Text-driven Photorealistic 3D Stylization For Arbitrary Meshes

Anonymous Author(s)
Affiliation
Address
email

Abstract

Creation of 3D content by stylization is a promising yet challenging problem in computer vision and graphics research. In this work, we focus on stylizing photorealistic appearance renderings of a given surface mesh of arbitrary topology. Motivated by the recent surge of cross-modal supervision of the Contrastive Language-Image Pre-training (CLIP) model, we propose to transfer the appearance style of a given 3D shape according to a text prompt in a photorealistic manner. Technically, we propose to disentangle the appearance style as the spatially varying bidirectional reflectance distribution function, the local geometric variation, and the lighting condition, which are jointly optimized, via supervision of the CLIP loss, by a spherical Gaussians based differentiable renderer. As such, our method enables photorealistic 3D style transfer by automatically predicting reflectance effects even for bare, low-quality meshes, without training on a task-specific dataset. Extensive experiments show that our method outperforms existing methods of text-driven 3D style transfer in terms of photorealistic quality, consistency of 3D geometry, and robustness when stylizing low-quality meshes.

1 Introduction

3D content creation by stylization (e.g., stylized according to text prompts [33], images [46], or 3D shapes [52]) has important applications in computer vision and graphics areas. The problem is yet challenging and traditionally requires manual efforts from experts of professional artists and a large amount of time cost. In the meanwhile, many online 3D repositories [6, 49, 43] can be easily accessed on the Internet whose contained surface meshes are bare contents without any styles. It is thus promising if automatic, diverse, and realistic stylization can be achieved given these raw 3D contents. We note that similar to recent 3D stylization works [33, 52, 20, 35, 23], we consider *style* as the particular appearance of an object, which is determined by color (albedo) and physical reflective effect of the object surface, while considering *content* as the global shape structure and topology prescribed by an explicit 3D mesh or other implicit representation of the object surface.

Stylization is usually guided by some sources of styling signals, such as a text prompt [33], a reference image [46], or a target 3D shape [52]. In this work, we choose to work with stylization by text prompts, motivated by the surprising effects recently achieved in many applications [46, 21, 10], by using the cross-modal supervision model of Contrastive Language–Image Pre-training (CLIP) [39]. The goal of this paper is to devise an end-to-end neural architecture system that can transfer, guided by a text prompt, the style of a given 3D shape of arbitrary topology. Note that our method can be directly applied to arbitrary meshes with arbitrary target styles, without additional learning/optimization procedure on a task-specific dataset; Figure 1 gives the illustration.

The most relevant work so far is Text2Mesh [33], which is the first to use the CLIP loss in mesh stylization; given an input mesh and a text prompt, it predicts stylized color and displacement for

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

each mesh vertex, and the stylized mesh is just a result of this colored vertex displacement procedure. However, Text2Mesh does not support stylization of lighting conditions, reflectance properties, and local geometric variations, which are necessary in order to produce a photorealistic appearance of 3D surface. To this end, we formulate the problem of mesh stylization as learning three unknowns of the surface, i.e., the spatially varying Bidirectional Reflectance Distribution Function (SVBRDF), the local geometric variation (normal map), and the lighting condition. We show that our method is able to learn the above unknowns supervised only by CLIP-bridged text prompts, and generate photorealistic rendering effects by approximating these unknowns during inference. Technically, by jointly encoding the text prompt and images of the given mesh rendered by a spherical Gaussians based differentiable renderer, our method is able to compare the embeddings of text and mesh style and then backpropagate the gradient signals to update the style parameters. Our approach disentangles the style into three components represented by learnable neural networks, namely continuous functions respectively of the point, its normal, and the viewing direction on the surface. Note that due to the learning of neural normal map, our method can produce fine-grained geometric details even on low-quality meshes with only a few faces; in contrast, vertex displacement-based methods, such as [33], often fail in such cases.

We finally summarize our contributions as follows.

- We propose a novel, end-to-end architecture for *text-driven photorealistic 3D style transfer*; our model learns to assign advanced physical lighting appearance and local geometric details to a raw mesh, by leveraging an off-the-shelf, pre-trained CLIP model.
- Our method can automatically predict material, normal map, and lighting condition for any bare mesh prescribed by text description, and can also handle low-quality meshes with only a few faces, due to our prediction of colors and reflectance properties for every intersection point of a camera ray.
- We conduct extensive ablation studies and experiments that show the advantages of our method. Notably, except for more photorealistic renderings, our results have no self-intersection in local geometry details; in contrast, such imperfections would appear in results from existing methods.

2 Related Work

Text-Driven Generation and Manipulation. Our work is primarily inspired by [33] and other methods [46, 21, 42, 22, 24] which use text prompt to guide 3D generation or manipulation. The optimization procedures of these methods are guided by the CLIP model [39]. Specifically, CLIP-NeRF [46] proposes a unified framework that allows manipulating NeRF, guided by either a short text prompt or an examplar image. Another work Dreamfields [21] chooses to generate 3D scenes from scratch in a NeRF representation. Different from those NeRF-based methods, CLIP-Forge [42] presents a method for shape generation represented by voxels. Text2Mesh [33] predicts color and geometric details for a given template mesh according to the text prompt. Concurrently, Khalid *et al.* [24] generate stylized mesh from a sphere guided by CLIP. Meanwhile, apart from 3D content creation, there are other works focusing on text-guided image generation and manipulation. StyleCLIP [38] and VQGAN-CLIP [10] use CLIP-loss to optimize latent codes in GAN's style space. GLIDE [37] is introduced for more photorealistic image generation and editing with text-guided diffusion models, while DAll-E [40] explores to generate images using transformer. Compared to these methods, we focus on improving the realism of text-driven 3D mesh stylization.

3D Style Transfer. Generating or editing 3D content according to a given style is a long-standing task in computer graphics [13, 15, 50]. NeuralCage [51] implements traditional cage-based deformation as differentiable network layers and generates feature-deserving deformations. Furthermore, 3DStyleNet [52] conducts 3D object stylization informed by a reference textured 3D shape. Other works are specific to styles of furniture [29], 3D colleges [13], LEGO [25], and portraits [18]. Compared to these global geometric transfer approaches, some works investigate the style transfer on a more local level. Specifically, Hertz *et al.* [20] propose a generative framework to learn local structures from the style of a given mesh, while Liu *et al.* [27] subdivide the given mesh for a coarse-to-fine geometric modeling. Unlike these methods, we aim to synthesize a wide range of realistic styles specified by a text prompt, making it more convenient to guide the process by high-level semantics.

