Text2Mesh: Text-Driven Neural Stylization for Meshes

Oscar Michel[1*] Roi Bar-On[1,2*] Richard Liu[1*] Sagie Benaim[2] Rana Hanocka[1]

[1]University of Chicago [2]Tel Aviv University

Figure 1. Text2Mesh produces color and geometric details over a variety of source meshes, driven by a target text prompt. Our stylization results coherently blend unique and ostensibly unrelated combinations of text, capturing both global semantics and part-aware attributes.

Abstract

In this work, we develop intuitive controls for editing the style of 3D objects. Our framework, Text2Mesh, stylizes a 3D mesh by predicting color and local geometric details which conform to a target text prompt. We consider a disentangled representation of a 3D object using a fixed mesh input (content) coupled with a learned neural network, which we term a neural style field network (NSF). In order to modify style, we obtain a similarity score between a text prompt (describing style) and a stylized mesh by harnessing the representational power of CLIP. Text2Mesh requires neither a pre-trained generative model nor a specialized 3D mesh dataset. It can handle low-quality meshes (non-manifold, boundaries, etc.) with arbitrary genus, and does not require UV parameterization. We demonstrate the ability of our technique to synthesize a myriad of styles over a wide variety of 3D meshes. Our code and results are available in our project webpage: https://threedle.github.io/text2mesh/.

1. Introduction

Editing visual data to conform to a desired style, while preserving the underlying content, is a longstanding objective in computer graphics and vision [14,21,23,24,30]. Key challenges include proper formulation of content, style, and the constituents for representing and modifying them.

To edit the style of a 3D object, we adapt a formulation of geometric content and stylistic appearance commonly used in computer graphics pipelines [2]. We consider *content* as the global structure prescribed by a 3D mesh, which defines the overall shape surface and topology. We consider *style* as the object's particular appearance or affect, as determined by its color and fine-grained (local) geometric details. We propose expressing the desired style through natural language (a text prompt), similar to how a commissioned artist is provided a verbal or textual description of the desired work. This is facilitated by recent developments in joint embeddings of text and images with CLIP [44]. A natural cue for modifying the appearance of 3D shapes is through 2D projections, as they correspond with how humans and machines perceive 3D geometry. We use a neural network to synthesize color and local geometric details over the 3D input shape, which we refer to as a *neural style field* (NSF). The weights of the NSF network are optimized such that the resulting 3D stylized mesh adheres to the style described by text. In particular, our neural optimization is guided by multiple 2D (CLIP-embedded) views of the stylized mesh matching our target text. Results of our technique, called Text2Mesh, are shown in Fig. 1. Our method produces different colors and local deformations for the same 3D mesh content to match the specified text. Moreover, Text2Mesh produces structured textures that are aligned with salient features (*e.g.* bricks in Fig. 2), without needing to estimate sharp 3D curves or a mesh parameterization [33,52]. Our method also demonstrates global understanding; *e.g.* in Fig. 3 human body parts are stylized in accordance with their semantic role. We use the weights of the NSF network to encode a stylization (*e.g.* color and displacements) over the *explicit* mesh surface. Meshes faithfully portray 3D shapes and can accurately represent sharp, extrinsic features

