Controllable Neural Style Transfer for Dynamic Meshes

Guilherme G. Haetinger
haetinger@disneyresearch.com
DisneyResearch|Studios
Switzerland

Jingwei Tang
jingwei.tang@disneyresearch.com
DisneyResearch|Studios
Switzerland

Raphael Ortiz
raphael.ortiz@disneyresearch.com
DisneyResearch|Studios
Switzerland

Paul Kanyuk
pkanyuk@pixar.com
Pixar Animation Studios
USA

Vinicius C. Azevedo
vinicius.azevedo@disneyresearch.com
DisneyResearch|Studios
Switzerland

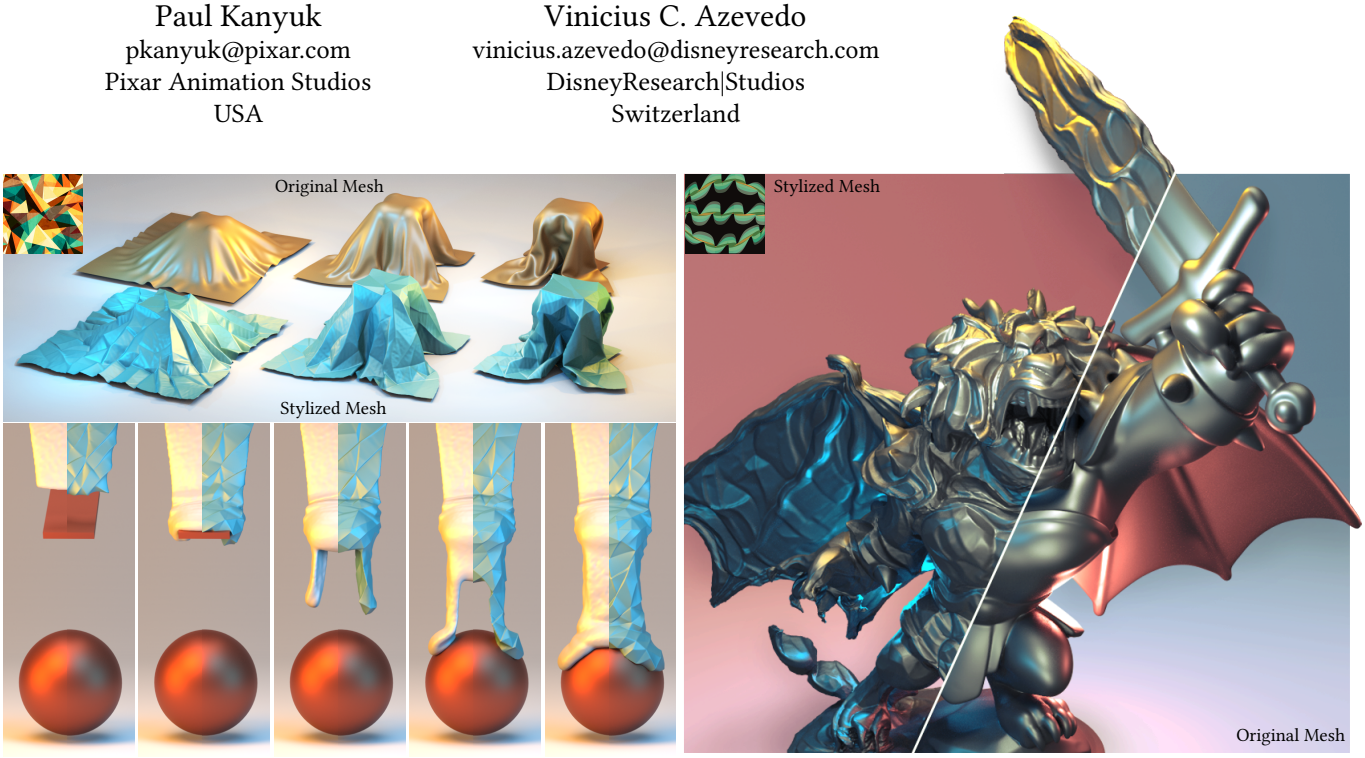


Figure 1: Our novel mesh style transfer method allows seamless transfer of 2D image styles to 3D meshes. It supports stylizing both static (right) and dynamic assets, such as cloth (top left) and liquid (bottom left) simulations. ©Disney/Pixar.