BRDF and Lighting Estimation. Previously, BRDF and lighting can only be estimated under complicated settings in laboratory environment [2, 30, 48] before the deep learning era. With the help of neural networks, some techniques can use simpler settings to estimate BRDF and lighting beyond geometry [14, 3, 1, 11]. They assume that scenes are under specific illumination, like one [11] or multiple [14] flashlights and linear light [16]. However, they often assume that the geometry is a plane [14] or known [3]. These requirements are difficult to be met in reality. Furthermore, recent works [7, 8, 28, 55] exploit neural networks to jointly predict geometry, BRDF and in some cases, lighting via 2D image loss. These methods can be split into two categories according to the geometry representation. Among them, those who use explicit representation [28, 7, 8] usually deform a sphere to obtain final shapes, which cannot represent arbitrary topology. Another pipeline of methods that use implicit geometry representation like SDF [53], occupancy function [32] or NeRF [44, 4, 54] can get higher quality shapes, but these representations are not handy to be processed by contemporary game engines. Unlike them, our task is to change the style of a given shape, so the explicit triangle mesh is preferred due to its convenience. As for BRDF and lighting representation, PhySG [53] and NeRD [4] use mixtures of spherical Gaussians to represent illumination; NeRFactor [54] and nvdiffrec [36] choose low-resolution envmaps and split sum lighting model, respectively. Different from all these 2D supervision methods, we jointly estimate spatially varying BRDF, lighting, and local geometry (normal map) supervised by text prompts. With the help of estimated BRDF and lighting, we can represent complicated light reflection in real world on stylized meshes.

3 Method

We contribute an end-to-end architecture which enables *text-driven photorealistic 3D style transfer* for any bare mesh, which is supervised by a semantic loss. The heart of our method is to disentangle the style of the input mesh as reflectance properties and scene lighting. Then, given a target style specified by a text prompt, we could learn the corresponding style parameters by leveraging the pre-trained CLIP model, and then the stylized images could be generated with the learned style parameters through our spherical Gaussians (SG) differentiable renderer. There are three unknowns in our defined style parameters: (i) *spatially varying BRDF (SVBRDF)*, including diffuse, roughness, specular, represented by parameters $\boldsymbol{\xi} \in \mathbb{R}^m$; (ii) *normal*, represented by parameters $\boldsymbol{\gamma} \in \mathbb{R}^n$; (iii) *lighting*, represented by parameters $\boldsymbol{\tau} \in \mathbb{R}^l$.

We use explicit triangle mesh representation to represent the input 3D shape. The input template mesh *M* consists of e vertices $\boldsymbol{V} \in \mathbb{R}^{e\times 3}$ and u faces $\boldsymbol{F} \in \{1, ..., n\}^{u\times 3}$ and is fixed throughout training. The object's style is described by a text prompt, and the aforementioned SVBRDF, normal and lighting parameters are optimized to fit the description. Compared to other implicit representation [53, 4], explicit representation here is more convenient to obtain and easy to use by current game engines and other applications.

Forward model. Given an object mesh *M*, we first scale it in a unit sphere and then randomly sample points near *M* as camera positions $\boldsymbol{c}$ to render images. For each pixel in rendered images, indexed by *p*, let $\boldsymbol{R}_p = \{\boldsymbol{c}_p + t\boldsymbol{\nu}_p \mid t \geq 0\}$ denotes the ray through pixel *p*, where $\boldsymbol{c}_p$ is randomly sampled on a sphere containing the object and $\boldsymbol{\nu}_p$ denotes the direction of the ray (i.e. the vector pointing from $\boldsymbol{c}_p$ to *p*). Let $\boldsymbol{x}_p$ denotes the first intersection of the ray $\boldsymbol{R}_p$ and the mesh *M*, $\boldsymbol{n}_p$ denotes the ground truth normal of point $\boldsymbol{x}_p$, $\hat{\boldsymbol{n}}_p = \boldsymbol{\Pi}(\boldsymbol{n}_p, \boldsymbol{x}_p; \boldsymbol{\gamma})$ is the estimated normal on surface point $\boldsymbol{x}_p$. For each surface point $\boldsymbol{x}_p$ with the estimated normal $\hat{\boldsymbol{n}}_p$, we suppose that $\boldsymbol{L}_i(\boldsymbol{\omega}_i; \boldsymbol{\tau})$ is the incident light intensity from direction $\boldsymbol{\omega}_i$, and SVBRDF $\boldsymbol{f}_r(\boldsymbol{\nu}_p, \boldsymbol{\omega}_i, \boldsymbol{x}_p; \boldsymbol{\xi})$ are the surface reflectance coefficients of the material at location $\boldsymbol{x}_p$ from viewing direction $\boldsymbol{\nu}_p$ and incident light direction $\boldsymbol{\omega}_i$. Then the observed light intensity $\boldsymbol{L}_p(\boldsymbol{\nu}_p, \boldsymbol{x}_p)$ is an integral over the hemisphere $\Omega = \{\boldsymbol{\omega}_i : \boldsymbol{\omega}_i \cdot \hat{\boldsymbol{n}}_p \geq 0\}$:

$$\boldsymbol{L}_p(\boldsymbol{\nu}_p, \boldsymbol{x}_p, \boldsymbol{n}_p; \boldsymbol{\xi}, \gamma, \boldsymbol{\tau}) = \int_{\Omega} \boldsymbol{L}_i(\boldsymbol{\omega}_i, \boldsymbol{x}_p; \boldsymbol{\tau}) \boldsymbol{f}_r(\boldsymbol{\nu}_p, \boldsymbol{\omega}_i, \boldsymbol{x}_p; \boldsymbol{\xi})(\boldsymbol{\omega}_i \cdot \boldsymbol{\Pi}(\boldsymbol{n}_p, \boldsymbol{x}_p; \boldsymbol{\gamma}))\mathrm{d}\boldsymbol{\omega}_i. \quad (1)$$

