 2^\circ$ and $-2^\circ < \varphi_{\text{azim}} < 2^\circ$. The images are slightly different but look almost identical.

When computing the SDS loss, at each iteration, we synchronize the noise ϵ and the noising timestep t for multi-view rendered images, *i.e.*, we randomly sample a single ϵ and t for each synthesis iteration and add the same amount of noise to the multi-view mesh images. Finally, we sample the noising timestep t, from the distribution $\mathcal{U}(t_{\text{min}}, t_{\text{max}})$. We start by $[t_{\text{min}}, t_{\text{max}}] = [0.2, 0.98]$, and linearly narrows down the distribution so that it become $[t_{\text{min}}, t_{\text{max}}] = [0.3, 0.5]$ by the end. We empirically set the ranges for t_{min} and t_{max}.

Why is SDS Loss a Noisy Signal?. In the main paper, we denote that the SDS loss is a noisy signal. By noisy, we refer to the incoherent nature of the SDS loss. From Sec. 3.1 and Eq. 1 in the main paper, we notice the SDS loss is dependent on the randomly sampled Gaussian noise $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ and the noising timestep $t \sim \mathcal{U}(t_{\text{min}}, t_{\text{max}})$. We sample ϵ and t for every iteration of the PBR texture map synthesis; thus, the SDS loss is highly likely to give incoherent direction for updating the DC-PBR $\mathcal{T}_\theta$. Moreover, as aforementioned, we use multi-view rendered images. Multi-view images contain different visible mesh parts, potentially providing incoherent update directions for $\mathcal{T}_\theta$.

To show the incoherent SDS gradient, we design a toy example. Given a mesh and a text input, we render the mesh in N adjacent views, *i.e.*, we sample elevation and azimuth in the range $-2^\circ < \varphi_{\text{elev}} < 2^\circ$ and $-2^\circ < \varphi_{\text{azim}} < 2^\circ$. Since the camera views are closely distributed, the rendered mesh images would look almost identical (see Fig. S1).

We investigated the backward gradients $\nabla\mathcal{L}_{\text{SDS}}$ applied on the diffuse map $\mathbf{K}^{\text{d}}_\theta$, computed from each view, with the identical text prompt and ϵ and t. We obtain a flattened, stacked gradient matrix from per-view SDS loss gradients on the diffuse map. Formally, we obtain the gradient matrix $\mathbf{G} \in \mathbb{R}^{N\times F}$, where N denotes the number of rendered views, and F denotes the flattened dimension of the gradient. Even though we compute the SDS loss with the same text prompt, same ϵ and t, and the almost identical rendered images indistinguishable in eyes, we observe that the gradient matrix $\mathbf{G}$ is in high rank. We first obtain the singular values of the gradient matrix $\mathbf{G}$ using the Singular Value Decomposition (SVD) and investigate the ratio of all the singular values with the smallest singular value. From a toy example, where $N = 5$, we obtain the singular value ratio as $[2.1868, 1.7937, 1.7838, 1.6689, 1.0000]$, *i.e*., the highest and lowest singular values do not deviate too much in terms of the scale, showing $\mathbf{G}$ is high rank. In other words, the SDS loss, computed with a pre-trained text-to-image diffusion model, provides incoherent guidance from multiple views and guides the optimization in incoherent directions.

We claim the SDS loss is a noisy signal from this observation. In Sec. 4.2, we empirically show our DC-PBR filters out the high-frequency noisy signal by inherently scheduling the optimization curriculum. Thus, DC-PBR is effective when combined with the noisy SDS loss.

A.3. Details of Frequency-based Analysis

We plot the energy-iteration plot in our frequency-based analysis of the proposed DC-PBR (Fig. 4 in the main paper). We first performed the Fast Fourier Transform (FFT) of the diffuse texture map obtained in each iteration. See Fig. S2a for the FFT result. Then, we define five non-overlapping frequency bands, depending on the radius from the center of the FFT image, as in Fig. S2b. Note that the ranges of the frequency bands are fixed during the optimization. Finally, we compute the energy of each frequency band by summing all the frequency response magnitudes (either absolute value or square works) in each frequency band.

