Pacific Graphics (2023)
R. Chaine, Z. Deng, and M. H. Kim (Editors)

Short Paper

A Style Transfer Network of Local Geometry for 3D Mesh Stylization

Hongyuan Kang[1], Xiao Dong[3], Xufei Guo[1], Juan Cao[2] and Zhonggui Chen[1†]

[1]School of Informatics, Xiamen University, China
[2]School of Mathematical Sciences, Xiamen University, China
[3]Guangdong Provincial Key Laboratory IRADS, BNU-HKBU United International College, China

Abstract

Style transfer of images develops rapidly, however, only a few studies focus on geometric style transfer on 3D models. In this paper, we propose a style learning network to synthesize local geometric textures with similar styles on source mesh, driven by specific mesh or image features. Our network modifies a source mesh by predicting the displacement of vertices along the normal direction to generate geometric details. To constrain the style of the source mesh to be consistent with a specific style mesh, we define a style loss on 2D projected images of two meshes based on a differentiable renderer. We extract a set of global and local features from multiple views of 3D models via a pre-trained VGG network, driving the deformation of the source mesh based on the style loss. Our network is flexible in style learning as it can extract features from meshes and images to guide geometric deformation. Experiments verify the robustness of the proposed network and show the outperforming results of transferring multiple styles to the source mesh. We also conduct experiments to analyze the effectiveness of network design.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Shape analysis;***

1. Introduction

Visual style transfer is a long-standing objective in the field of computer vision. There has been a lot of research on image style transfer [GEB16, JAFF16, IZZE17, ZPIE17, CCK*18, CUYH20, KLA19], but style editing and learning on 3D data remains to be deeply explored due to the irregular and unordered representation. The core of 3D mesh style transfer is to edit the mesh model to conform to a desired style while maintaining the underlying content. Usually, we consider content as the global shape and topology prescribed by a mesh, and style as the object's fine-grained local geometric details.

With the rapid development of style learning in the image field, some works studying the style transfer on mesh models are proposed. There are some traditional methods that optimize the deformation energy to generate a specific style for the mesh, such as cubic stylization [LJ19, LJ21, KFDA21] and legolization [LYH*15]. These methods treat shapes as geometric styles, and only one shape style can be learned. In recent years, some style learning networks have been proposed [YAK*20, HHGCO20] to transfer surface details to the mesh from the feature of other mesh. These studies lead to more options for mesh editing and stylization, but they are limited to geometric style learning between meshes.

Mesh stylization is an important mesh editing and creation technique. In many cases, the model creator hopes to edit the current model to show some existing style that comes from a model or from a texture image. This mesh editing method provides creators with powerful applicability and functionality. For the above purpose, we need a unified style learning method that can transfer styles from both meshes and images to a target mesh.

We design a deep learning framework for synthesizing local geometric structures, which can generate high-quality 3D mesh models with rich local geometric details. The style source can be a specified mesh or an image with a certain style. Taking style transfer between meshes as an example, our network first deforms the input mesh by predicting the displacement of mesh vertices. We then compare the current style of the deformed mesh with the style mesh. To evaluate the style of a mesh, we adopt a differential renderer to project the mesh to a set of 2D images from multiple views, and then extract the latent features of images with a pre-trained feature encoder. Our network iteratively optimizes the predicted displacement of the input mesh to obtain the final model with a specific style under the guide of style loss. Our specific contributions are summarized as follows:

- We propose a deep style transfer network that transfers the geometric texture of a style mesh to an input mesh. Our network defines a style loss on images generated by a differentiable ren-

[†] Zhonggui Chen is the corresponding author.

DOI: 10.2312/pg.20231272

derer and guides the deformation of the input mesh by predicting the displacement of the mesh vertices.

- Our stylization network is more generic than the existing methods as it can learn multiple styles from both 3D meshes and 2D images. We prove the network's effectiveness for learning surface texture styles and demonstrate the results of multiple geometric styles.

2. Related Work

3D mesh stylization is a popular research topic, and current methods include traditional methods based on energy constraints and deep learning methods that use networks to learn deformations. Based on different understandings of geometric styles, some works focus on changing the shape and size of the model, and some works concentrate on editing the model's surface texture. We here review the related methods according to different style types.

2.1. Geometric stylization on shapes

Several traditional methods have been proposed for stylizing 3D models. Xu et al. [XLZ*10] introduced a style-content separation method that analyzes the part correspondence of 3D objects, treating anisotropic part scales as shape styles. Liu et al. [LJ19] presented a cubic stylization algorithm that transforms objects into a cubic style by aligning rotated vertex normals with coordinate axes using an As-Rigid-As-Possible (ARAP) energy optimization. Kohlbrenner et al. [KFDA21] generalized the possibilities of cubic stylization by choosing a set of preferred normals from Gauss sphere and their optimization algorithm, similar to ARAP, maintains a constant linear system in the global optimization. Liu et al. [KFDA21] proposed the spherical shape analogies method that is also based on optimizing surface normals. Specifically, they optimize a set of rotations that align the normals of the input mesh with the desired normals. Furthermore, Luo et al. [LYH*15] devised a Legolization method that employs force-based analysis to estimate the physical stability of a given sculpture and automatically generates a LEGO brick layout from a 3D model.