For each image $I \in [0, 1]^{H\times W\times 3}$ corresponding to a camera position $\boldsymbol{c}$, we calculate the pixel color by Equation (1). The rendered image is encoded into a latent code by pre-trained CLIP model, which is expected to be aligned with the encoded text prompt. Finally, a loss measuring the divergence between encoded image and text representations is used to simultaneously train the model parameters, i.e., $\boldsymbol{\xi}, \boldsymbol{\gamma}$, and $\boldsymbol{\tau}$, resulting in photorealistic renderings matching the text description. We summarize our approach in Figure 1 and clarify each step in the following: Sec. 3.1 outlines our intersection

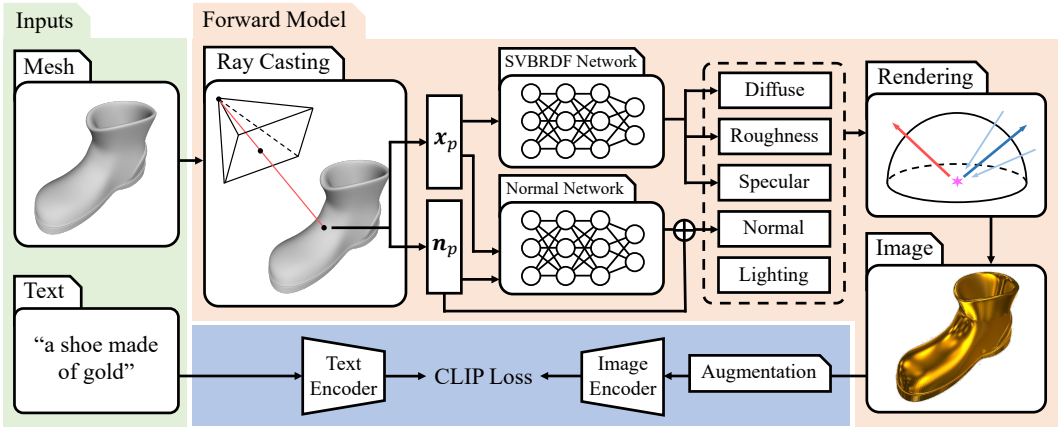

Figure 1: Given a scaled bare mesh and a text prompt (e.g., "a shoe made of gold" as in this figure), we first specify a camera location $\boldsymbol{c}$ and cast rays $\boldsymbol{R}$ to intersect with the given mesh. For each intersecting camera ray, we obtain a surface point $\boldsymbol{x}_p$ and normal $\boldsymbol{n}_p$, which are used for predicting spatially varying BRDF parameters and normal variation by two MLPs, i.e., the SVBRDF Network and Normal Network.. Meanwhile, the lighting is represented using a series of spherical Gaussians parameters $\{\boldsymbol{\mu_k}, \lambda_k, \boldsymbol{a}_k\}$. For each iteration, we render the image using a differentiable SG renderer and then encode the augmented image using the image encoder of a CLIP model, which backpropagates the gradients to update all learnable parameters.

calculation, Sec. 3.2 presents our appearance style model, Sec. 3.3 illustrates the rendering procedure, and the loss function we use to train our network is described in Sec. 3.4.

3.1 Intersection of camera rays and mesh

Given an input 3D mesh M and target text, we first scale M in a unit sphere and find an anchor view on the sphere, then other camera positions are randomly sampled using Gaussian distribution centered around the anchor view, whose direction vectors are set to look at the origin of the unit bounding sphere. Then for each sampled camera position $\boldsymbol{c}$ and a pixel p, we get a camera ray $\boldsymbol{R}_p = \{\boldsymbol{c} + t\boldsymbol{\nu}_p \mid t \geq 0\}$, pointing from $\boldsymbol{c}$ to p. After that, we use ray casting [41] to find the first intersection point $\boldsymbol{x}_p$ and intersection face $\boldsymbol{f}_p$ of $\boldsymbol{R}_p$ and mesh M, while the face normal $\boldsymbol{n}_p$ of $\boldsymbol{f}_p$ is used as surface normal at point $\boldsymbol{x}_p$.

3.2 Appearance modeling

To generate photorealistic style of object M, we use three components in Equation (1) to model appearance: (1) an environment map, (2) a normal map, and (3) SVBRDF consists of diffuse term and specular term. To calculate Equation (1), some works use Monte Carlo sampling [26], split sum approximation [36], spherical Gaussians [53] or spherical Harmonics [17] to solve the integration. In this paper, we choose to use spherical Gaussians (SG) to efficiently approximate the rendering equation in closed form.

An n-dimensional spherical Gaussian is a spherical function that takes following form [47]:

$$G(\boldsymbol{\nu}; \boldsymbol{\mu}, \lambda, \boldsymbol{a}) = \boldsymbol{a} e^{\lambda(\boldsymbol{\nu}\cdot\boldsymbol{\mu}-1)}, \tag{2}$$

where $\boldsymbol{\nu} \in \mathbb{S}^2$ is the function input, $\boldsymbol{\mu} \in \mathbb{S}^2$ is the lobe axis, $\lambda \in \mathbb{R}_+$ is the lobe sharpness, and $\boldsymbol{a} \in \mathbb{R}_+^n$ is the lobe amplitude ($\boldsymbol{a} \in \mathbb{R}_+^3$ for RGB color). Spherical Gaussians have two nice properties to ensure the convenience of calculating integration:

(1) the multiplication of two spherical Gaussians is another spherical Gaussian:

$$G(\boldsymbol{\nu}; \boldsymbol{\mu_1}, \lambda_1, \boldsymbol{a_1}) G(\boldsymbol{\nu}; \boldsymbol{\mu_2}, \lambda_2, \boldsymbol{a_2}) = G(\boldsymbol{\nu}; \frac{\boldsymbol{\mu}_m}{\|\boldsymbol{\mu}_m\|}, (\lambda_1+\lambda_2)\|\boldsymbol{\mu}_m\|, \boldsymbol{a}_1\boldsymbol{a}_2 e^{\lambda_m(\|\boldsymbol{\mu}_m\|-1)}), \tag{3}$$

where $\lambda_m = \lambda_1 + \lambda_2$, $\boldsymbol{\mu}_m = \frac{\lambda_1\boldsymbol{\mu}_1+\lambda_2\boldsymbol{\mu}_2}{\lambda_1+\lambda_2}$.