ABSTRACT

In recent years, animation movies are shifting from realistic representations to more stylized depictions that support unique design languages. To favor that, recent works implemented a Neural Style Transfer (NST) pipeline that supports the stylization of 3D assets by 2D images. In this paper we propose a novel mesh stylization technique that improves previous NST works in several ways. First, we replace the standard Gram-Matrix style loss by a Neural Neighbor formulation that enables sharper and artifact-free results. To support large mesh deformations, we reparametrize the optimized mesh positions through an implicit formulation based on the Laplace-Beltrami operator that better captures silhouette gradients that are common in inverse differentiable rendering setups. This reparametrization is coupled with a coarse-to-fine stylization setup, which enables deformations that can change large structures of the mesh. We provide artistic control through a novel method that enables directional and temporal control over synthesized styles by a guiding vector field. Lastly, we improve the previous time-coherency schemes and develop an efficient regularization that controls volume changes during the stylization process. These improvements enable high quality mesh stylizations that can create unique looks for both simulations and 3D assets.

SIGGRAPH Conference Papers '24, July 27–August 01, 2024, Denver, CO, USA

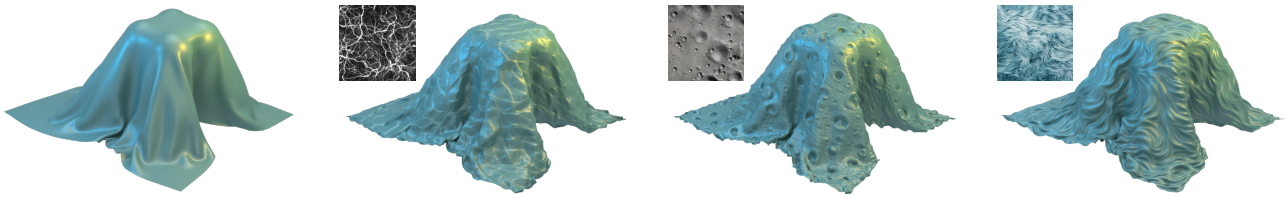
ACM ISBN 979-8-4007-0525-0/24/07
https://doi.org/10.1145/3641519.3657474

CCS CONCEPTS

• **Computing methodologies** → *Physical simulation*; **Mesh geometry models** .

KEYWORDS

Style Transfer, Optimizations, Meshes, Physics Simulations



(a) Original Mesh (b) Darkmatter Style (c) Moon Surface Style (d) Fur Style

Figure 2: Cloth stylization. Our method enables unique stylized cloth simulations, that efficiently transfer image-based styles to dynamic animated meshes.

ACM Reference Format:
Guilherme G. Haetinger, Jingwei Tang, Raphael Ortiz, Paul Kanyuk, and Vinicius C. Azevedo. 2024. Controllable Neural Style Transfer for Dynamic Meshes. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24 (SIGGRAPH Conference Papers '24), July 27–August 01, 2024, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3641519.3657474

1 INTRODUCTION

Since the debut of the first 3D animated movie, Toy Story, the industry developed a consistent and sophisticated set of tools for modeling, animating and rendering assets. In recent years, more stylized depictions are favored over realistic representations to support storytelling. This prompted researchers and animators to team up in developing tools that can support new design paradigms: artistically stylizing volumes [Kanyuk et al. 2023], dynamic mesh sharpening filters [Somasundaram et al. 2023], and artist driven linework [Grochola et al. 2023] represent a few recent successful examples.

Among these recent techniques, image-based stylization of 3D assets allows artists to achieve new unique looks in an efficient way. Previous works, however, were either focused on volumetric data [Aurand et al. 2022; Kim et al. 2019, 2020], static meshes [Liu and Jacobson 2019; Liu et al. 2018], or lacked artistic control [Gao et al. 2023; Michel et al. 2021] to be directly incorporated in animation and VFX pipelines. Moreover, mesh appearance modelling works tend to either be restricted to closely follow the surface of the input mesh [Hertz et al. 2020; Liu et al. 2018], or to be solely focused on texture synthesis [Chen et al. 2022; Höllein et al. 2022; Mishra and Granskog 2022; Yin et al. 2021].