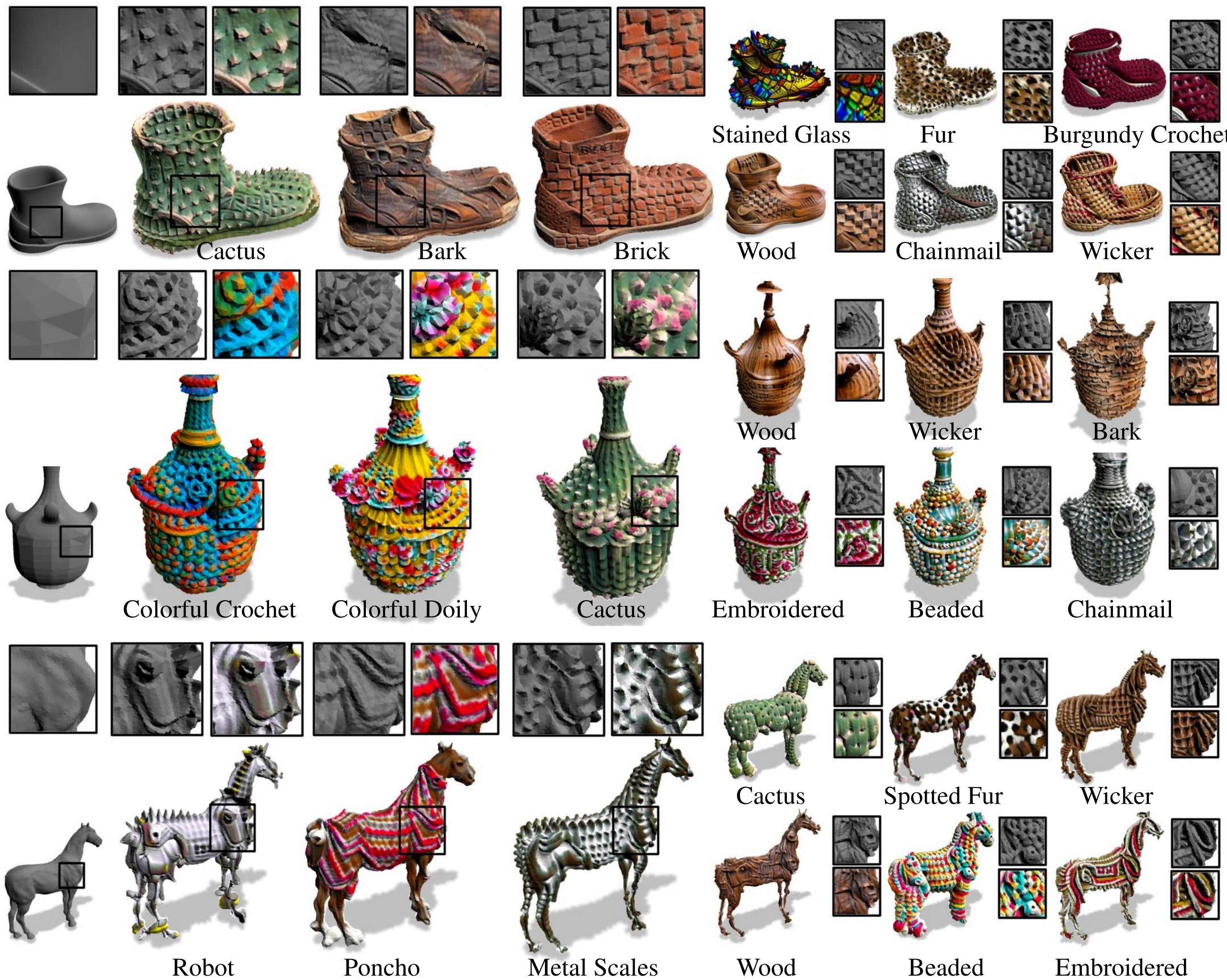


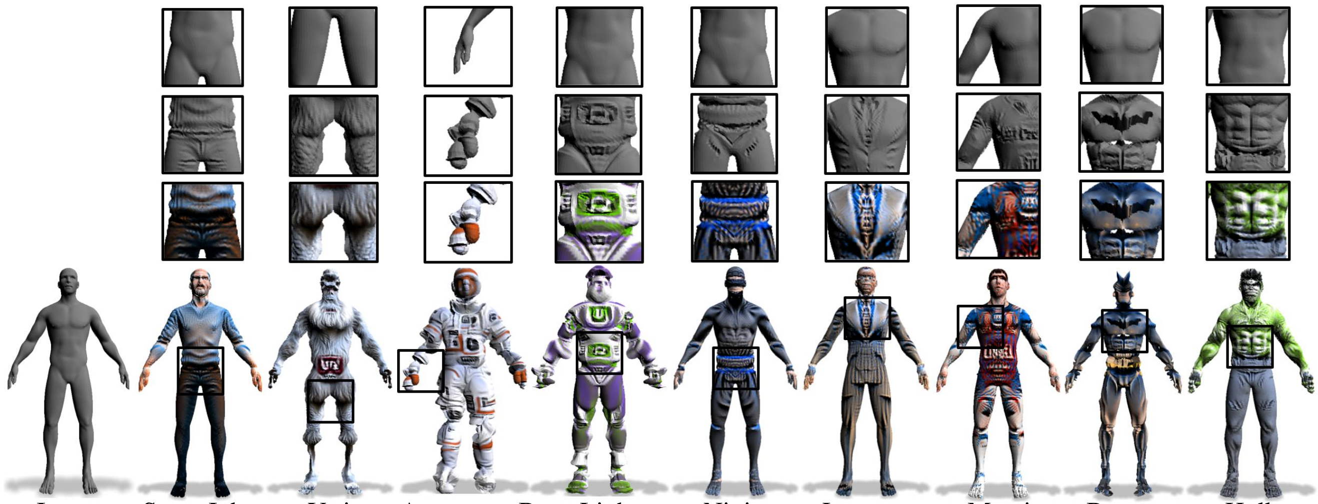
Figure 2. Given a source mesh (gray), our method produces stylized meshes (containing color and local geometric displacements) which conform to various target texts. Insets show a close up of the stylization (with color), and the underlying geometry produced by the deformation component (without color). Insets of the source mesh are also shown on the left most column.

using a high level of detail. Our neural style field is *complementary* to the mesh content, and appends colors and small displacements to the input mesh. Specifically, our neural style field network maps points on the mesh surface to style attributes (*i.e.*, RGB colors and displacements).

We guide the NSF network by rendering the stylized 3D mesh from multiple 2D views and measuring the similarity of those views against the target text, using CLIP's embedding space. However, a straightforward optimization of the 3D stylized mesh which maximizes the CLIP similarity score converges to a degenerate (*i.e.* noisy) solution (see Fig. 5). Specifically, we observe that the joint text-image embedding space contains an abundance of *false positives*, where a valid target text and a degenerate image (*i.e.* noise, artifacts) result in a high similarity score. Therefore, employing CLIP for stylization requires careful regularization.

We leverage multiple *priors* to effectively guide our NSF network. The 3D mesh input acts as a *geometric prior* that imposes global shape structure, as well as local details that indicate the appropriate position for stylization. The weights of the NSF network act as a *neural prior* (*i.e.* regularization technique), which tends to favor smooth solutions [19, 46, 58]. In order to produce accurate styles which contain high-frequency content with high fidelity, we use a frequency-based positional encoding [56]. We garner a strong signal about the quality of the neural style field by rendering the stylized mesh from multiple 2D views and then applying 2D augmentations. This results in a system which can effectively avoid degenerate solutions, while still maintaining high-fidelity results.

The focus of our work is text-driven stylization, since text is easily modifiable and can effectively express complex concepts related to style. Text prescribes an abstract notion of style, allowing the network to produce different



Input Steve Jobs Yeti Astronaut Buzz Lightyear Ninja Lawyer Messi Batman Hulk

Figure 3. Given the same input bare mesh, our neural style field network produces deformations for outerwear of various types (capturing fine details such as creases in clothing and complementary accessories), and distinct features such as muscle and hair. The synthesized colors consider both local geometric details and global part-aware semantics. Insets of the source mesh are shown in the top row and insets of the stylized output are shown in the middle (uncolored) and bottom (colored) rows.

valid stylizations which still adhere to the text. Beyond text, our framework extends to additional target modalities, such as images, 3D meshes, or even cross-modal combinations.

In summary, we present a technique for the semantic manipulation of style for 3D meshes, harnessing the representational power of CLIP. Our system combines the advantages of *explicit* mesh surfaces and the generality of neural fields to facilitate intuitive control for stylizing 3D shapes. A notable advantage of our framework is its ability to handle low-quality meshes (*e.g.*, non-manifold) with arbitrary genus. We show that Text2Mesh can stylize a variety of 3D shapes with many different target styles.