In Fig. 4-left, the optimization starts from a monotonic gray image, which has high zero-frequency energies, *i.e*., DC component. However, note that we visualize each frequency band's 'average' energy. DC component occupies only a tiny area ($\ll 1\%$) in the lowest frequency band; thus, the pixel brightness hardly affects the band's energy.

During *Paint-it* optimization, our DC-PBR representation automatically schedules the curriculum to learn low frequency first, then mid frequency, and filters out the highest frequency, *i.e*., noise.

A.4. Details of User Study

We conducted a user study to assess the realism of the synthesized PBR texture maps. We showed ten untextured meshes from the Objaverse [16] dataset and showed five different results obtained from the methods: Latent-Paint [35], Fantasia3D [10], Text2Tex [9], TEXTure [45], and *Paint-it* (ours). We asked 30 users with engineering/non-engineering backgrounds to rate the realism of the rendered images in the score range 1 to 5, *i.e*., 1: very unrealistic, 2: unrealistic, 3: neutral, 4: realistic, and 5: very realistic. The order of the methods was randomly shuffled for fairness. The interface

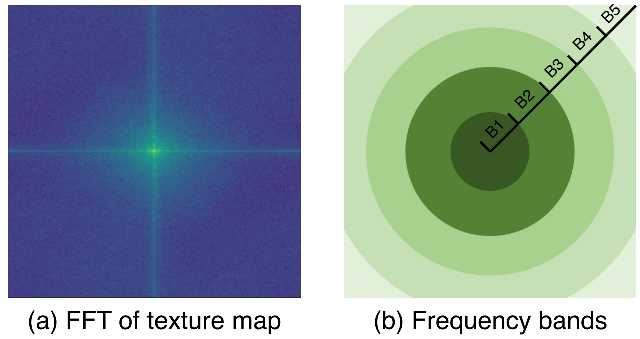

Figure S2. (a) Visualization of the FFT image of the diffuse map. (b) Visualization of the pre-defined non-overlapping frequency bands. Center: low frequency, Outer: high frequency.

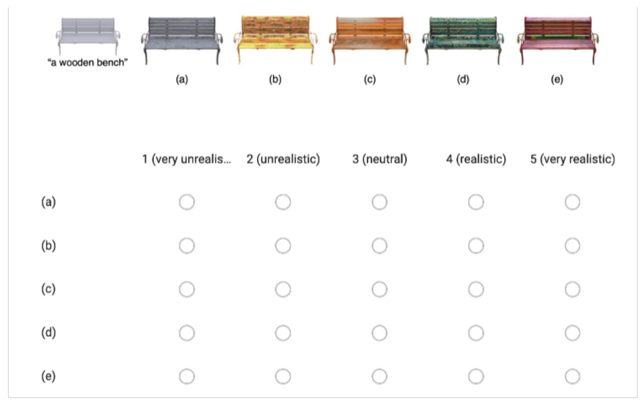

Figure S3. The users were asked to rate the realism of the rendered mesh images, textured with five different methods. The order of the methods was randomly shuffled for each question.

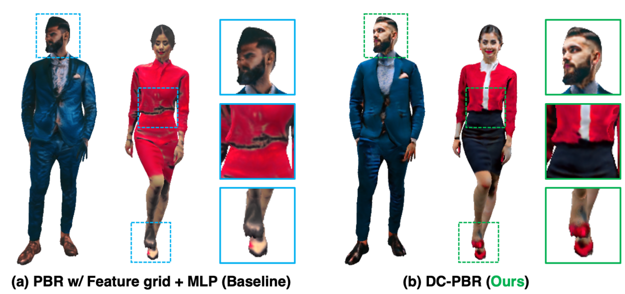

Figure S4. Feature grid+MLP (baseline) vs. DC-PBR (ours)

of the user study is shown in Fig. S3.