In recent years, several network-based stylization methods have been proposed. Inspired by traditional cage-based methods [JMD*07, LLCO08], Wang et al. [YAK*20] proposed a neural network with cage-prediction and deformation-prediction models to warp source shape to target shape. 3DStyleNet [YGS*21] predicts a part-aware affine transformation that naturally warps the source shape to imitate the geometric style of the target, while a differentiable renderer facilitates the transfer of texture style.Huang et al. [HLZ*22] proposed a network to learn 3D shape style based on the decomposition of shape style representation from the latent space.

2.2. Geometry stylization on textures

Some researchers focused on transferring surface textures rather than shapes to the target 3D mesh. In earlier work, mesh editing algorithms were based on Laplacian coordinates [SCOL*04, BH11] could transfer texture or coating style to the target mesh. In recent years, many research works have focused on transferring specified styles to target meshes. The source of style can be texture images, meshes, or even text prompts. We introduce related stylization methods according to different style sources in the following.

In terms of image-to-mesh style learning, some work [YHBZ01, Tur01] proposed to synthesize a texture directly on the surface of a model, including color, transparency, and vertex displacements. Wang et al. [WSH*16] proposed a solution for transporting texture images to 3D models, by factoring out geometric and perspective distortions from illumination effects and compensating for both. Oechsle et al. [OMN*19] proposed Texture Fields for texture reconstruction of 3D objects based on regressing a continuous 3D function parameterized with a neural network. Liu et al. [LTJ18] developed a network that allows users to edit an input 3D surface by simply selecting an image processing filter that can back-propagate the changes in the image domain to the mesh vertex positions.

In terms of mesh-to-mesh texture transferring, Berkiten et al. [BHS*17] proposed a method that calculates a displacement map to transfer details from high-quality models to simple shapes without textures. Hertz et al. [HHGCO20] introduced a deep network that learns geometric texture from local triangular patches and generates vertex displacements to synthesize local geometries.

Text-to-mesh stylization has been motivated by the increasing utilization of cross-modal supervision in the CLIP model [RKH*21]. Michel et al. [MBOL*22] introduced Text2Mesh, a method that predicts a 3D mesh with color and geometric details that conform to a given text prompt. To achieve photorealistic appearances for meshes, Chen et al. [CCL*22] proposed a disentanglement approach that separates the appearance style into reflectance and scene lighting. These components are jointly optimized using supervision from the CLIP loss.

In our work, we shift the focus of stylization towards surface texture details and explore diverse sources of style to produce a variety of stylized outcomes. Given a styled mesh or texture image, our network employs a progressive refinement approach to adapt the input mesh to the desired style. This makes our method more versatile compared to prior works in the field.

3. Method

3.1. Network architecture

We illustrate the architecture of the style transfer network in Fig. 1. The network takes a source mesh and a style mesh as input. The surface texture of the style mesh serves as a representation of the desired geometric style that we aim to transfer to the source mesh.

As shown in Fig. 1, to make the smooth source mesh learn the bumpy surface texture of the style mesh, we calculate the normal direction for each vertex and feed all vertices into a position-encoded network. The network predicts the displacement of each vertex along its corresponding normal direction, resulting in a synthesized mesh with new vertex positions. It is worth noting that the connectivity of the mesh remains unchanged throughout the training process. We use a differentiable renderer to project generated mesh and style mesh to get 2D images from different viewpoints. In addition, we partition the global image into smaller local images to capture geometric styles at different scales. To ensure faithful

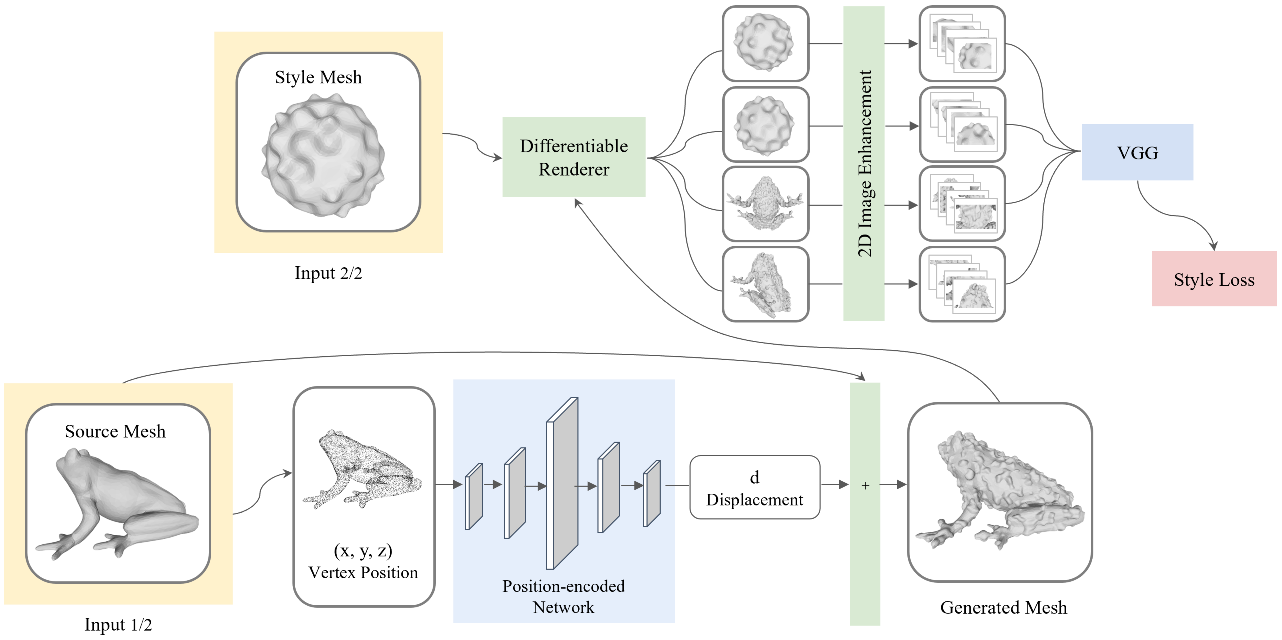

Figure 1: *The overall architecture of the neural style transfer network for 3D meshes.*