2. Related Work

Text-Driven Manipulation. Our work is similar in spirit to image manipulation techniques controlled through textual descriptions embedded by CLIP [44]. CLIP learns a joint embedding space for images and text. StyleCLIP [43] perform CLIP-guided image editing using a pre-trained StyleGAN [26, 27]. VQGAN-CLIP [9, 10, 45] leverage CLIP for text-guided image generation. Concurrent work uses CLIP to fine-tune a pre-trained StyleGAN [12], and for image stylization [6]. Another concurrent work uses the ShapeNet dataset [5] and CLIP to perform unconditional 3D voxel generation [48]. The above techniques leverage a pre-trained generative network or a dataset to avoid the degenerate solutions common when using CLIP for synthesis. The first to leverage CLIP for synthesis without the need for a pre-trained network or dataset is CLIPDraw [11]. CLIPDraw generates text-guided 2D vector graphics, which conveys a type of drawing style through vector strokes. Concurrent work [25] uses CLIP to optimize over parameters of the SMPL human body model to create digital creatures. Prior to CLIP, text-driven control for deforming 3D shapes was explored [67, 68] using specialized 3D datasets.

Geometric Style Transfer in 3D. Some approaches analyze 3D shapes and identify similarly shaped geometric elements and parts which differ in style [22, 32, 37, 61, 66]. Others transfer geometric style based on content/style separation [4, 8, 49, 63]. Other approaches are specific to categories of furniture [38], 3D collages [13], LEGO [31], and portraits [16]. 3DStyleNet [64] edits shape content with a part-aware low-frequency deformation and synthesizes colors in a texture map, guided by a target mesh. Mesh Renderer [28] changes color and geometry driven by a target image. Liu et al. [36] stylize a 3D shape by adding geometric detail (without color), and ALIGNet [17] deforms a template shape to a target one. The above methods rely on 3D datasets, while other techniques use a single mesh exemplar for synthesizing geometric textures [20] or producing mesh refinements [35]. Shapes can be edited to contain cubic stylization [34], or stripe patterns [29]. Unlike these methods, we consider a wide range of styles, guided by an intuitive and compact (text) specification.

Texture Transfer in 3D. Aspects of a 3D mesh style can be controlled by texturing a surface through mesh parameterization [15, 33, 52, 55]. However, most parameterization approaches place strict requirements on the quality of the input mesh (e.g., a manifold, non-intersecting, and low/zero genus), which do not hold for most meshes in the wild [51]. We avoid parameterization altogether and opt to modify appearance using a neural field which provides a style value

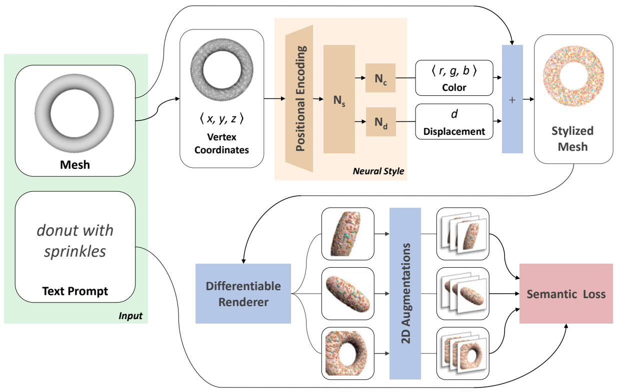


Figure 4. Text2Mesh modifies an input mesh to conform to the target text by predicting color and geometric details. The weights of the neural style network are optimized by rendering multiple 2D images and applying 2D augmentations, which are given a similarity score to the target from the CLIP-based semantic loss.

(*i.e.*, an RGB value and a displacement) for every vertex on the mesh. Recent work explored a neural representation of texture [41], here we consider both color and local geometry changes for the manipulation of style.

Neural Priors and Neural Fields. A recent line of work leverages the inductive bias of neural networks for tasks such as image denoising [58], surface reconstruction [18, 19], point cloud consolidation [39], image synthesis, and editing [50, 53, 70]. Our framework leverages the inductive bias of neural networks to act as a prior which guides Text2Mesh away from degenerate solutions present in the CLIP embedding space. Specifically, our stylization network acts as a neural prior, which leverages positional encoding [56] to synthesize fine-grained stylization details.

NeRF [40] and follow ups [42,65,69] have demonstrated success on 3D scene modeling. They leverage a neural field to represent 3D objects using network weights. However, neural fields *entangle* geometry and appearance, which limits separable control of content and style. They struggle to accurately portray sharp features, are slow to render, and difficult to edit. Thus, several techniques were proposed enabling ease of control [62], and introducing acceleration strategies [47]. Instead, we use a disentangled representation of a 3D object using an *explicit* mesh shape representation and a neural style field which controls appearance. This avoids parametrization, and can be used to manipulate appearance and generate high resolution outputs.

3. Method

An illustration of our method is provided in Fig. 4. As an overview, the 3D object *content* is defined by an input mesh M with vertices $V \in \mathbb{R}^{n\times 3}$ and faces $F \in \{1,\dots,n\}^{m\times 3}$, and is fixed throughout training. The object's *style* (color and local geometry) is modified to conform to a target text prompt t, resulting in a stylized mesh M^S. The NSF learns to map points on the mesh surface $p \in V$ to an RGB color and displacement along the normal direction. We render M^S from multiple views and apply 2D augmentations that are embedded using CLIP. The CLIP similarity between the rendered and augmented images and the target text is used as a signal to update the neural network weights.

3.1. Neural Style Field Network

Our NSF network predicts a style attribute for every vertex, resulting in a *style field* defined over the entire shape surface. Our style field is encoded by an MLP, which maps a point $p \in V$ on the mesh M to a color and displacement along the surface normal $(c_p, d_p) \in (\mathbb{R}^3, \mathbb{R})$. This formulation tightly couples the style field to the source mesh, enabling local geometric modifications.