A.5. Details & Discussion about Optimization Time

The optimization time for synthesizing PBR texture maps takes about 15 min. for general object meshes in Objaverse [16]. For more complex cases, such as 3D humans or animals, we additionally use face-focused mesh renderings; thus, it takes about 30 min. to complete. We optimized both the baseline (Eq. (4) in the main paper, pixel optimization) and our DC-PBR until convergence, and there's no

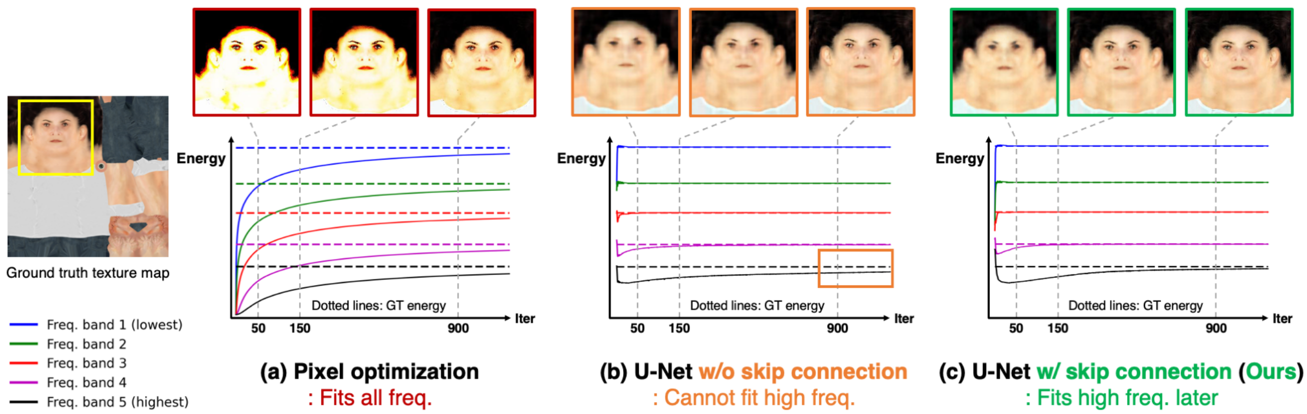

Figure S5. **Effect of skip connection: Texture map fitting.** When fitting a texture map with different parameterizations, U-Net without skip connections fails to fit the highest frequency band. This hints that the skip connections are responsible for representing fine-grained details.

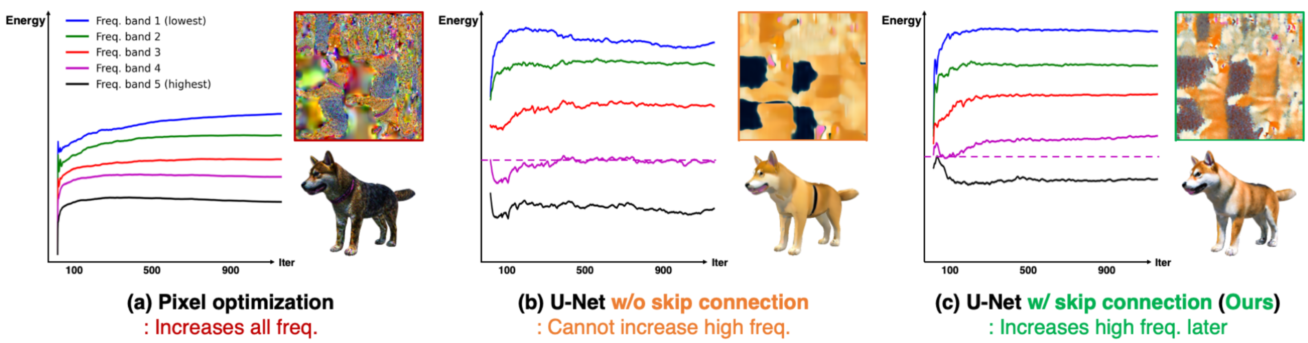

Figure S6. **Effect of skip connection: Text-driven texture synthesis.** When synthesizing PBR texture maps with different parameterizations, U-Net without skip connections cannot increase the mid-to-high frequency band (purple). Proposed DC-PBR with U-Net and skip connections can synthesize fine-grained details in texture maps, resulting in high-fidelity synthesis results.

significant time difference. Instead, DC-PBR helps the optimization with noisy SDS loss find a better solution than the vanilla pixel optimization. We use a single NVIDIA RTX A6000 GPU for the optimization.