In this paper we propose a novel mesh stylization technique that is able to produce sharp, temporally-coherent and controllable stylizations of dynamic meshes. The proposed method can seamlessly stylize assets stemming from cloth (Figure 2) and liquid (Figures 12, 8) simulations, while also enabling detailed control over the evolution of the stylized patterns over time.

At the heart of the proposed tool is a carefully designed pipeline that improves previous stylization methods in several ways. First, we replace the standard Gram-Matrix-based [Gatys et al. 2016] style loss by a Neural Neighbor [Kolkin et al. 2022] formulation that enables sharper and artifact-free results. In order to support large mesh deformations, we reparametrize the optimized mesh positions through an implicit formulation based on the Laplace-Beltrami operator [Nicolet et al. 2021] to better capture silhouette gradients, commonly present in inverse differentiable rendering setups. We couple this reparametrization with a coarse-to-fine stylization setup, which enables deformations that can change large portions of the mesh.

Control is one of the often overlooked aspects of image-based stylization. We propose a novel method that enables control over synthesized directional styles on the mesh by a guided vector field. This is embodied by augmenting the style loss with multiple orientations of the style image, which are combined with a screen-space guiding field that spatially modulates which style direction should be used. Lastly, we improve the previous time-coherency schemes [Kim et al. 2020] and develop an efficient regularization that controls volume changes during the stylization process. These improvements enable novel mesh stylizations that can create unique looks for both simulations and 3D assets (Figure 1).

2 RELATED WORK

Neural Style Transfer. Image-based Neural Style Transfer (NST) [Gatys et al. 2016] taps into the representation power of Convolutional Neural Networks trained for classification tasks [Simonyan and Zisserman 2014; Szegedy et al. 2014] to develop an innovative approach that can transfer styles between images. NST iteratively optimizes the resulting image with two complimentary objectives: a content loss that measures differences between CNN features, and a style loss that computes secondary statistics for channels within a specific layer. Many follow-up works adopted the pipeline proposed by Gatys et al. [2016]: [Li and Wand 2016; Li et al. 2017d,c; Risser et al. 2017] explored alternative feature representations to improve the style loss objective, and single [Jing et al. 2018; Johnson et al. 2016; Ulyanov et al. 2016; Wang et al. 2017], multiple [Chen et al. 2017; Dumoulin et al. 2017; Li et al. 2017a; Zhang and Dana 2019] and arbitrary style [Ghiasi et al. 2017; Huang and Belongie 2017; Li et al. 2019, 2017b; Shen et al. 2018] convolutional neural networks were trained to improve style transfer efficiency. Among notable extensions, a neural neighbor method [Kolkin et al. 2022] drastically improves stylization quality. It works by spatially decomposing the content and style images into feature vectors, and then replacing each individual content feature vector with its closest style feature through a nearest neighbor search.

Neural style transfer was also extended for 3D assets. Volumetric [Aurand et al. 2022; Guo et al. 2021; Kim et al. 2019, 2020] and mesh style transfer [Kato et al. 2018; Liu et al. 2018] pipelines utilize differentiable rendering to generate images through a set of well-distributed views to obtain a image-space loss/energy function. Stylization of 3D assets were performed with neural radiance fields

representations [Huang et al. 2021; Liu et al. 2023c; Nguyen-Phuoc et al. 2022; Zhang et al. 2022], applied to mesh texture synthesis [Frühstück et al. 2019; Höllein et al. 2022; Sendik and Cohen-Or 2017; Yin et al. 2021; Zhou et al. 2018], and combined with CLIP embeddings [Gao et al. 2023; Michel et al. 2021; Mishra and Granskog 2022] for text-based stylization. For a thorough review of 3D neural stylization, we refer to Chen et al. [2023].

Geometric Methods. Other than image-based style transfer, a set of methods focus on geometric approaches to directly modify meshes based on the style of other 3D shapes. Similarly to image analogies [Hertzmann et al. 2001], Ma et al. [2014] formulates 3D shape style transfer by computing analogies between assets. The idea is further extended by Liu and Jacobson [2021] to handle free-form deformations of the source shape. Liu and Jacobson [2019] implements cubic stylization through as-rigid-as-possible (ARAP) energy [Sorkine and Alexa 2007]. Kohlbrenner et al. [2021] extends [Liu and Jacobson 2019] by allowing interactive choices of the surface normal directions. Learning-based methods have also been employed to learn high frequency details for geometric texture synthesis [Hertz et al. 2020], perform style-aware mesh subdivision [Liu et al. 2020], and extract geometric features for stylization [Kang et al. 2023].