style transfer, we compare the latent style features extracted from a pre-trained VGG [SZ14] of the generated mesh and style mesh. The style loss considers both global and local features of the model. Finally, Our network optimizes the surface texture of the generated mesh by minimizing the image style similarity loss. The training process iteratively updates the vertices of the generated mesh along the normal direction until the desired style is achieved.

3.2. Position-encoded network

We denote the input source mesh as C, which consists of n_v vertices represented by $V \in \mathbb{R}^{n_v \times 3}$ and n_t faces represented by $T \in \{1,\dots,n_v\}^{n_t \times 3}$. The source mesh remains fixed during the training process. We use S to represent the style mesh and M to represent the generated mesh.

The position-encoded network generates a style attribute for each vertex, which is expressed as the displacement of that vertex on the mesh surface. This displacement occurs along the normal direction and is constrained within a small range. This design tightly couples the style field to the source mesh, resulting in subtle geometric modifications on the surface. Adjusting the vertex positions of the mesh surface generates texture details that represent a specific geometric style. In our approach (Fig. 1), we take the vertex coordinates as input and use a lightweight MLP network to predict the displacement along the vertex normal. This network only modifies the vertex positions while keeping the mesh topology fixed. As a result, the shape of the generated mesh remains largely unchanged, but the surface texture becomes rich and stylized.

We attempted to feed the vertex coordinates to the position-encoded network. However, we noticed that the MLP network struggled to learn high-frequency texture details directly from this low-dimensional input. To address this issue and generate high-frequency details, we utilized a position encoding technique [TSM*20]. This technique involves feeding the network with Fourier features to enable learning of high-frequency functions in low-dimensional domains. Instead of directly providing the 3D vertex coordinates to the MLP, we utilized a Fourier feature mapping to featurize the input coordinates. For each vertex $v \in V$, its positional encoding $\gamma(v)$ is given as follows:

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

where B is a random Gaussian matrix with dimension $n_v \times 3$, and each entry follows a normal distribution $\mathrm{N}\left(0,\sigma^2\right)$. We use σ as a hyperparameter to control the frequency function of learning styles. In experiments we compare the effect of different σ values on the network's ability to learn high-frequency features.

After applying Fourier feature mapping, the position-encoded features are transformed from three dimensions (vertex coordinates) to a h_f dimensional space, here $h_f = 256$. The resulting high-dimensional feature $\gamma(v)$ serves as the input to the first layer in MLP. In our MLP network, we use 8 fully connected layers with different feature channels. For each vertex v_i, the last layer outputs a single value, which represents the displacement d_i of the vertex along its normal direction $\hat{n}_i$. We calculate the new position of v_i by formula $d \cdot \hat{n}_i$. Since the connection between vertices of the mesh remains unchanged, the network gives fine-grained adjustments to the surface geometry of the generated mesh M.

3.3. Style feature learning based on differentiable rendering

We evaluate the style similarity between styled mesh and generated mesh by comparing their projected 2D images. We then encode tex-

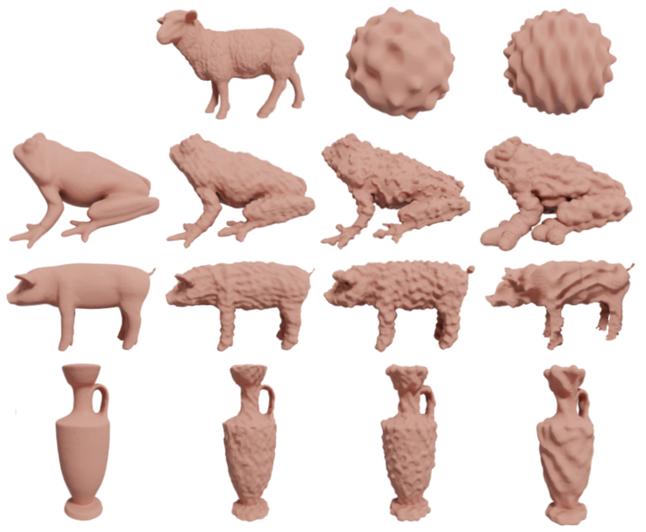

Figure 2: *Different local geometric stylized results. The first row shows the style meshes, the first column shows the source meshes and the rest are the stylized meshes generated by our network.*

ture images to get latent style features with a pre-trained encoder. We will introduce each stage in detail.