(2) the integration of a single spherical Gaussian has closed-form solution:

$$\int_{\Omega} G(\boldsymbol{\nu}; \boldsymbol{\mu}, \lambda, \boldsymbol{a}) \mathrm{d}\boldsymbol{\nu} = 2\pi \frac{\boldsymbol{a}}{\lambda}(1 - e^{-2\lambda}). \tag{4}$$

The above two properties guarantee that if we use SG to represent each multiplication term in Equation (1), the formula in integral can be written as a single SG, which can be calculated efficiently.

Environment map. We represent environment map $\boldsymbol{L}_i(\boldsymbol{\omega}_i)$ with multiple spherical Gaussians (SG):

$$\boldsymbol{L}_i(\boldsymbol{\omega}_i) = \sum_{k=0}^{M} G(\boldsymbol{\omega}_i; \boldsymbol{\mu_k}, \lambda_k, \boldsymbol{a}_k). \tag{5}$$

The light energy is:

$$E(\boldsymbol{L}) = \sum_{k=0}^{M} \frac{2\pi \boldsymbol{a}_k}{\lambda_k(1 - e^{-2\lambda_k})}. \tag{6}$$

When energy is too large or too small, the rendered image will be too bright or too dark, resulting in the optimization falling in a local minimum. To alleviate this issue, we initialize the total energy to 1.

Normal map. To provide more geometry details for meshes when rendering, for each $\boldsymbol{x}_p$ and $\boldsymbol{n}_p \in \{(1, \theta_p, \varphi_p) | \theta_p \in (0, 2\pi), \varphi_p \in (0, \pi)\}$, we estimate a normal offset to $\boldsymbol{n}_p$ to get estimated normal $\hat{\boldsymbol{n}}_p \in \left\{(1, \hat{\theta}_p, \hat{\varphi}_p) | \hat{\theta}_p \in (0, 2\pi), \hat{\varphi}_p \in (0, \pi)\right\}$:

$$(\hat{\theta}_p, \hat{\varphi}_p) = (\theta_p + \triangle\theta, \varphi_p + \triangle\varphi) = \boldsymbol{\Pi}(\beta(\boldsymbol{x}_p, \theta_p, \varphi_p); \boldsymbol{\gamma}), \tag{7}$$

$$(\triangle\theta, \triangle\varphi) = \triangle\boldsymbol{\Pi}(\beta(\boldsymbol{x}_p, \theta_p, \varphi_p); \boldsymbol{\gamma}), \tag{8}$$

where $\triangle\boldsymbol{\Pi}$ is a neural network (MLP) estimating the normal offset. For every input, we apply a positional encoding layer $\beta(l) = [\cos(2\pi \boldsymbol{B} l), \sin(2\pi \boldsymbol{B} l)]^T$ to synthesize high-frequency details, where $\boldsymbol{B}$ is a random Gaussian matrix whose entry is randomly drawn from $\mathcal{N}(0, \sigma^2)$ or an arithmetic sequence ranging from 1 to N like NeRF[34]. $(\hat{\theta}_p, \hat{\varphi}_p)$ is clamped to lie in reasonable ranges.

SVBRDF. Below we take apart the SVBRDF $\boldsymbol{f}_r(\boldsymbol{\nu}, \boldsymbol{\omega}_i, \boldsymbol{x}_p)$ into diffuse term $\boldsymbol{f}_d(\boldsymbol{x}_p)$ and specular term $\boldsymbol{f}_s(\boldsymbol{\nu}, \boldsymbol{\omega}_i, \boldsymbol{x}_p)$. For diffuse term, we use an MLP $\boldsymbol{\Phi}_1(\beta(\boldsymbol{x}_p); \boldsymbol{\xi}_1)$ to calculate spatially varying diffuse albedo, where $\beta(l)$ is positional encoding layer. Then diffuse term is

$$\boldsymbol{f}_d(\boldsymbol{x}_p) = \frac{\boldsymbol{\Phi}_1(\beta(\boldsymbol{x}_p); \boldsymbol{\xi}_1)}{\pi}. \tag{9}$$

For specular term, we use simplified Disney BRDF model [5]:

$$\boldsymbol{f}_s(\boldsymbol{\nu}, \boldsymbol{\omega}_i) = \mathcal{M}(\boldsymbol{\nu}, \boldsymbol{\omega}_i)\mathcal{D}(\boldsymbol{\nu}, \boldsymbol{\omega}_i), \tag{10}$$

where $\mathcal{M}$ accounts for Fresnel and shadowing effects, and $\mathcal{D}$ is normal distribution. Following [53], we use spherical Gaussians to represent specular term, so the specular SVBRDF function can be written as:

$$\boldsymbol{f}_s(\boldsymbol{\nu}, \boldsymbol{\omega}_i, \boldsymbol{x}_p) = \sum_{k=0}^{N} G(\boldsymbol{h}; \hat{\boldsymbol{n}}_p, \frac{\lambda_k}{4\boldsymbol{h} \cdot \boldsymbol{\nu}}, \mathcal{M}_p \boldsymbol{a}_k), \tag{11}$$

where the half vector $\boldsymbol{h} = \frac{\boldsymbol{\nu} + \boldsymbol{\omega}_i}{\|\boldsymbol{\nu} + \boldsymbol{\omega}_i\|_2}$, the approximated Fresnel and shadowing term $\mathcal{M}_p \approx \mathcal{M}(\boldsymbol{\nu}, 2(\boldsymbol{\nu} \cdot \hat{\boldsymbol{n}}_p)\hat{\boldsymbol{n}}_p - \boldsymbol{\nu})$. $\{\lambda_k, \boldsymbol{a}_k\}$ are predicted by an MLP $\boldsymbol{\Phi}_2(\beta(\boldsymbol{x}_p); \boldsymbol{\xi}_2)$. The complete SVBRDF function is

$$\boldsymbol{f}_r(\boldsymbol{\nu}, \boldsymbol{\omega}_i, \boldsymbol{x}_p) = \boldsymbol{f}_d(\boldsymbol{x}_p) + \boldsymbol{f}_s(\boldsymbol{\nu}, \boldsymbol{\omega}_i, \boldsymbol{x}_p). \tag{12}$$

Following [31], we use an SG to approximate term $\boldsymbol{\nu} \cdot \hat{\boldsymbol{n}}_p = G(\boldsymbol{\nu}; 0.0315, \hat{\boldsymbol{n}}_p, 32.7080) - 31.7003$. Note that all components in Equation (1) are represented by SGs, so the integration can be calculated in closed form.