Diffusion Models for 3D Assets. Alongside the development of text-to-image generative models [Radford et al. 2021; Rombach et al. 2022], recent works focus on generating mesh textures with textual prompts through diffusion models [Chen et al. 2022; Richardson et al. 2023]. Many works also developed methods on the direct generation of 3D assets from text [Poole et al. 2022; Shi et al. 2023; Wang et al. 2023] or from single-view images [Liu et al. 2023b,a; Qian et al. 2023]. To the best of our knowledge, transferring 2D image styles to 3D assets through diffusion models is still an under-explored area.

3 METHOD

We follow a similar pipeline as the previous volumetric [Aurand et al. 2022; Kim et al. 2019, 2020] and mesh style transfer [Liu et al. 2018] methods (Figure 3): 3D assets renders are generated by a differentiable renderer through a set of Poisson-distributed views to obtain an image-space loss (energy) function. This loss function is minimized with respect to the mesh vertex positions $\mathbf{x}$ to obtain a stylized look. This is expressed by

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_{\theta \sim \Theta} \mathcal{L}_s(\mathcal{R}_\theta(\mathbf{x}), I^s), \tag{1}$$

where $\mathcal{R}$ is a differentiable renderer with a camera setup θ sampled from a distribution Θ of all possible configurations. The style loss ($\mathcal{L}_s$) receives the rendered ($\mathcal{R}_\theta(\mathbf{x})$) and style ($I^s$) images to evaluate the style matching objective. The stylization process is also required to ensure that the content of the generated image matches the original input. This is implemented either by initializing the optimization with the original image and making sure that the optimized variable is bounded [Aurand et al. 2022; Kim et al. 2019, 2020], or by using additional content losses [Gatys et al. 2016; Huang and Belongie 2017]. We initialize the stylized mesh to be the original mesh in our pipeline.

3.1 Neural Neighbor Style Transfer

Central to an efficient stylization is a dimensionality reduced image representation that will allow the decomposition of the image into its representative elements. Commonly, image features are computed through the feature activation maps from a pretrained classification networks such as VGG [Simonyan and Zisserman 2014] or Inception [Szegedy et al. 2014]. The style of an image is then extracted by computing the secondary statistics of those features. Specifically, the Gram matrix models correlations through a dot product between channels of a single classification network layer. However, the performance of Gram matrices can be subpar. The problem arises when synthesizing high-frequency details: the Gram matrix optimization can guide the result to focus on correlations that are not relevant in smaller scales. This creates a "washed out" stylization that converges to a local minima where high-frequency details are mixed or not clearly visible (Figure 4).

(a) Original Mesh (b) Gram Matrix (c) Neural Neighbor

Figure 4: Neural Neighbor Stylization. By stylizing the mesh with the neural neighbor loss (Equation (2)) a sharper, artifact-free stylization can be achieved: (a) original mesh; (b) stylizing with the traditional Gram matrix style loss; (c) results obtained by neural neighbor stylization equation.

Neural neighbor style transfer [Kolkin et al. 2022] fixes this issue by first spatially decomposing the content and style images into feature vectors, and then replacing each individual content feature vector with its closest style feature through a nearest neighbor search. This generates a set of style features that preserves the layout of the original image, allowing the optimization to process image corrections that can synthesize high-frequency details. The neural neighbor stylization defines the style loss as the cosine distance $\mathcal{D}_{\cos}$ between the replaced features and the feature to be optimized as

$$\mathcal{L}(I, I^s)_s = \frac{1}{N} \sum_{i=1}^{N} \mathcal{D}_{\cos}\left(\mathcal{N}\left(\mathcal{F}(I_i), \mathcal{F}(I^s)\right), \mathcal{F}(I_i)\right), \tag{2}$$

where $\mathcal{F}$ is the zero-centered feature extraction network, $\mathcal{N}$ is the function that replaces the features of a given i-th pixel of the image to be optimized I_i with the nearest neighbor feature on the style image I^s, and N is the number of pixels of the image to be optimized I.