3.3.1. Rendering and image feature augmentation

To render the 3D model, we begin by specifying an initial view and utilize a differentiable renderer to project the model into a 2D image. We render both the source mesh and style mesh at random views based on a Gaussian distribution with a standard deviation of $\sigma = \pi/4$, centered around the initial view, and then evaluate the similarity of all the views between the two models. For a rendering view D_θ, we first obtain a global projection $\omega_{global} \in \Omega_{global}$ of the model. Since our network focuses on learning local geometric features, we augment the images by randomly cropping the global image to obtain a local projection $\omega_{local} \in \Omega_{local}$. By simultaneously supervising the learning of both global and local views, our network can more precisely guide the refinement of the local geometry of the 3D model. Specifically, given the view D_θ, we denote the global and local views as $\hat{D}_{\theta 1}$ and $\hat{D}_{\theta 2}$, respectively, i.e.,

$$\hat{D}_{\theta 1} = \omega_{global}(D_\theta), \hat{D}_{\theta 2} = \omega_{local}(D_\theta), \tag{2}$$

where ω_{global} involves a random perspective transformation on the model. ω_{local} generates a random perspective and a random crop, yielding a local view representing 10% of the global view.

In our network, we render the model from multiple views $D_\theta, \theta \in \Theta$, where Θ is the set of all rendering views. The augmentation of global and local views provides rich texture images for both style mesh and generated mesh. This allows for improved supervision and better control over the generation of local geometry in the generated mesh when predicting vertex displacement.

3.3.2. Style feature extraction

The global and local texture images of style mesh and generated mesh are encoded into the implicit space $F(\cdot) \in \mathbb{R}^{hw \times n}$ using a feature extractor. Here, h and w refer to the height and width of the rendered images, and n is the total number of feature channels in the implicit space. To compute the style loss, we select a set of style layers $l_s = \left\{l_s^1, l_s^2, \ldots, l_s^q\right\}$ from the feature encoder. In the subsequent definition, we omit the superscript and subscript and use l to represent a feature layer.

In the study of image-to-image style transfer, researchers commonly employ pre-trained models such as VGG to extract features from images [GEB*17, LLKY19]. Our approach follows a similar strategy and utilizes VGG-19 as the image feature extractor to guide the style learning process. To capture the style of images at different resolutions, we choose a combination of some convolution layers as our style layer l_s. In Sec. 4.3, we give experiments to investigate the impact of selecting different style feature layers on the quality of the final style transfer results.

The Gram matrix is a commonly used representation of the style correlation between feature channels. Given a feature layer l with n channels, let $F^l(\cdot)$ represent the implicit features produced by that layer. We compute the Gram matrix $G\left(F^l(\cdot)\right) \in \mathbb{R}^{n \times n}$ to measure the spatial correlation of different feature channels in that layer. Specifically, $G\left(F^l(\cdot)\right) = (F^l(\cdot))^\top (F^l(\cdot))$. The ij-th element corresponds to the inner product of the i-th column vector and the j-th column vector of the feature layer $F^l(\cdot)$.

The global style features extracted by the pre-trained encoder for the style mesh S and the generated mesh M under the global views can be expressed respectively as follows:

$$F_1(S) = \sum_{\theta \in \Theta} \sum_{l \in l_s} G\left(F^l(\hat{D}_{\theta 1}(S))\right), \tag{3}$$

$$F_1(M) = \sum_{\theta \in \Theta} \sum_{l \in l_s} G\left(F^l(\hat{D}_{\theta 1}(M))\right). \tag{4}$$

Similarly, we can extract the local style features of the style mesh S and the generated mesh M under the local views as follows:

$$F_2(S) = \sum_{\theta \in \Theta} \sum_{l \in l_s} G\left(F^l(\hat{D}_{\theta 2}(S))\right), \tag{5}$$

$$F_2(M) = \sum_{\theta \in \Theta} \sum_{l \in l_s} G\left(F^l(\hat{D}_{\theta 2}(S))\right). \tag{6}$$

3.3.3. Style loss function

The style loss function evaluates the dissimilarity of geometric style between the style mesh S and the generated mesh M, considering both global and local features:

$$Loss(S,M) = \mathcal{L}_1(S,M) + \lambda * \mathcal{L}_2(S,M), \tag{7}$$

where global style loss is defined as follows:

$$\mathcal{L}_1\,(S,M) = \|F_1(S) - F_1(M)\|_2^2, \tag{8}$$

The local style loss is defined similarly. With the guidance of the style loss computed on rendered images, the surface of the predicted mesh M gradually deforms, resulting in a geometric texture similar to the styled mesh S.

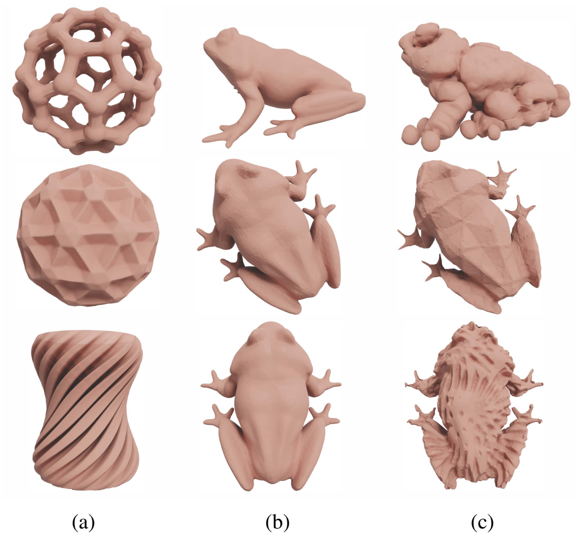

Figure 3: *Multiple geometric style transfer results for the frog model. (a) Style mesh; (b) source mesh; (c) style transfer results.*

4. Experiments

We discuss parameter settings, comparison experiments and ablation study to show the effectiveness of our style transfer network.