3.3 Rendering

Given our geometric and appearance components, we perform rendering of a ray $\boldsymbol{R}_p$'s color as follows: (1) find the intersection point $\boldsymbol{x}_p$ and normal $\boldsymbol{n}_p$ between $\boldsymbol{R}_p$ and the mesh using ray casting; (2) predict the normal offset and get the estimated normal $\hat{\boldsymbol{n}}_p$; (3) compute the diffuse albedo $\boldsymbol{\Phi}_1(\boldsymbol{x}_p; \boldsymbol{\xi}_1)$ and specular albedo $\boldsymbol{\Phi}_2(\boldsymbol{x}_p; \boldsymbol{\xi}_2)$; (4) use estimated normal $\hat{\boldsymbol{n}}_p$, environment map $\{\boldsymbol{\mu}_k, \lambda_k, \boldsymbol{a}_k\}$, SVBRDF $\boldsymbol{\Phi}(\boldsymbol{x}_p; \boldsymbol{\xi})$, and viewing direction $\boldsymbol{\nu}_p$ to compute the color for $\boldsymbol{R}_p$; (5) conduct steps (1)-(4) repeatedly for every pixel and then finally gather them as the rendered image. The overall pipeline is illustrated in Figure 1. Note that our pipeline is fully differentiable w.r.t. all the optimizable parameters.

3.4 Text-based correspondence loss

Our optimization is guided by a pre-trained CLIP [39] model. For each iteration, we sample a batch of camera positions. For each camera position $\boldsymbol{c}$, we can render an image $\boldsymbol{I} \in [0,1]^{H \times W \times 3}$ whose pixel's color is calculated by Equation (1) and a mask $\boldsymbol{O} \in \{0,1\}^{H \times W}$. Then we use the mask $\boldsymbol{O}$ to add background to image $\boldsymbol{I}$, where each pixel value in the background is randomly sampled from the normal distribution $\mathcal{N}(0, \sigma^2)$. Next, in 2D augmentation phase, we randomly crop some areas of $\boldsymbol{I}$ and resized them to (224, 224), resulting in the augmentations of $\boldsymbol{I}_a$. Subsequently, $\boldsymbol{I}$ and augmented images $\boldsymbol{I}_a$ are all encoded into the multimodal space by the image encoder of CLIP, generating an averaged latent code $\mathcal{L}_i \in \mathbb{R}^{512}$. Meanwhile, the input text prompt is encoded to $\mathcal{L}_t \in \mathbb{R}^{512}$ by the text encoder of CLIP. Finally, we use the following cosine similarity as the overall objective:

$$Loss(\mathcal{L}_i, \mathcal{L}_t) = -\frac{\mathcal{L}_i \cdot \mathcal{L}_t}{\|\mathcal{L}_i\|_2 \|\mathcal{L}_t\|_2}. \tag{13}$$

Implementation details. The environment map is represented by 32 or 64 spherical Gaussians, and the specular BRDF is represented by 1 SG. The normal estimation network $\boldsymbol{\Pi}(\boldsymbol{x}_p, \theta_p, \varphi_p; \boldsymbol{\gamma}) \in \mathbb{R}^2$ consists of 3 layers with hidden layers of width 256. The SVBRDF network, $\boldsymbol{\Phi}(\boldsymbol{x}_p; \boldsymbol{\xi}) \in \mathbb{R}^7$, with hidden layers of width 256, predicts diffuse, specular and roughness per surface point; the three ones share the first 2 layers of $\boldsymbol{\Phi}(\boldsymbol{x}_p; \boldsymbol{\xi})$ and the parameters in next 3 layers are exclusive. We use the pre-trained CLIP model with ViT-B/32 backbone. We adopt the AdamW optimizer and initial learning rate is set as 5×10^{-4}. The learning rate is decayed by 0.7 in every 500 iterations.

4 Experiment

Following Text2Mesh[33], we examine our method in a variety of meshes and different text prompts. The object meshes are collected from: COSEG[43], Thingi10K[56], ShapeNet[6], Turbo Squid[45], and ModelNet[49]. To test our method's robustness to low-quality meshes, we downsample the number of meshes' vertices and faces in Sec. 5, while in other experiments we directly stylize the given mesh without any pre-processing. Except in Sec. 5, meshes used in our paper contain an average of $73,966$ faces, 16% non-manifold edges, 0.2% non-manifold vertices, and 12% boundaries. Before stylizing the given mesh, we scale the input into a unit sphere and move the center of its vertices to the origin. Our method is trained on a single Nvidia RTX 3090 GPU and takes about 10 minutes for 1500 iterations.

In Sec. 4.1, we examine our method on different meshes with text prompts and compare our results to state-of-the-art text-driven mesh stylization approach Text2Mesh[33]. After that, in Sec. 4.2, we degenerate the quality of meshes by downsampling the number of faces to 5000, verifying the robustness of our method to sparse mesh data. Next in Sec. 4.3, we conduct the ablation study to explore the impact of our proposed modules and the different backbones of CLIP model.

4.1 Neural stylization and controls

Our method generates photorealistic details with fine granularity and avoid self-intersection in the meantime. As presented in Figure 2, our method not only presents realistic reflective effects like highlight and shade in smooth materials like gold and silver, but also tackles uneven surfaces by producing dark and bright shadings via the estimated normals. Furthermore, in Figure 3, our approach demonstrates a consistent 3D stylization in different views and exhibits natural variation in texture.