In our pipeline, Equation (2) is plugged into Equation (1) with $I = \mathcal{R}_\theta(\mathbf{x})$, and mesh vertices are optimized such that at each iteration the cosine distance between the zero-centered extracted features of the rendered mesh and the style image are minimized. In the original work of Kolkin et al. [2022], the authors include

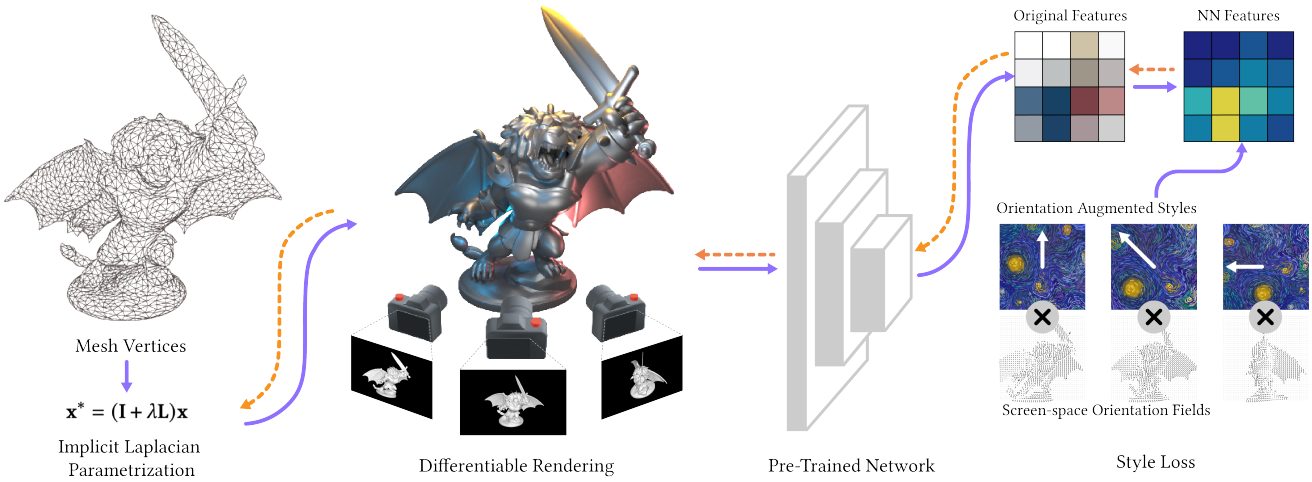


Figure 3: Mesh stylization pipeline for a single frame. In the forward pass (purple arrow), an implicit Laplacian parametrization is applied to mesh vertices. This parametrization allows the optimization process to modify large portions of the mesh. Differentiable rendering is used with Poisson sampled cameras for a view-consistent stylization. Combined with user-input orientation fields, the nearest neighbour search in the feature space efficiently captures discrepancies between the rendered and style images. The loss function gradient is backpropagated (orange arrows) to the Laplacian parametrization space and updates the mesh parameters. The technique has not yet been used in any Pixar films. ©Disney/Pixar.

a formulation that stabilizes the optimization by anchoring the nearest neighbor loss using the features of the input content image, instead of the one that is being interactively optimized I. We noticed in our experiments that not including this term does not pose any instability issues.

3.2 Multi-level Optimization through Laplacian Smoothing

Key to our approach is the decomposition of the stylization into multiple levels, enforcing a coarse-to-fine optimization process. Naively optimizing a multi-scale image-space loss function (Equation (1)), however, is not enough to modify large structures of the mesh. This happens because geometric gradients in differentiable rendering contain sparse values stemming from silhouette modifications [Nicolet et al. 2021]. Despite having large values, these silhouette gradients are not able to significantly modify large structures of the mesh due their sparsity. Therefore, the optimization process can get stuck in creating small scale structures that are overly restricted to the mesh surface.