4.1. Experimental setting

To facilitate style transfer learning, we gather a diverse collection of triangular mesh models, encompassing various categories such as animals, humans, artifacts and more. Prior to training, we perform preprocessing on each mesh model by relocating it to the geometric center and normalizing its scale to fit within a unit sphere.

Training settings For the position-encoded network, we encode the 3D coordinates of each vertex using a Fourier feature encoding, resulting in a 256-dimensional feature vector $\gamma(v)$. We set the standard deviation σ of the Gaussian matrix B to 5.0. The high-dimensional feature $\gamma(v)$ is processed by an MLP with 8 fully connected layers to determine the new position of vertex v. To prevent drifting vertices, we limit the displacement d of the vertex to the range $(-0.1, 0.1)$. During differential renderer, we apply random perspective transformations to both global (ω_{global}) and local (ω_{local}) projections. In specific, we sample 5 global views for each mesh and randomly crop each global view to obtain 4 local views to augment texture images. The global image is rendered at a resolution of $h \times w = 250 \times 250$, while the local image has a resolution of 25×25. Before encoding the images with VGG, we apply mean and standard deviation normalization to each channel. In practice, we use the values $(0.481, 0.458, 0.408)$ for mean normalization and $(0.268, 0.261, 0.276)$ for standard deviation normalization. In the loss function, the weighting between the global and local feature loss λ is set to 0.1 in experiments.

We utilize the NVIDIA Kaolin library [FTSL*22] in PyTorch that offers differentiable rendering capabilities. For training, we use the Adam optimizer [KB14] with an initial learning rate 0.0005, and decay it by a factor of 0.9 every 100 iterations. The training process is conducted on a single Nvidia GeForce RTX 3090 GPU and takes around 25 minutes to complete for 1000 iterations.

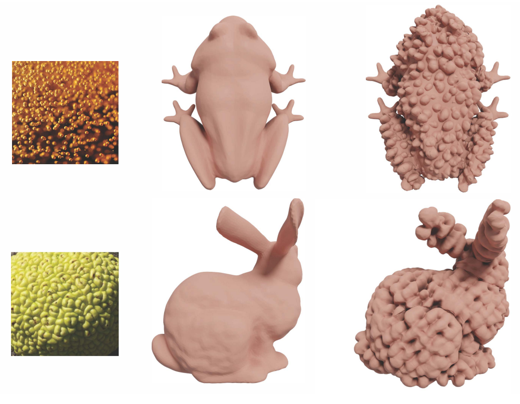

Figure 4: *Stylized results of transferring 2D texture image to 3D meshes. (a) Style image; (b) source mesh; (c) style transfer results.*

4.2. Qualitative results

4.2.1. Style transfer from mesh

We present the style transfer results on various mesh models, including animals and artifacts. Fig. 2 shows the style transfer results using the sheep model, small wave sphere model, and large wave sphere model as style meshes. For the sheep model, the overall style is represented by the undulating features on the body as its body takes up a large proportion on the rendered images. The sheep model exhibits a small and dense undulation shape, and the small wave sphere model has a relatively larger and sparser raised structure on its surface. The large wave model displays a more prominent raised structure. The first row represents the style meshes, and the first column represents the source meshes. The stylized meshes exhibit different local geometries, demonstrating a strong ability of our network to learn various geometric styles.

To demonstrate the style learning ability of our network, we show multiple geometric styles transferred to a frog model in Fig. 3. For buckyball style, the stylized frog exhibits a more rounded shape overall while preserving the local spherical features such as the eyes, paws, and fingers. For the rail sphere style, the back of the frog model shows angular edges reminiscent of the surface of the rail sphere model. For the stripe style, the stylized frog effectively captures the striped structure, with varying degrees of stripes on the head, back, and limbs, displaying a distinct layering effect. These results prove the successful transfer of different geometric styles to the frog model, resulting in visually appealing outcomes.

4.2.2. Style transfer from image

Our network is capable of learning styles from images. Since the style is provided by a texture image, we only need to render the source mesh. We encode given style image using the VGG network to obtain its latent features, which are then compared with

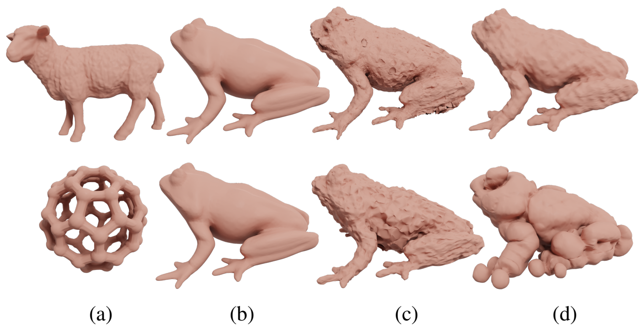

Figure 5: *Comparison of our network with Hertz's geometric texture synthesis network [HHGCO20]. (a) Style mesh; (b) source mesh; (c) results of Hertz's method; (d) results of our network.*

the style of rendered images of the source mesh to calculate their style loss. We design an image-to-mesh style learning module that can be plugged into the network, allowing it to accept various style sources and increase its versatility. The stylized results using given texture images are shown in Fig. 4. The first image represents a style with multiple bubbles of different sizes. As a result, the stylized frog model exhibits bulbous shapes of varying sizes, resembling the bubble style depicted in the image. The second row shows a style image depicting a peeling surface with blocks of different sizes and separations. Our network considers both global and local features in style learning, thus the surface of the bunny shows similar compartmentalized appearance.