In practice, we treat the given vertices of M as query points into this field, and use a differentiable renderer to visualize the style over the given triangulation. Increasing the number of triangles in M for learning a higher resolution neural field is trivial, *e.g.*, by inserting a degree 3 vertex (see Supplementary B). Even using a standard GPU (11GB of VRAM) our method handles meshes with up to 180K triangles. We are able to render stylized objects using very high resolutions, as shown in Supplementary B.

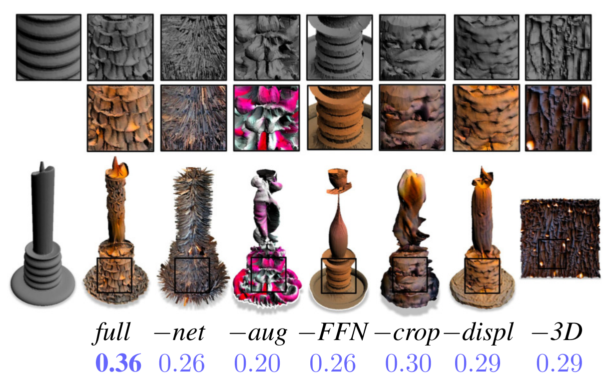


Figure 5. Ablation on the priors used in our method (*full*) for a candle mesh and target 'Candle made of bark': w/o our style field network (*−net*), w/o 2D augmentations (*−aug*), w/o positional encoding (*−FFN*), w/o crop augmentations for ψ_{local} (*−crop*), w/o the *geometry-only* component of L_{sim} (*−displ*), and learning over a 2D plane in 3D space (*−3D*). We show the CLIP score ($\text{sim}(\hat{S}^{\text{full}}, \phi_{\text{target}})$); see Sec. 3 for more details.

Since our NSF uses low-dimensional coordinates as input to an MLP, this exhibits a spectral bias [46] toward smooth solutions (*e.g.* see Fig. 5). To synthesize high-frequency details, we apply a positional encoding using Fourier feature mappings, which enables MLPs to overcome the spectral bias and learn to interpolate high-frequency functions [56]. For every point p its positional encoding $\gamma(p)$ is given by:

$$\gamma(p) = [\cos(2\pi \mathbf{B} p), \sin(2\pi \mathbf{B} p)]^{\mathrm{T}} \tag{1}$$

where $\mathbf{B} \in \mathbb{R}^{n\times 3}$ is a random Gaussian matrix where each entry is randomly drawn from $\mathcal{N}\left(0, \sigma^2\right)$. The value of σ is chosen as a hyperparameter which controls the frequency of the learned style function. We show in Sec. 4.1 that this allows for user control over the frequency of the output style.

First, we normalize the coordinates $p \in V$ to lie inside a unit bounding box. Then, the per-vertex positional encoding features $\gamma(p)$ are passed as input to an MLP N_s, which then branches out to MLPs N_d and N_c. Specifically, the output of N_c is a color $c_p \in [0,1]^3$, and the output of N_d is a displacement along the vertex normal d_p. To prevent content-altering displacements, we constrain d_p to be in the range $(-0.1, 0.1)$. To obtain our stylized mesh prediction M^S, every point p is displaced by $d_p \cdot \vec{n}_p$ and colored by c_p. Vertex colors propagate over the entire mesh surface using an interpolation-based differentiable renderer [7]. During training we also consider the displacement-only mesh M^S_{displ}, which is the same as M^S without the predicted vertex colors (replaced by gray). Without the use of M^S_{displ} in our final loss formulation (Eq. (5)), the learned geometric style is noisier ($-displ$ ablation in Fig. 5).

3.2. Text-based correspondence

Our neural optimization is guided by the multi-modal embedding space encoded by a pre-trained CLIP [44] model. Given the stylized mesh M^S and the displaced mesh M^S_{displ}, we sample n_θ views around a pre-defined anchor view and render them using a differentiable renderer. For each view, θ, we render two 2D projections of the surface, I_θ^{full} for M^S and I_θ^{displ} for M^S_{displ}. Next, we draw a 2D augmentation $\psi_{\text{global}} \in \Psi_{\text{global}}$ and $\psi_{\text{local}} \in \Psi_{\text{local}}$ (details in Sec. 3.3). We apply ψ_{global}, ψ_{local} to the full view and ψ_{local} to the uncolored view, and embed them into CLIP space. Finally, we average the embeddings across all views:

$$\hat{S}^{\text{full}} = \frac{1}{n_\theta}\sum_\theta E\left(\psi_{\text{global}}\left(I_\theta^{\text{full}}\right)\right) \in \mathbb{R}^{512}, \tag{2}$$

$$\hat{S}^{\text{local}} = \frac{1}{n_\theta}\sum_\theta E\left(\psi_{\text{local}}\left(I_\theta^{\text{full}}\right)\right) \in \mathbb{R}^{512}, \tag{3}$$

$$\hat{S}^{\text{displ}} = \frac{1}{n_\theta}\sum_\theta E\left(\psi_{\text{local}}(I_\theta^{\text{displ}})\right) \in \mathbb{R}^{512}. \tag{4}$$

That is, we consider an augmented representation of our input mesh as the average of its encoding from multiple augmented views. The target t is similarly embedded through CLIP by $\phi_{\text{target}} = E(t) \in \mathbb{R}^{512}$. Our loss is then:

$$\mathcal{L}_{\text{sim}} = \sum_{\hat{S}} \text{sim}\left(\hat{S}, \phi_{\text{target}}\right) \tag{5}$$

where $\hat{S} \in \{\hat{S}^{\text{full}}, \hat{S}^{\text{displ}}, \hat{S}^{\text{local}}\}$ and $\text{sim}(a,b) = \frac{a\cdot b}{|a|\cdot|b|}$ is the cosine similarity between a and b. We repeat the above with new sampled augmentations n_{aug} times for each iteration. We note that the terms using $\hat{S}^{\text{full}}$ and $\hat{S}^{\text{local}}$ update N_s, N_c and N_d while the term using $\hat{S}^{\text{displ}}$ only updates N_s and N_d. The separation into a *geometry-only* loss and *geometry-and-color* loss serves to encourage meaningful changes in geometry ($-displ$ in Fig. 5).