Extending *Paint-it* to large-scale 3D scenes would be an interesting future direction. However, *Paint-it*'s PBR texture generation for large-scale scenes would take a longer optimization time. As suggested in FPRF [29], we may try semantic feature distillation to accelerate the optimization when stylizing large-scale 3D scenes using SDS.

B. Additional Experiment

B.1. More Baseline for PBR Representation

For a PBR representation baseline, other than direct pixel optimization (Eq. (4) in the main paper), we implement a multi-resolution hash encoding of grid features and subsequent MLP [39] to model the disentangled PBR texture maps. In Fig. S4, our DC-PBR representation yields smoother and more vivid texture results than the new baseline. In Table S1, our DC-PBR obtains a better FID score than other PBR representation baselines (Pixel optim., & Feat. grid+MLP). We postulate that the SDS gradients only propagate to local hash grids in the new baseline, lacking non-local texture smoothness, as supported in [32]. On the other hand, the spatial CNN kernels of our DC-PBR are beneficial in naturally imposing texture smoothness.

B.2. Effects of DC-PBR Design Choice

In this section, we investigate the effect of our DC-PBR design choice. Specifically, we study the effect of our U-Net with skip connections in synthesizing the PBR texture maps.

As discussed in the main paper and Sec. A.1, we use a *randomly initialized* U-Net with skip connections, shortly, U-Net+skip. Deep Image Prior [54] used U-Net+skip and claimed skip connections inherently promote self-similarity across multi-scales, which is beneficial for inverse problems. We wanted to investigate how the skip connections affect the DC-PBR optimization with SDS loss. Thus, we conduct the same experiment as in Sec. 4 of the main paper, but with U-Net, without (*w/o*) skip connections.

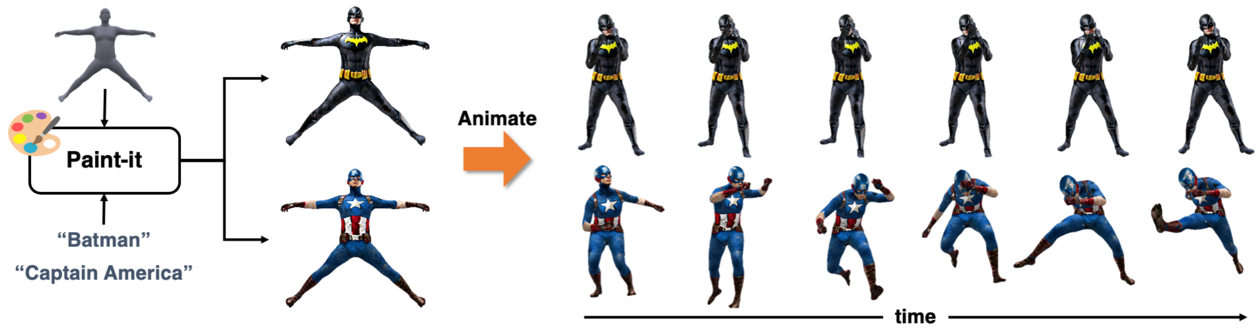

Figure S7. ***Paint-it*: Dynamic virtual 3D humans.** We synthesize PBR texture maps given text and rigged mesh, *e.g.*, SMPL-X [41], using *Paint-it*. Then, we animate the textured 3D humans using sequential pose parameters (can be retrieved [62] or generated [30]).

	Pixel optim.	Only K^d	Feat. grid+MLP	**DC-PBR (Ours)**
FID (↓)	216.6	59.39	55.98	**34.46**

Table S1. FID for ablation study

Fitting behavior. Following the experiment in Sec. 4.1, we fit a randomly initialized U-Net *w/o* skip connections given a ground-truth texture map. In Fig. S5b, the energy-iteration plot shows that parameterizing a texture map with U-Net *w/o* skip connection fails to fit high frequency.