4.2.3. Comparison experiments

Our network specializes in local geometric stylization. We compare with two most relevant stylization works: Hertz's method [HHGCO20] and Text2Mesh [MBOL*22]. Hertz's method is a mesh-to-mesh geometry synthesis network that uses a generative adversarial network (GAN) to synthesize local mesh patches with target style. Text2Mesh is dedicated to color and geometric texture generation of meshes driven by CLIP-based text. While those methods focus on transferring geometric styles from style sources to meshes, our network expands the range of style transfer applications by extracting styles from both mesh and image sources.

In Fig. 5, we give the comparison of our network and Hertz's method on sheep and buckyball style. Our method successfully captures the undulating shape of the sheep style, while Hertz's method fails. Our method also generates geometry that is consistent with the spherical effect of the buckyball style, resulting in a more aesthetically pleasing output. Hertz's network precomputes additional style meshes of multiple resolutions given the buckyball mesh, and selects certain scales for style learning. We observed a loss of surface detail with multi-scale style meshes, so that Hertz's method did not learn the expected texture fluctuations on frog surface.

In Fig. 6, we present a comparison between Text2Mesh and our network using the same image as the style. While Text2Mesh primarily relies on text descriptions, the CLIP model's correlation alignment between images and corresponding text descriptors allows for the use of images as style guidance as well. In the figure, we selected texture images featuring diagonal stripes as the

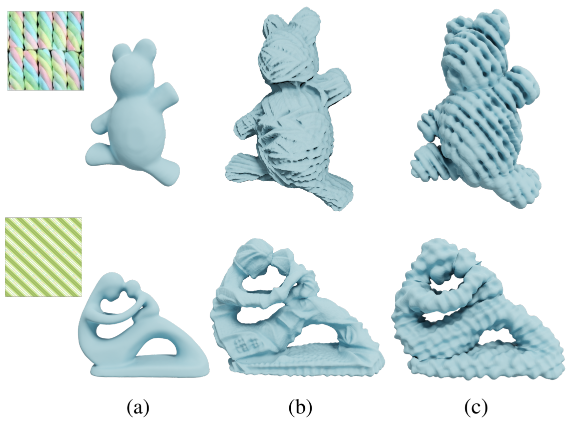

Figure 6: *Comparison of our network with Text2Mesh [MBOL*22]. (a) Style image and source mesh; (b) results of Text2Mesh; (c) results of our network.*

style guidance. We can observe that our approach captures the styles of the images with greater clarity and accuracy compared to Text2Mesh. Since Text2Mesh relies on text descriptions for specifying styles, it can be challenging for users to accurately describe specific geometric styles, limiting its application. In contrast, our network ensures faithful style transfer based on style meshes or images, making it more versatile and user-friendly.

4.3. Parameter settings

Parameters in position encoding A random Gaussian matrix B is used to map vertex coordinates $v \in \mathbb{R}^3$ to a high-dimensional Fourier feature space in position-encoded network. Each entry in B is sampled from a normal distribution $N\left(0,\sigma^2\right)$, where σ is a hyperparameter that controls the frequency function for position encoding. We conducted experiments to compare the effect of style transfer with different σ values, as shown in Fig 7. Increasing σ resulted in more pronounced geometric styles, bumpy features, and higher frequency details, while smaller values of σ had minimal effect and fewer details. Based on empirical observations, we set σ to 5.0 in implementation, as it produces a distinct stylistic output without excessive focus on details.

Impact of different feature layers To extract the style of texture images generated by differentiable renderer, we adopt a pretrained VGG-19 as the feature extractor. VGG-19 model consists of 5 convolutional blocks with 64, 128, 256, 512, and 512 feature channels respectively. Each convolutional block contains multiple layers, and there are 16 convolutional layers in total. The choice of style feature representation layers plays a critical role. The first two convolutional blocks have feature channels less than 256, which we consider as low-level features. Therefore, we assign the convolutional layers [1, 2, 3, 4] to represent the low-level style of the model. We select layers [10, 12, 13, 14] with 512 feature channels as the high-level style. In experiments we found that using either low-level features or high-level features alone did not adequately capture the local geometric features of meshes. As depicted in Fig. 8(b), when only lower layers are used, the shape of

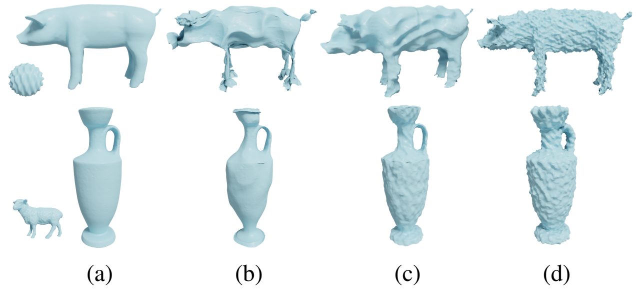

Figure 7: *Style transfer results of the Gaussian matrix B with different values of σ in the position-encoded network. (a) source mesh; (b) σ = 1.0; (c) σ = 5.0; (d) σ = 10.0.*