3.3. Viewpoints and Augmentations

Given an input 3D mesh and target text, we first find an *anchor* view. We render the 3D mesh at uniform intervals around a sphere and obtain the CLIP similarity for each view and target text. We select the view with the highest (*i.e.* best) CLIP similarity as the *anchor* view. Often there are multiple high-scoring views around the object, and using any of them as the anchor will produce an effective and meaningful stylization. See Supplementary C for details.

We render multiple views of the object from randomly sampled views using a Gaussian distribution centered around the anchor view (with $\sigma = \pi/4$). We average over the CLIP-embedded views prior to feeding them into our loss, which encourages the network to leverage view consistency. For all our experiments, $n_\theta = 5$ (number of sampled views). We show in Supplementary C that setting n_θ beyond 5 does not meaningfully impact the results.

Figure 6. Our neural texture field stylizes the entire 3D shape.

The 2D augmentations generated using ψ_{global} and ψ_{local} are critical for avoiding degenerate solutions (see Sec. 4.2). ψ_{global} involves a random perspective transformation and ψ_{local} generates both a random perspective and a random crop that is 10% of the original image. Cropping allows the network to focus on localized regions when making fine grained updates to the surface geometry and color. (-*crop* in Fig. 5). Additional details are given in Supplementary D.

4. Experiments

We examine our method across a diverse set of input source meshes and target text prompts. We consider a variety of sources including: COSEG [54], Thingi10K [71], Shapenet [5], Turbo Squid [57], and ModelNet [59]. Our method requires no particular quality constraints or preprocessing of inputs, and the breadth of shapes we stylize in this paper and in our project webpage illustrates its ability to handle low-quality meshes. Meshes used in the main paper and the project webpage contain an average of 79,366 faces, 16% non-manifold edges, 0.2% non-manifold vertices, and 12% boundaries. Our method takes less than 25 minutes to train on a single GPU, and high quality results usually appear in less than 10 minutes.

In Sec. 4.1, we demonstrate the multiple control mechanisms enabled by our method. In Sec. 4.2, we conduct a series of ablations on the key priors in our method. We further explore the synergy between learning color and geometry in tandem. We introduce a user study in Sec. 4.3 where our stylization is compared to a baseline method. In Sec. 4.4, we show that our method can easily generalize to other target modalities beyond text, such as images or 3D shapes. Finally, we discuss limitations in Sec. 4.6.

4.1. Neural Stylization and Controls

Our method generates details with high granularity while still maintaining global semantics and preserving the underlying content. For example in Fig. 2, given a vase mesh and target text 'colorful crochet', the stylized output includes knit patterns with different colors, while preserving the structure of the vase. In Fig. 3, our method demonstrates a global semantic understanding of humans. Different body parts such legs, head and muscles are stylized appropriately in accordance with their semantic role, and these styles are blended seamlessly across the surface to form a cohesive texture. Moreover, our neural style field network generates structured textures which are aligned to sharp curves and features (see bricks in Figs. 1 and 2 and in the project webpage). We show in Fig. 6 and in the project webpage that our method styles the entire mesh in a consistent manner that is part-aware and exhibits natural variation in texture.

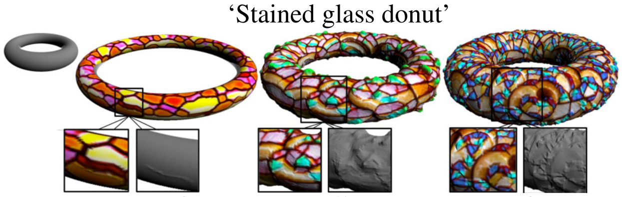


Figure 7. Increasing the range of input frequencies in the positional encoding using increasing SD σ for matrix $\mathbf{B}$ in Eq. (1).

Fine Grained Controls. Our network leverages a positional encoding which enables direct control over the encoding frequency through the σ term of the $\mathbf{B}$ matrix in Eq. (1). In Fig. 7, we show the results of three different frequency values when stylizing a source mesh of a torus with the target text 'stained glass donut'. Increasing the frequency value increases the frequency of style details on the mesh and encourages sharper geometric detail. Our method can also synthesize consistent styles of varying levels of specificity. Fig. 8 displays styles of increasing detail and specificity for two input shapes. Note the retention of the style details from each level of target granularity to the next. Though the primary mode of style control is through the text target, we explore how the network adapts to the source shape geometry. In Fig. 10, the target text prompt is fixed to 'cactus'. We consider different input source spheres with increasing protrusion frequency. Observe that both the frequency and structure of the generated style fits to the pre-existing structure of the input surface. Thus our method can preserve the content of the input mesh without compromising the quality of the stylization.

Figure 8. Increasing the target text prompt granularity for a source mesh of a lamp and iron. Top row targets: (a). 'Lamp', (b). 'Luxo lamp', (c). 'Blue steel luxo lamp', (d). 'Blue steel luxo lamp with corrugated metal. Bottom row targets: (a). 'Clothes iron', (b). 'Clothes iron made of crochet', (c). 'Golden clothes iron made of crochet', (d). 'Shiny golden clothes iron made of crochet'.

Our ability to modify style while preserving the input mesh enables morphing [3] (see Fig. 9). To morph between two stylizations, we apply linear interpolation between the style values of every point on the mesh.

Figure 9. Morphing between two different stylizations (geometry and color). Left: 'wooden chair', right: 'colorful crochet chair'.