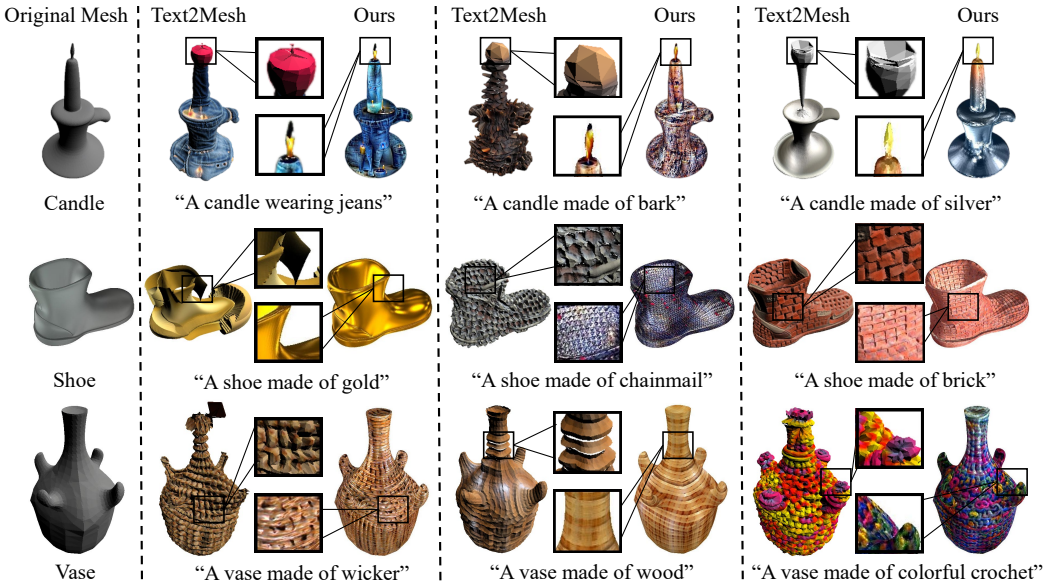

Figure 2: 3D stylization results of our method and Text2Mesh given the same mesh and text prompt. Our method produces results of higher fidelity and more realistic rendering. See Sec. 4.1 for the details.

Table 1: Mean opinion scores (1-5) for Q1-Q3 (cf. Sec. 4.1), for our method and baseline.

	(Q1): Overall	(Q2): Content	(Q3): Style
Text2Mesh	3.30($\pm$0.75)	3.53($\pm$0.79)	3.42($\pm$0.66)
Ours	**4.02**($\pm$0.77)	**3.98**($\pm$0.74)	**3.94**($\pm$0.75)

Similar to Text2Mesh, we incorporate 3D symmetry prior for those meshes that need style symmetry across axis, like person shown in Figure 3. Specifically, for each point $\boldsymbol{x}_p = (x, y, z)$ and a shape with bilateral symmetry across the X-Y plane, we apply a symmetry prior to the positional encoder, i.e., replacing $\beta(x, y, z)$ as $\beta(x, y, |z|)$. With this symmetry prior, the stylized Iron Man in Figure 3 is symmetrical across z-axis, which satisfies the real situation for a person. Meanwhile, because our method disentangles geometry, material and lighting representation, we can relight the stylized mesh. In Figure 4, we change the style of stylized shoe mesh, which is supervised by text prompt "a shoe made of gold", by replacing the lighting spherical Gaussians parameters $\{\boldsymbol{\mu_k}, \lambda_k, \boldsymbol{a}_k\}$.

Comparison with Text2Mesh. In this part we compare our method with a pioneering work in text-driven 3D stylization, Text2Mesh[33], which operates on the point colors and vertex's displacement. We use official implementation of Text2Mesh and also train it for 1500 iterations for a fair comparison. Compared to Text2Mesh, our method can handle both smooth mental appearance and rough materials, generating realistic renderings with highlight effects. For example in Figure 2, given a bare shoe mesh and a text prompt "a shoe made of gold", our method can produce realistic golden reflectance that has light and shade contrast, which is not observed in the competitive Text2Mesh. For materials that need apparent bump, for example, "a vase made of wicker", our approach successfully generates bump materials without incorporating local self-intersection. In addition, we randomly chose 73 users to evaluate 9 source meshes and style text prompt combinations. Each of them was asked three questions: (Q1) "How natural is the output results?"(Q2) "How well does the output match the original content?" (Q3) "How well does the output match the target style?" We report the the mean opinion scores with standard deviations for both Text2Mesh and ours in Table 1. Our method outperforms the Text2Mesh baseline across all questions, with an advantage of 0.72, 0.45, and 0.52 for Q1-Q3, respectively.

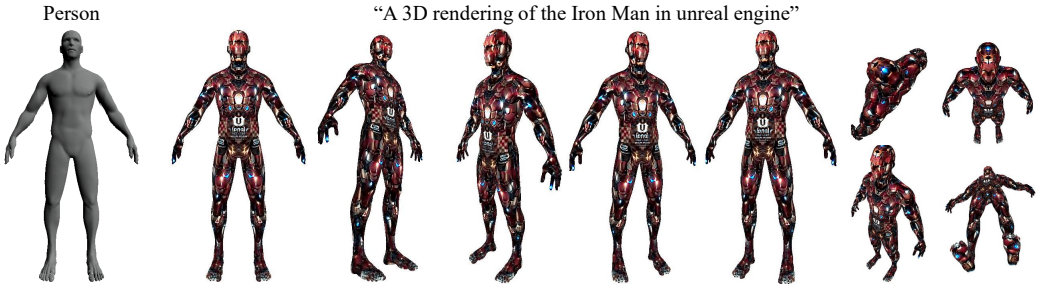

Figure 3: Given a bare person mesh and text prompt, our method can stylize the entire 3D Iron Man appearance and therefore generates 3D-consistent renderings.

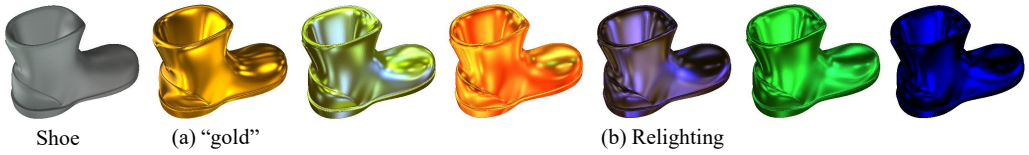

Figure 4: Incorporating lighting in our SG renderer allows the disentanglement of accurate lighting and physical reflectance properties.

4.2 Robustness to low-quality meshes

To measure the robustness of our approach to low-quality meshes, we show stylization results of original meshes and downsampled meshes in Figure 5. The original lamp mesh and alien mesh have $41,160$ faces and $68,430$ faces, respectively. To generate the low-quality meshes, we both downsample them to $5,000$ faces using Meshlab [9]. On low-quality meshes with less faces, our method still maintains the similar performance to the original mesh, while Text2Mesh is trapped in self-intersection and monotonous color prediction. The reason for the phenomenon is that we make spatially varying prediction for each intersection point of the camera ray and mesh, and the local geometric variation compensates for the lack of geometric details in low-quality meshes. As for displacement-based methods like Text2Mesh, they rely on moving each vertex's position to generate bumpy geometry, which makes them deviate with vertices number shrinking.