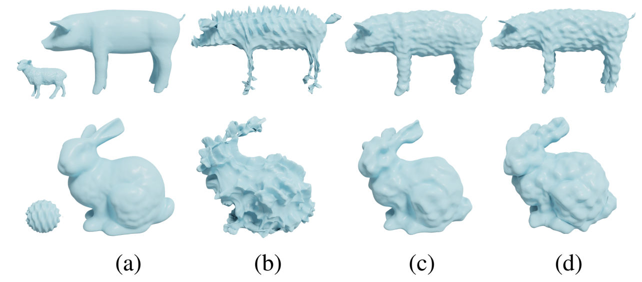

Figure 8: *The results obtained by selecting different style feature layers. (a) Source mesh; (b) style layers: [1, 2, 3, 4]; (c) style layers: [1, 3, 5, 9, 13] and (d) style layers: [10, 12, 13, 14].*

the source mesh is significantly distorted, and the style has a greater impact on the overall shape rather than surface textures. In contrast, when only higher layers are used, as shown in Fig. 8(d), the mesh is able to learn the geometric style, but the undulation of its surface texture is not clearly visible, resulting in indistinct geometric textures. To better represent the geometric style, we extract one layer from each of the five convolutional blocks with different resolutions to combine low-dimensional and high-dimensional features. In implementation, we select layers[1, 3, 5, 9, 13] for style representation. Fig. 8(c) demonstrates the learned style. Our network successfully preserves the shape structure of the original mesh and synthesizes local geometry on the surface, resulting in smooth and natural textures that are indistinguishable from the style mesh.

4.4. User study

We conducted a user study to evaluate the visual quality of the stylized meshes generated by our network. The study consisted of 16 sets of source meshes, style meshes (or style images), and stylized meshes. Participants were asked to rate how well the style transfer results matched the target style on a scale of 1-5, where 1 represents "not a match" and 5 represents "perfect". A total of 43 graduate students with a background in computer science participated in the study. In Fig. 9, we list the "source model-style model" pairs on the x-axis, and the average rating on the y-axis. All stylized meshes received a score higher than 3 and the majority of models achieved an average score higher than 4, indicating that our network successfully generates the specific geometric styles on the source meshes that are similar to those of the style meshes.

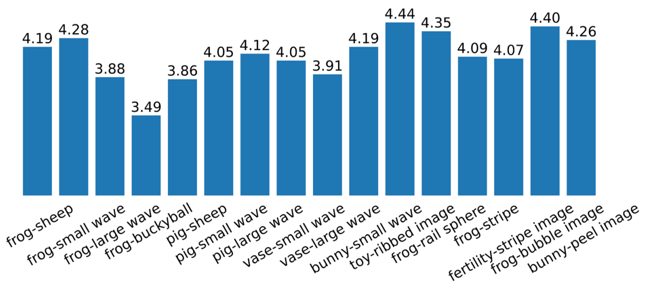

Figure 9: *Average scores (1-5) for stylized meshes in user study. For each stylized mesh, we use "model-style" to denote the source mesh and style mesh (image) respectively.*

5. Conclusion

We propose a neural style transfer network that leverages a differentiable renderer to transfer geometric textures from style meshes or style images. Our network utilizes the position-encoded module to predict the vertex displacement along the normal direction, facilitating the synthesis of local geometric structures. We project 3D mesh models with a differentiable renderer to get 2D images and then extract style features with VGG module. The style loss based on both global and local texture features helps the network to generate fine-grained geometric textures on mesh surfaces. Compared with previous work, our network is more generic since its strong ability in transferring styles from meshes or images to the target mesh. Our network can function as a stylization tool, which provides great convenience in editing and styling mesh models.

Our network is unable to effectively learn holes on the surface of the style mesh, as the frog-bucky model shown in Fig. 5(d), which received the lowest rating in the user study. This limitation arises from the fact that our network focuses on updating the vertex positions while keeping the original triangular surface connections intact. As a result, topological changes are not possible within the model. Consequently, although our network can successfully learn and transfer local spherical shapes from the buckyball model to improve the overall roundness of the frog model, it remains challenging to accurately capture and transfer high-genus structures that are present in the style mesh. In future work, we aim to explore style transfer involving high-level structures.

Acknowledgements

The research was supported by the National Natural Science Foundation of China (Nos. 61972327, 62272402, 62372389), the Natural Science Foundation of Fujian Province (No. 2022J01001), the Fundamental Research Funds for the Central Universities (No. 20720220037) and the Start-up Fund from BNU-HKBU United International College (No. UICR0700052-23).

References

[BH11] BIAN Z., HU S.-M.: Preserving detailed features in digital bas-relief making. *Computer Aided Geometric Design 28*, 4 (2011), 245–256. 2

[BHS*17] BERKITEN S., HALBER M., SOLOMON J. M., MA C., LI H., RUSINKIEWICZ S. M.: Learning detail transfer based on geometric features. *Computer Graphics Forum 36* (2017), 361–373. 2

[CCK*18] CHOI Y., CHOI M., KIM M., HA J.-W., KIM S., CHOO J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 8789–8797. 1

[CCL*22] CHEN Y., CHEN R., LEI J., ZHANG Y., JIA K.: Tango: Text-driven photorealistic and robust 3d stylization via lighting decomposition. *arXiv preprint arXiv:2210.11277* (2022). 2

[CUYH20] CHOI Y., UH Y., YOO J., HA J.-W.: Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 8188–8197. 1