4.2. Text2Mesh Priors

Our method incorporates a number of priors that allow us to perform stylization without a pre-trained GAN. We show an ablation for each prior in Fig. 5. Removing the style field network ($-$*net*), and instead directly optimizing the vertex colors and displacements, results in noisy and arbitrary displacements over the surface. In [11] random 2D augmentations are necessary to generate meaningful CLIP-guided drawings. Similarly in our method, removing 2D augmentations results in a stylization completely unrelated to the target text prompt. Without Fourier feature encoding ($-$*FFN*), the generated style loses all fine-grained details. Without the cropping augmentation ($-$*crop*), the output is similarly unable to synthesize fine-grained style details. Removing the *geometry-only* component of L_{sim} (*−displ*) hinders geometric refinement, and the network instead compensates by simulating geometry through shading (see also Fig. 11). Without a meaningful geometric prior (*−3D*) offered by a source mesh, the 2D plane mesh is treated as an image canvas. For each result in Fig. 5, we report the CLIP similarity score, $\text{sim}(\hat{S}^{\text{full}}, \phi_{target})$, as defined in Sec. 3. We obtain the highest score across different ablations, see Fig. 5. Ideally, visual quality and CLIP score would be directly correlated. However, *-3D* obtains a high CLIP similarity, despite having no relation to the target content. This establishes the importance of the geometric prior in steering our method away from potential degenerate solutions.

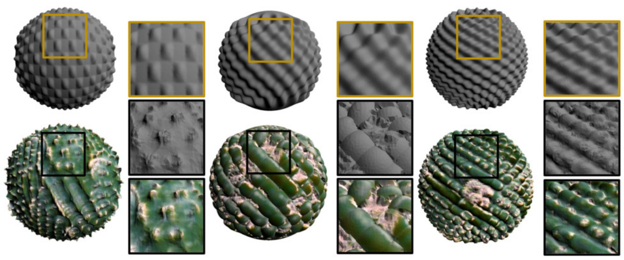


Figure 10. Texturing input source spheres (yellow) with protrusions of increasing frequency and with a fixed target of a 'Cactus'. As can be seen, the final style frequency increases accordingly.

Interplay of Geometry and Color. Our method utilizes the interplay between geometry and color for effective stylization, as shown in Fig. 11. Learning to predict only geometric manipulations produces inferior geometry compared to learning geometry and color together, as the network attempts to simulate shading by generating displacements for self-shadowing. For example in "Batman" in Fig. 3 the bat symbol on the chest is the result of a deep concavity formed through displacements alone. Similarly learning to predict only color results in the network attempting to hallucinate geometric detail through shading, leading to a flat and unrealistic texture that nonetheless is capable of achieving a relatively high CLIP score when projected to 2D. Fig. 11 illustrates this adversarial solution, where the "Color" mode achieves a similar CLIP score as our "Full" method.

4.3. Stylization Fidelity

Our method performs the task of general text-driven stylization of meshes. Given that no approaches exist for this task, we evaluate our method's performance by extending VQGAN-CLIP [9]. This baseline synthesizes color inside a binary 2D mask projected from the 3D source shape (without 3D deformations) guided by CLIP. Further, the baseline is initialized with a rendered view of the 3D source. We conduct a user study to evaluate the perceived quality of the generated outputs, the degree to which they preserve the source content, and how well they match the target style.

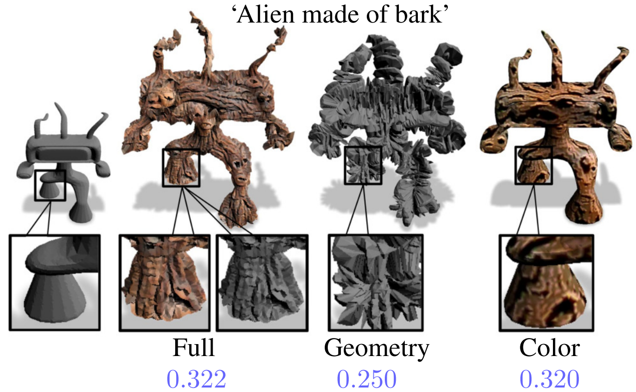


Figure 11. Interplay between geometry and color for stylization. *Full* - our method, *Color* - only color changes, and *Geometry* - only geometric changes. We also display the CLIP similarity.

	(Q1): Overall	(Q2): Content	(Q3): Style
VQGAN	2.83 (±0.39)	3.60 (±0.68)	2.59 (±0.44)
Ours	**3.90** (±0.37)	**4.04** (±0.53)	**3.91** (±0.51)

Table 1. Mean opinion scores (1-5) for Q1-Q3 (see Sec. 4.3), for our method and baseline (control score: 1.16).

We had 57 users evaluate 8 random source meshes and style text prompt combinations. For each combination, we display the target text and the stylized output in pairs. The users are then asked to assign a score (1-5) to three factors: (Q1) "How natural is the output depiction of {*content*} + {*style*}?" (Q2) "How well does the output match the original {*content*}?" (Q3) "How well does the output match the target {*style*}?". We report mean opinion scores (MOS) with standard deviations for each factor averaged across all style outputs for our method and the baseline in Tab. 1. Our method outperforms the VQGAN baseline across all questions, with a difference of 1.07, 0.44, and 1.32 for Q1-Q3, respectively. Though VQGAN is somewhat effective at representing the natural content, it struggles to synthesize these representations with style in a meaningful way. Examples of our baseline outputs are provided in Supplementary E. Visual examples of generated styles and screenshots of the user study are also discussed in Supplementary E.

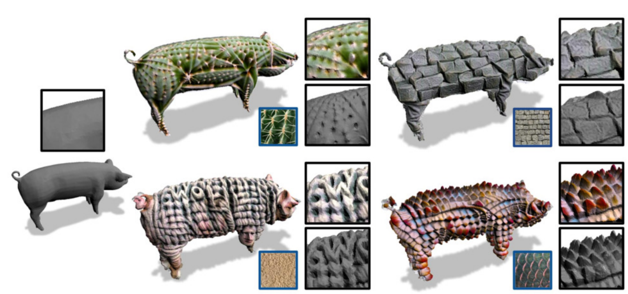


Figure 12. Stylization driven by an image target. Our method can stylize meshes using an image to describe the desired style.