With this observation, we reparametrize the optimized positions of Equation (1) through an implicit formulation based on the Laplace-Beltrami operator $\mathbf{L}$ in a similar way as Nicolet et al. [2021]:

$$\mathbf{x}^* = (\mathbf{I} + \lambda\mathbf{L})\mathbf{x}, \tag{3}$$

where $\mathbf{I}$ is the identity matrix and λ is a parameter to control the smoothness of the parametrization. This reparametrization effectively modifies the gradient in each optimization step as

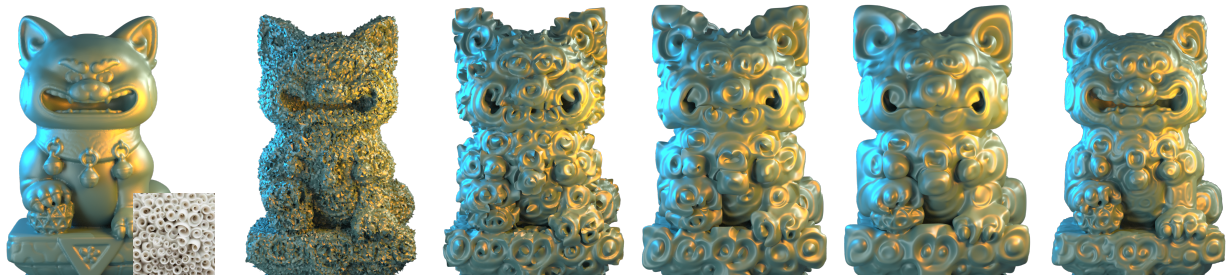
$$\mathbf{x}^* \leftarrow \mathbf{x}^* - \eta(\mathbf{I} - \lambda\mathbf{L})^{-1}\frac{\partial\mathcal{L}}{\partial\mathbf{x}^*}, \tag{4}$$

with η being the learning rate. The effect of reparametrizing the mesh positions with Equation (3) is that the sparse silhouette gradients as well as the image-space modifications are diffused to larger regions of the mesh during the backpropagation step. Figure 5 demonstrates the effect of the reparametrization: with λ set to zero in (b), the optimization process is unable to modify large structures of the mesh; higher λ values enable coarser mesh updates in (c) – (e).

To better synthesize structures at different scales, we implement a coarse-to-fine strategy that receives as input a set of rendered and style images, with optimizing images with the smallest size as the first level. The output of each coarse level stylization serves as the initialization of the next finer level. This approach also has to leverage the influence of the reparametrization proposed on Equation (3). As we progress to finer levels, we decrease the Laplacian coefficient λ to make more local/detailed changes. Figure 5(f) demonstrates how these parameters work together to create a stylized result that can modify both large scale structures and small scale details of the mesh.

3.3 Guided Stylization with Orientation Fields

One of the often overlooked aspects of incorporating mesh stylization into animation pipelines is that some styles have directional features relevant to the final result. Consider the style image that has a distinctive directional component shown in Figure 6: if one naively stylizes the mesh with this image, stylized patterns can be synthesized in arbitrary directions. We propose two modifications that allow the method to be better oriented given an input orientation field. First, the neural neighbor style loss is augmented by rotating the style image into several different orientations. Each rotated image is associated with a directional vector that indicates

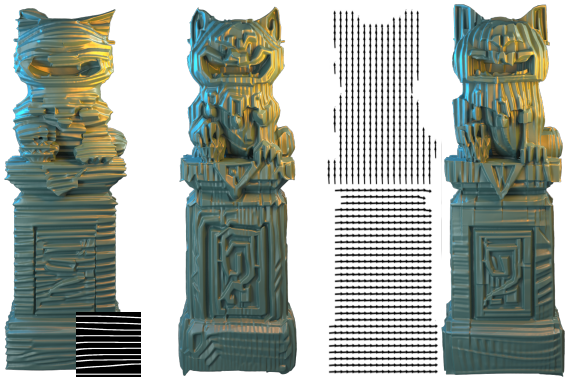


(a) Original Mesh **(b)** $\lambda = 0$ **(c)** $\lambda = 1$ **(d)** $\lambda = 5$ **(e)** $\lambda = 20$ **(f)** $\lambda = 20, 5, 0.5$

Figure 5: Stylization results for multiple λ parameters over multiple scales. As λ decreases, so does the stylization locality, until it becomes uncoordinated noise at $\lambda = 0$. The last results illustrates how using multiple λs helps generating the best stylization over all scales. ©Disney/Pixar.

the orientation of the style. Second, we allow a user-specified orientation vector field defined on the mesh. The directional vectors are then combined with a screen-space orientation field (Figure 3, bottom right) to compute a set of per-pixel weights associated with each rotated style image.