4.3 Ablation study

To verify the effectiveness of each module in our approach, we conduct ablation study by removing different components from our method. The results are shown in Figure 6. Compared to the full model, when we remove spatially varying specular and roughness prediction (*-spec, rough* in Figure

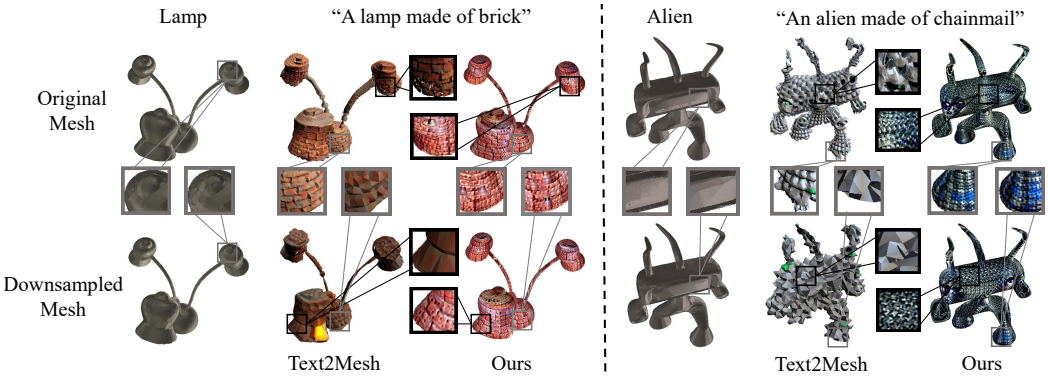

Figure 5: A robustness comparison of our method and Text2Mesh on high-quality meshes (cf. the top row) and downsampled meshes (cf. the second row). The results of Text2Mesh degenerates significantly as the mesh quality downgrades, while our method presents consistent results on meshes of various qualities.

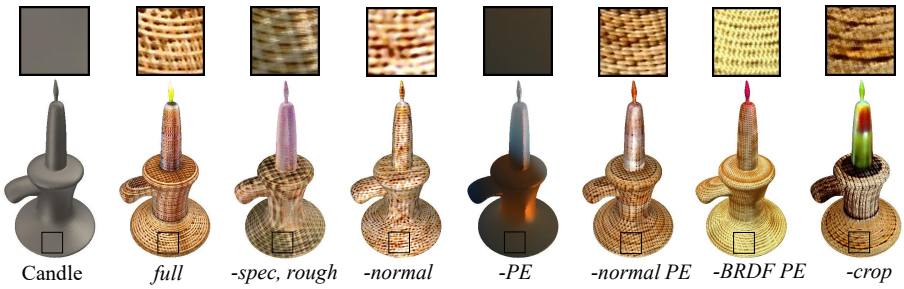

Figure 6: Ablation experiments on the proposed modules in our method, where the text prompt is "A candle made of wicker". Please refer to the Sec. 4.3 for the detailed definitions.

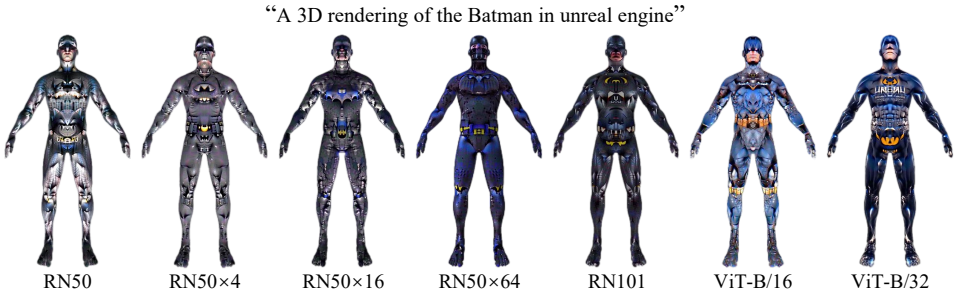

Figure 7: Qualitative results of varying backbone network in CLIP. RN stands for ResNet[19] and ViT represents Vision Transformer [12]. The results with transformer-based CLIP present more natural appearance than these based on ResNet.

6), the whole SVBRDF is only represented by diffuse color and therefore it cannot generate realistic view-dependent highlight in renderings. If we disable normal estimation network (*-normal* in Figure 6), the stylized appearance becomes smooth and fails to generate enough geometric variations, verifying the effectiveness of the normal estimation network on generating geometric bump details. To provide enough high-frequency details when stylizing, we use the positional encoding layer to wrap the input points and normals for both SVBRDF prediction and the normal estimation network. If we remove positional encoder (*-PE*, *-normal PE* and *-BRDF PE* in Figure 6), our results will lack in high-frequency variation in both color and geometry. In addition, local crop augmentation (*-crop* in Figure 6) also contributes obviously to the stylized performance, since it makes our network focus on small areas on surface and avoid blurred results. Moreover, we ablate the backbone network used in CLIP model. As illustrated in Figure 7, the results with transformer-based CLIP present more natural appearance than these based on ResNet.

5 Limitations and conclusions

Our main limitation is the simplified shading model to represent indirect lighting effects and shadow, impeding our method from representing complicated lighting in large scenes which need to simulate multiple light bounces. The reason for choosing spherical Gaussians to represent our shading is to accelerate the rendering and training process. To extend our model for large scene stylization, indirect lighting term and acceleration techniques can be incorporated to guarantee both quality and speed. We will investigate these problems in future research.

In this paper, we propose a novel framework to generate photorealistic appearance style for an arbitrary 3D mesh according to a text prompt. By disentangling the appearance style as SVBRDF, local geometric variation, and lighting condition, we jointly learn them under the supervision of the CLIP loss via a spherical Gaussians based differentiable renderer. Compared to existing competitors, our method could automatically predict reflectance effects even for bare, low-quality meshes. Our method can be applied as a style transfer tool for bare meshes and a data augmentation technique for large-scale appearance generation, which will save much time and labor for experts. However, on the other hand, it will possibly make junior artists responsible for creating simple texture and lighting for bare meshes out of employment.