4.4. Beyond Textual Stylization

Beyond text-based stylization, our method can stylize meshes toward different target modalities such as a 2D image or even a 3D object. For a target 2D image I_t, ϕ_{target} in Eq. (5), represents the image-based CLIP embedding of I_t. For a target mesh T, ϕ_{target} is the average embedding in CLIP space of the 2D renderings of T, with equivalent view samples as those for the source mesh. Furthermore, we can combine targets across different modalities by simply summing $\mathcal{L}_{sim}$ over each target. In Fig. 12 we consider a source pig mesh with different image targets. In Fig. 13(a-b), we consider stylization using a target mesh and in Fig. 13(c-d), we combine both a target mesh and target text. Our method successfully adheres to the target style.

(a) (b) Target 1 (c) (d) Target 2

Figure 13. Neural stylization driven by mesh targets. (a) & (c) are styled using Targets 1 & 2, respectively. (b) & (d) are styled with text in addition to the mesh targets: (b) 'a cactus that looks like a cow', (d) 'a mouse that looks like a duck'.

Input No Symmetry Prior Symmetry Prior

Figure 14. Effect of the symmetry prior on a UFO mesh input with text prompt: 'colorful UFO'.

4.5. Incorporating Symmetries

We can make use of shape symmetry priors to enforce style consistency across the axis of symmetry. Such symmetries can be introduced into our model by modifying the input to our positional encoding in Eq. (1). For instance, given a point $p = (x, y, z)$ and a shape with bilateral symmetry across the X-Y plane, one can apply a function prior to the the positional encoding such that $\gamma(x, y, |z|)$. We show the effect of this symmetry prior on a UFO mesh in Fig. 14. This prior is effective even when the triangulation is not perfectly symmetrical. A full investigation into incorporating additional symmetries within positional encoding is an interesting direction for future work.

4.6. Limitations

Our method implicitly assumes a synergy between the input 3D geometry and the target style prompt (see Fig. 15). However, stylizing a 3D mesh (*e.g.*, dragon) towards an unrelated/unnatural prompt (*e.g.*, *stained glass*) may result in a stylization that ignores the geometric prior and effectively erases the source shape content. Therefore, in order

'Stained glass' 'Stained glass dragon' 'Candy' 'Candy Aramdillo'

Figure 15. Geometric content and target style synergy. If the target style is unrelated to the 3D mesh content, the stylization may ignore the 3D content. Results are improved when including the content in the target text prompt.

to preserve the original content when editing towards a mismatched target prompt, we simply include the object category in the text prompt (*e.g.*, *stained glass dragon*) which adds a content preservation constraint into the target.

5. Conclusion

We present a novel framework for *stylizing* input meshes given a target text prompt. Our framework learns to predict colors and local geometric details using a neural stylization network. It can predict structured textures (*e.g.* bricks), without a directional field or mesh parameterization. Traditionally, the texture structure over 3D surfaces has been guided by 3D shape analysis techniques (as in [60]). In this work, the texture structure is driven by rendered images, which is how they are perceived in the real world.

Without relying on a pre-trained GAN network or a 3D dataset, we are able to manipulate a myriad of meshes to adhere to a wide variety of styles. Our system is capable of generating out-of-domain stylized outputs, *e.g.*, a stained glass shoe or a cactus vase (Fig. 2). Our framework uses a pre-trained CLIP [44] model, which has been shown to contain bias [1]. We postulate that our proposed method can be used to visualize, understand, and interpret such model biases in a more direct and transparent way.

As future work, our framework could be used to manipulate 3D *content* as well. Instead of modifying a given input mesh while preserving its topology, one could learn to generate meshes from scratch driven by a text prompt. Moreover, our NSF is tailored to a single 3D mesh. It may be possible to train a network to stylize a collection of meshes towards a target style in a feed-forward manner.

6. Acknowledgements

We thank Greg Shakhnarovich, Haochen Wang, Hsueh-Ti Derek Liu and Xiaodan Du for their comments. We thank Vincent LaGrassa for help with figures. Finally, we thank the rest of the 3DL lab for their invaluable feedback and support.

References

[1] Sandhini Agarwal, Gretchen Krueger, Jack Clark, Alec Radford, Jong Wook Kim, and Miles Brundage. Evaluating

clip: towards characterization of broader capabilities and downstream implications. *arXiv preprint arXiv:2108.02818*, 2021.

[2] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. *Real-Time Rendering, Fourth Edition*. A. K. Peters, Ltd., USA, 4th edition, 2018.

[3] Marc Alexa. Recent advances in mesh morphing. In *Computer graphics forum*, volume 21, pages 173–198. Wiley Online Library, 2002.

[4] Xu Cao, Weimin Wang, Katashi Nagao, and Ryosuke Nakamura. Psnet: A style transfer network for point cloud stylization on geometry and color. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3337–3345, 2020.

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

[6] Hila Chefer, Sagie Benaim, Roni Paiss, and Lior Wolf. Image-based clip-guided essence transfer. *arXiv preprint arXiv:2110.12427*, 2021.

[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*, 32:9609–9619, 2019.

[8] Zhiqin Chen, Vladimir G Kim, Matthew Fisher, Noam Aigerman, Hao Zhang, and Siddhartha Chaudhuri. Decor-gan: 3d shape detailization by conditional refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15740–15749, 2021.

[9] Katherine Crowson. Notebook to generate images from text phrases with vqgan and clip, 2021. https://github.com/justinjohn0306/VQGAN-CLIP.

[10] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2020.

[11] Kevin Frans, Lisa B. Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *CoRR*, abs/2106.14843, 2021.

[12] Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *arXiv preprint arXiv:2108.00946*, 2021.

[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] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[15] Mark Gillespie, Boris Springborn, and Keenan Crane. Discrete conformal equivalence of polyhedral surfaces. *ACM Transactions on Graphics (TOG)*, 40(4):1–20, 2021.

[16] Fangzhou Han, Shuquan Ye, Mingming He, Menglei Chai, and Jing Liao. Exemplar-based 3d portrait stylization. *arXiv preprint arXiv:2104.14559*, 2021.

[17] Rana Hanocka, Noa Fish, Zhenhua Wang, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Alignet: partial-shape agnostic alignment via unsupervised learning. *ACM Transactions on Graphics (TOG)*, 38(1):1, 2018.