DC-PBR synthesis behavior. Similarly, for our task, *i.e.*, text-driven DC-PBR optimization, U-Net *w/o* skip connection fails to increase the mid-to-high frequency band (purple line), resulting in blurry texture maps. In contrast, our DC-PBR, parameterized in U-Net+skip, successfully increases the mid-to-high frequency band, synthesizing fine-grained texture maps. We conclude that the skip connections are in charge of synthesizing fine-grained, high-frequency details of texture maps. This observation aligns with the Deep Image Prior's claim, where skip connections benefit the inverse problems with multi-scale feature awareness. We additionally showed the frequency level behavior of the skip connection through the experiments (see Fig. S6).

C. Additional Results

C.1. More Quantitative Comparison

In Tab. S1, we report FID scores for the ablation study (in Sec. 5.3 in the main paper). We used 410 meshes from Objaverse to get the real sample set and added 50 meshes for the generated sample set. The results support that our full DC-PBR enhances the realism of the generated texture.

C.2. More Qualitative Results

We provide more qualitative results of our *Paint-it*. Given untextured meshes from Objaverse [16] and RenderPeople [3], we obtain the text prompts from 1) manually writing the requirements, *e.g.*, a Spiderman lego minifigure, or 2) generating an automatic text caption using multi-modal large-language models, *e.g.*, GPT-4. Then, we conduct *Paint-it* optimization to synthesize PBR texture maps and render the textured meshes (see Fig. S8 and Fig. S9).

C.3. More Comparison Results

In Figs. S10 and S11, we provide more qualitative results that compare *Paint-it* and recent competing methods [9, 10, 35, 45]. As in Fig. 6 of the main paper, we synthesize texture maps using each method for the same untextured meshes and text prompts. Overall, *Paint-it* synthesizes much realistic and vivid texture on the meshes, thanks to the PBR texture representation and texture smoothness induced by our proposed DC-PBR. Note that Fantasia3D [10] also synthesizes PBR materials, but in a per-point independent manner; thus, it lacks texture smoothness and yields substantial jitterings. Moreover, given an untextured mesh, Fantasia3D converts the mesh into a signed distance function (SDF) representation, DMTet [48]. Such auxiliary re-meshing optimization leads to severe geometric quality degradation, *e.g.*, floating artifacts on a toy bicycle example, in Fig. S10.

C.4. *Paint-it* for Animated Meshes

Paint-it can also synthesize high-quality PBR texture maps for animatable meshes and generate dynamic 3D assets. Since *Paint-it* does not perform a re-meshing process and preserves the original UV texture coordinates, we can synthesize maps for any rigged meshes, *e.g.*, T-posed human mesh, and animate with any motion sequences.

In this paper, we used SMPL-X [41]. We first take the canonical posed SMPL-X mesh and synthesize PBR texture maps using *Paint-it*. To animate the textured avatars, one may use the motion captured mesh sequences of 3D human bodies [13, 20, 21, 59, 60], faces [18, 63, 64] or even animals [5, 61]. Generative models for natural body or facial motions [44, 52, 53] could also be applied for animation. We may also use the posed meshes and perform advanced augmentations as proposed in CLIP-Actor [62]. We visualize the synthesized animated meshes in Fig. S7.

Figure S8. **Qualitative results of *Paint-it*: Objaverse dataset [16]**. Given any untextured mesh from the existing mesh database, *Paint-it* synthesizes high-fidelity, locally smooth, and realistic object PBR texture maps.