[FTSL*22] FUJI TSANG C., SHUGRINA M., LAFLECHE J. F., TAKIKAWA T., WANG J., LOOP C., CHEN W., JATAVALLABHULA K. M., SMITH E., ROZANTSEV A., PEREL O., SHEN T., GAO J., FIDLER S., STATE G., GORSKI J., XIANG T., LI J., LI M., LEBAREDIAN R.: Kaolin: A pytorch library for accelerating 3d deep learning research, 2022. 5

[GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2414–2423. 1

[GEB*17] GATYS L. A., ECKER A. S., BETHGE M., HERTZMANN A., SHECHTMAN E.: Controlling perceptual factors in neural style transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 3985–3993. 4

[HHGCO20] HERTZ A., HANOCKA R., GIRYES R., COHEN-OR D.: Deep geometric texture synthesis. *ACM Transactions on Graphics 39*, 4 (2020), 1–11. 1, 2, 6

[HLZ*22] HUANG X., LI S., ZHANG S., HAO A., QIN H.: Distribution-motivated 3d style characterization based on latent feature decomposition. *Computer-Aided Design* (2022), 103399. 2

[IZZE17] ISOLA P., ZHU J.-Y., ZHOU T., EFROS A. A.: Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 1125–1134. 1

[JAFF16] JOHNSON J., ALAHI A., FEI-FEI L.: Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision* (2016), Springer, pp. 694–711. 1

[JMD*07] JOSHI P., MEYER M., DEROSE T., GREEN B., SANOCKI T.: Harmonic coordinates for character articulation. *ACM Transactions on Graphics 26*, 3 (2007), 71–es. 2

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 5

[KFDA21] KOHLBRENNER M., FINNENDAHL U. P., DJUREN T., ALEXA M.: Gauss stylization: Interactive artistic mesh modeling based on preferred surface normals. *Computer Graphics Forum 40* (2021), 33–43. 1, 2

[KLA19] KARRAS T., LAINE S., AILA T.: A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 4401–4410. 1

[LJ19] LIU H.-T. D., JACOBSON A.: Cubic stylization. *ACM Transactions on Graphics 38*, 6 (2019), 1–10. 1, 2

[LJ21] LIU H.-T. D., JACOBSON A.: Normal-driven spherical shape analogies. *Computer Graphics Forum 40* (2021), 45–55. 1

[LLCO08] LIPMAN Y., LEVIN D., COHEN-OR D.: Green coordinates. *ACM Transactions on Graphics 27*, 3 (2008), 1–10. 2

[LLKY19] LI X., LIU S., KAUTZ J., YANG M.-H.: Learning linear transformations for fast image and video style transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 3809–3817. 4

[LTJ18] LIU H.-T. D., TAO M., JACOBSON A.: Paparazzi: Surface editing by way of multi-view image processing. *ACM Transactions on Graphics 37*, 6 (2018), 221–1. 2

[LYH*15] LUO S.-J., YUE Y., HUANG C.-K., CHUNG Y.-H., IMAI S., NISHITA T., CHEN B.-Y.: Legolization: Optimizing lego designs. *ACM Transactions on Graphics 34*, 6 (2015), 1–12. 1, 2

[MBOL*22] MICHEL O., BAR-ON R., LIU R., BENAIM S., HANOCKA R.: Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 13492–13502. 2, 6

[OMN*19] OECHSLE M., MESCHEDER L., NIEMEYER M., STRAUSS T., GEIGER A.: Texture fields: Learning texture representations in function space. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 4531–4540. 2

[RKH*21] RADFORD A., KIM J. W., HALLACY C., RAMESH A., GOH G., AGARWAL S., SASTRY G., ASKELL A., MISHKIN P., CLARK J., ET AL.: Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning* (2021), PMLR, pp. 8748–8763. 2

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2004), pp. 175–184. 2

[SZ14] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014). 3

[TSM*20] TANCIK M., SRINIVASAN P., MILDENHALL B., FRIDOVICH-KEIL S., RAGHAVAN N., SINGHAL U., RAMAMOORTHI R., BARRON J., NG R.: Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems 33* (2020), 7537–7547. 3

[Tur01] TURK G.: Texture synthesis on surfaces. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 347–354. 2

[WSH*16] WANG T. Y., SU H., HUANG Q., HUANG J., GUIBAS L. J., MITRA N. J.: Unsupervised texture transfer from images to model collections. *ACM Transactions on Graphics 35*, 6 (2016), 177–1. 2

[XLZ*10] XU K., LI H., ZHANG H., COHEN-OR D., XIONG Y., CHENG Z.-Q.: Style-content separation by anisotropic part scales. In *ACM SIGGRAPH Asia 2010 Papers* (2010), pp. 1–10. 2

[YAK*20] YIFAN W., AIGERMAN N., KIM V. G., CHAUDHURI S., SORKINE-HORNUNG O.: Neural cages for detail-preserving 3D deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 75–83. 1, 2

[YGS*21] YIN K., GAO J., SHUGRINA M., KHAMIS S., FIDLER S.: 3DStylenet: Creating 3D shapes with geometric and texture style variations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 12456–12465. 2

[YHBZ01] YING L., HERTZMANN A., BIERMANN H., ZORIN D.: Texture and shape synthesis on surfaces. In *Rendering Techniques 2001: Proceedings of the Eurographics Workshop in London, United Kingdom, June 25–27, 2001 12* (2001), Springer, pp. 301–312. 2

[ZPIE17] ZHU J.-Y., PARK T., ISOLA P., EFROS A. A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2223–2232. 1