[18] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[19] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh: A self-prior for deformable meshes. *arXiv preprint arXiv:2005.11084*, 2020.

[20] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Deep geometric texture synthesis. *ACM Transactions on Graphics (TOG)*, 39(4):108–1, 2020.

[21] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 2001. Association for Computing Machinery.

[22] Ruizhen Hu, Wenchao Li, Oliver Van Kaick, Hui Huang, Melinos Averkiou, Daniel Cohen-Or, and Hao Zhang. Co-locating style-defining elements on 3d shapes. *ACM Transactions on Graphics (TOG)*, 36(3):1–15, 2017.

[23] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy.*, pages 1510–1519, 2017.

[24] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

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

[26] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410, 2019.

[27] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8110–8119, 2020.

[28] 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.

[29] Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. Stripe patterns on surfaces. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015.

[30] Nicholas Kolkin, Jason Salavon, and Gregory Shakhnarovich. Style transfer by relaxed optimal transport and self-similarity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10051–10060, 2019.

[31] 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, 2021.

[32] Honghua Li, Hao Zhang, Yanzhen Wang, Junjie Cao, Ariel Shamir, and Daniel Cohen-Or. Curve style analysis in a set of shapes. In *Computer Graphics Forum*, volume 32, pages 77–88. Wiley Online Library, 2013.

[33] Minchen Li, Danny M. Kaufman, Vladimir G. Kim, Justin Solomon, and Alla Sheffer. Optcuts: Joint optimization of surface cuts and parameterization. *ACM Transactions on Graphics*, 37(6), 2018.

[34] Hsueh-Ti Derek Liu and Alec Jacobson. Cubic stylization. *arXiv preprint arXiv:1910.02926*, 2019.

[35] Hsueh-Ti Derek Liu, Vladimir G Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. Neural subdivision. *ACM Transactions on Graphics (TOG)*, 39(4):124–1, 2020.

[36] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. Paparazzi: surface editing by way of multi-view image processing. *ACM Trans. Graph.*, 37(6):221–1, 2018.

[37] Zhaoliang Lun, Evangelos Kalogerakis, and Alla Sheffer. Elements of style: learning perceptual shape style similarity. *ACM Transactions on graphics (TOG)*, 34(4):1–14, 2015.

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

[39] Gal Metzer, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Self-sampling for neural point cloud consolidation. *ACM Transactions on Graphics (TOG)*, 40(5):1–14, 2021.

[40] 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.

[41] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4531–4540, 2019.

[42] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020.

[43] Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. Styleclip: Text-driven manipulation of stylegan imagery, 2021.

[44] 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. *arXiv preprint arXiv:2103.00020*, 2021.

[45] Alec Radford, Ilya Sutskever, Jong Wook Kim, Gretchen Krueger, and Sandhini Agarwal. Clip: Connecting text and images, 2021.

[46] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2019.

[47] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps, 2021.

[48] 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.

[49] Mattia Segu, Margarita Grinvald, Roland Siegwart, and Federico Tombari. 3dsnet: Unsupervised shape-to-shape 3d style transfer. *arXiv preprint arXiv:2011.13388*, 2020.

[50] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4570–4580, 2019.

[51] Nicholas Sharp. *Intrinsic Triangulations in Geometry Processing*. PhD thesis, Carnegie Mellon University, August 2021.

[52] Nicholas Sharp and Keenan Crane. Variational surface cutting. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

[53] Assaf Shocher, Shai Bagon, Phillip Isola, and Michal Irani. Ingan: Capturing and retargeting the "dna" of a natural image. In *The IEEE International Conference on Computer Vision (ICCV)*, 2019.

[54] 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.

[55] Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. Bounded-distortion piecewise mesh parameterization. In *IEEE Visualization, 2002. VIS 2002.*, pages 355–362. IEEE, 2002.

[56] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *CoRR*, abs/2006.10739, 2020.

[57] TurboSquid. Turbosquid 3d model repository, 2021. https://www.turbosquid.com/.

[58] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[59] 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.

[60] Kai Xu, Daniel Cohen-Or, Tao Ju, Ligang Liu, Hao Zhang, Shizhe Zhou, and Yueshan Xiong. Feature-aligned shape texturing. *ACM Trans. Graph.*, 28(5):1–7, dec 2009.

[61] Kai Xu, Honghua Li, Hao Zhang, Daniel Cohen-Or, Yueshan Xiong, and Zhi-Quan Cheng. Style-content separation by anisotropic part scales. In *ACM SIGGRAPH Asia 2010*

Papers, SIGGRAPH ASIA '10, New York, NY, USA, 2010. Association for Computing Machinery.

[62] Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. Geometry processing with neural fields. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2021.

[63] Kangxue Yin, Zhiqin Chen, Hui Huang, Daniel Cohen-Or, and Hao Zhang. Logan: Unpaired shape transform in latent overcomplete space. *ACM Transactions on Graphics (TOG)*, 38(6):1–13, 2019.

[64] Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 3dstylenet: Creating 3d shapes with geometric and texture style variations. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2021.

[65] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4578–4587, 2021.

[66] Fenggen Yu, Yan Zhang*, Kai Xu, Ali Mahdavi-Amiri, and Hao Zhang. Semi-supervised co-analysis of 3d shape styles from projected lines. *ACM Transactions on Graphics (TOG)*, 37(2):1–17, 2018.

[67] Mehmet Ersin Yumer, Siddhartha Chaudhuri, Jessica K Hodgins, and Levent Burak Kara. Semantic shape editing using deformation handles. *ACM Transactions on Graphics (TOG)*, 34(4):1–12, 2015.

[68] M Ersin Yumer and Niloy J Mitra. Learning semantic deformation flows with 3d convolutional networks. In *European Conference on Computer Vision*, pages 294–311. Springer, 2016.

[69] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[70] Gang Zhao and Jeff Huang. Deepsim: deep learning code functional similarity. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 141–151, 2018.

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