References

[1] Miika Aittala, Timo Aila, and Jaakko Lehtinen. Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics (ToG)*, 35(4):1–13, 2016.

[2] Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. Practical SVBRDF capture in the frequency domain. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 32(4):110:1–110:12, August 2013.

[3] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824*, 2020.

[4] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. Nerd: Neural reflectance decomposition from image collections. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12684–12694, 2021.

[5] Brent Burley and Walt Disney Animation Studios. Physically-based shading at disney. In *ACM SIGGRAPH*, volume 2012, pages 1–7. vol. 2012, 2012.

[6] 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.

[7] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. *Advances in Neural Information Processing Systems*, 32, 2019.

[8] Wenzheng Chen, Joey Litalien, Jun Gao, Zian Wang, Clement Fuji Tsang, Sameh Khamis, Or Litany, and Sanja Fidler. Dib-r++: Learning to predict lighting and material with a hybrid differentiable renderer. *Advances in Neural Information Processing Systems*, 34, 2021.

[9] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, Guido Ranzuglia, et al. Meshlab: an open-source mesh processing tool. In *Eurographics Italian chapter conference*, volume 2008, pages 129–136. Salerno, Italy, 2008.

[10] Katherine Crowson, Stella Biderman, Daniel Kornis, Dashiell Stander, Eric Hallahan, Louis Castricato, and Edward Raff. Vqgan-clip: Open domain image generation and editing with natural language guidance. *arXiv preprint arXiv:2204.08583*, 2022.

[11] Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (ToG)*, 37(4):1–15, 2018.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[13] Ran Gal, Olga Sorkine, Tiberiu Popa, Alla Sheffer, and Daniel Cohen-Or. 3d collage: Expressive non-realistic modeling. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 7–14, 2007.

[14] Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images. *ACM Trans. Graph.*, 38(4):134:1–134:15, July 2019.

[15] Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. Automatic unpaired shape deformation transfer. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[16] Andrew Gardner, Chris Tchou, Tim Hawkins, and Paul Debevec. Linear light source reflectometry. *ACM Transactions on Graphics (TOG)*, 22(3):749–758, 2003.

[17] Robin Green. Spherical harmonic lighting: The gritty details. In *Archives of the game developers conference*, volume 56, page 4, 2003.

[18] Fangzhou Han, Shuquan Ye, Mingming He, Menglei Chai, and Jing Liao. Exemplar-based 3d portrait stylization. *IEEE Transactions on Visualization and Computer Graphics*, 2021.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Deep geometric texture synthesis. *ACM Trans. Graph.*, 39(4), jul 2020.

[21] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. *arXiv preprint arXiv:2112.01455*, 2021.

[22] Nikolay Jetchev. Clipmatrix: Text-controlled creation of 3d textured meshes. *arXiv preprint arXiv:2109.12922*, 2021.

[23] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3907–3916, 2018.

[24] Nasir Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Text to mesh without 3d supervision using limit subdivision. *arXiv preprint arXiv:2203.13333*, 2022.

[25] Kyle Lennon, Katharina Fransen, Alexander O'Brien, Yumeng Cao, Matthew Beveridge, Yamin Arefeen, Nikhil Singh, and Iddo Drori. Image2lego: Customized lego set generation from images. *arXiv preprint arXiv:2108.08477*, 2021.

[26] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.

[27] Hsueh-Ti Derek Liu, Vladimir G Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. Neural subdivision. *arXiv preprint arXiv:2005.01819*, 2020.

[28] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7708–7717, 2019.

[29] Zhaoliang Lun, Evangelos Kalogerakis, Rui Wang, and Alla Sheffer. Functionality preserving shape style transfer. *ACM Transactions on Graphics (TOG)*, 35(6):1–14, 2016.

[30] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3):759–769, July 2003.

[31] Julian Meder and Beat Brüderlin. Hemispherical gaussians for accurate light integration. In *International Conference on Computer Vision and Graphics*, pages 3–15. Springer, 2018.

[32] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[33] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. *arXiv preprint arXiv:2112.03221*, 2021.

[34] 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*, pages 405–421. Springer, 2020.

[35] Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah. Differentiable image parameterizations. *Distill*, 3(7):e12, 2018.

[36] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. *arXiv preprint arXiv:2111.12503*, 2021.

[37] 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.

[38] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2085–2094, 2021.

[39] 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.

[40] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[41] Scott D Roth. Ray casting for modeling solids. *Computer graphics and image processing*, 18(2):109–144, 1982.

[42] Aditya Sanghi, Hang Chu, Joseph G Lambourne, Ye Wang, Chin-Yi Cheng, and Marco Fumero. Clip-forge: Towards zero-shot text-to-shape generation. *arXiv preprint arXiv:2110.02624*, 2021.

[43] Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, pages 1–10, 2011.

[44] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7495–7504, 2021.

[45] TurboSquid. Turbosquid 3d model repository. `https://www.turbosquid.com/`, 2022.

[46] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. *arXiv preprint arXiv:2112.05139*, 2021.

[47] Jiaping Wang, Peiran Ren, Minmin Gong, John Snyder, and Baining Guo. All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph.*, 28(5):1–10, dec 2009.

[48] Hongzhi Wu, Julie Dorsey, and Holly Rushmeier. A sparse parametric mixture model for btf compression, editing and rendering. *Computer Graphics Forum*, 30(2):465–473, 2011.

[49] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[50] Jie Yang, Kaichun Mo, Yu-Kun Lai, Leonidas J Guibas, and Lin Gao. Dsm-net: Disentangled structured mesh net for controllable generation of fine geometry. *arXiv preprint arXiv:2008.05440*, 2(3), 2020.

[51] Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 75–83, 2020.

[52] Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 3dstylenet: Creating 3d shapes with geometric and texture style variations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12456–12465, 2021.

[53] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5453–5462, 2021.

[54] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Transactions on Graphics (TOG)*, 40(6):1–18, 2021.

[55] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image gans meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. *arXiv preprint arXiv:2010.09125*, 2020.

[56] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] see Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] see Section 5
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code and data will be released after acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Implementation details in Section 3 and Section 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] see Table 1.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] see Implementation details in Section 3.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]