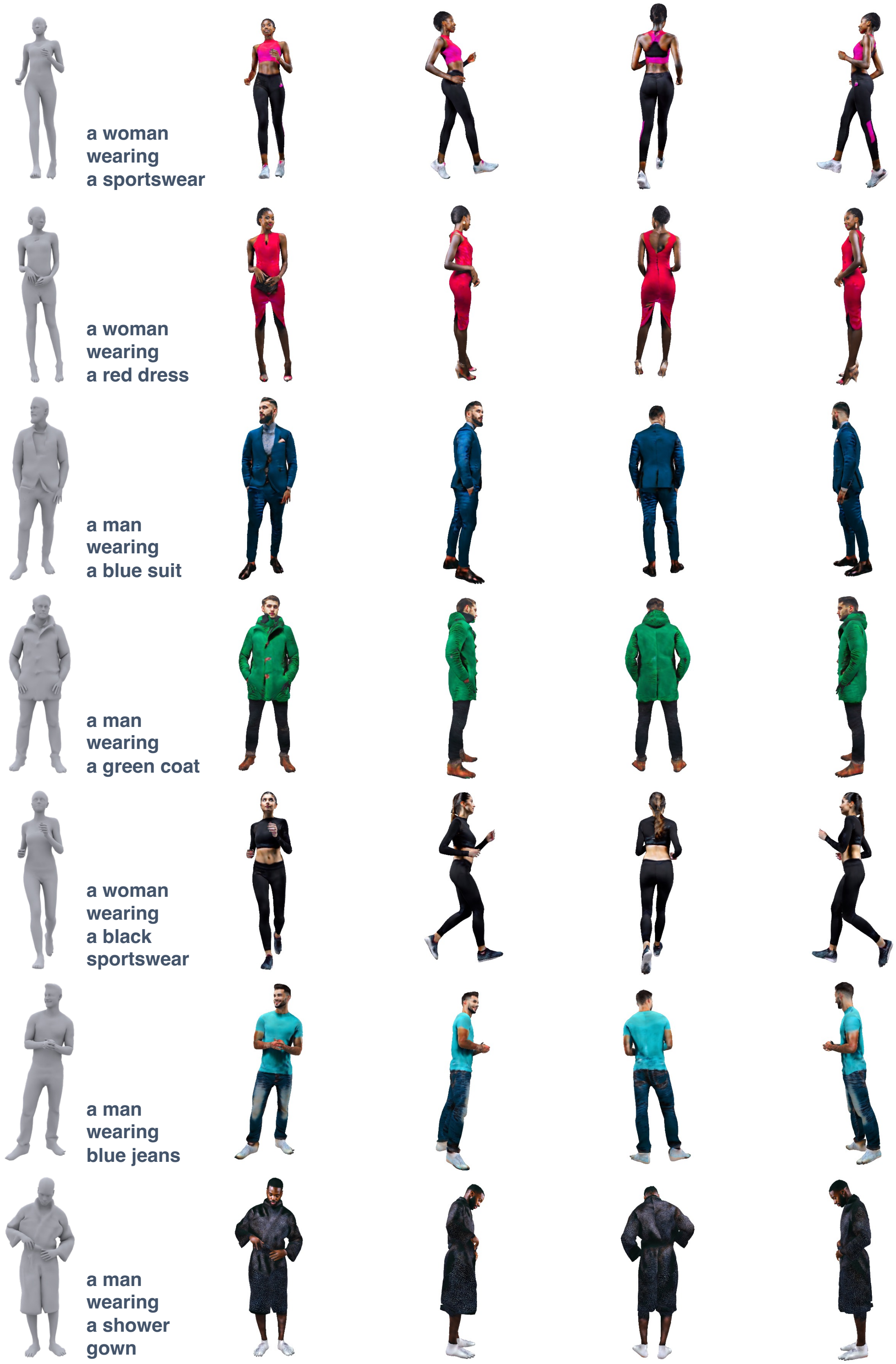

Figure S9. **Qualitative results of *Paint-it*: RenderPeople dataset [3].** Given untextured clothed human meshes, *Paint-it* synthesizes high-fidelity, vivid, and multi-view consistent human and cloth PBR textures. We render four different views of the textured mesh.

Figure S10. **Comparison results: Objaverse dataset [16].**

Figure S11. **Comparison results: RenderPeople dataset [3].**