We employ a simplified rendering process for the orientation field: the user-specified orientation vectors are mapped to RGB components of a textured mesh, and then rendered with a flat shading and no lights for each camera view. The rendering still considers occlusions, so only visible orientation fields will be projected to the screen-space. These 2D orientation vectors are then combined with the directional vectors of the rotated style images through a dot product, creating several per-pixel masks that serve as weights for the style losses represented by the rotated style images. We show the effects of the control guides in Figure 6: a style image with horizontal lines is used to stylize the Panda Statue in (a); by augmenting the style loss with several orientations (b), the stylization is able to match features that are oriented in different directions, but the sense of directionality is lost; by introducing the orientation field in (c), the stylization is able to produce patterns following the user-specified input.



(a) No Aug. **(b) 8 Aug.** **(c) 8 Aug. + Guide**

Figure 6: Guided Stylization with Orientation Fields. (a) Style transfer without rotation augmentations on the Panda Statue. (b) Using 8 rotation augmentations. (c) Guiding the style loss in (b) with orientation fields. ©Disney/Pixar.

3.4 Enforcing Temporal Coherency

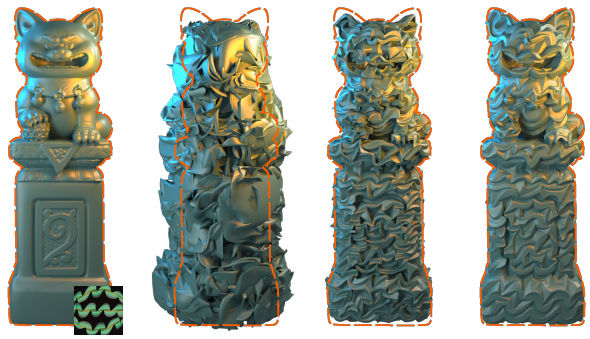
The mesh style loss in Equation (1) is only defined for a single frame, and is therefore not temporally coherent: directly optimizing it produces patterns that abruptly change across different frames. A common strategy is to align stylization displacements computed individually for different frames with the velocities defined by the underlying animation [Aurand et al. 2022; Kim et al. 2019]. The adjacent aligned displacement fields are then smoothed out to ensure that transitions between frames are continuous. To implement temporal coherency efficiently we adopt an approach that is similar to Aurand et al. [2022]: displacement contributions across multiple frames are accumulated every time-step, which requires a *single* alignment and smoothing step. For each $t > 0$, this amounts to blending displacements with

$$\mathbf{d}_t \leftarrow (1 - \gamma_\mu(\alpha))\, \mathbf{d}_t + \gamma_\mu(\alpha)\, \mathcal{T}(\mathbf{d}_{t-1}, \mathbf{u}_{t-1}), \tag{5}$$

where $\mathbf{d}_t = \hat{\mathbf{x}}_t^* - \mathbf{x}_t^*$ is the mesh displacement at timestep t and $\mathbf{u}_t$ represents vertex velocity of the animated mesh. We highlight that the displacements are computed over the Laplacian parametrized variable $\mathbf{x}^*$, which also further ensures smoothness in temporal coherency.

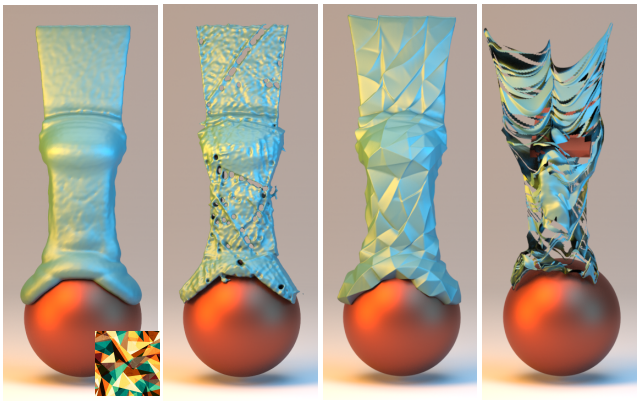
While previous EMA-based NST [Aurand et al. 2022] pipelines are able to produce temporally coherent stylizations for volumetric data, we found that this approach produces sub-par results for mesh stylizations. Therefore, we improve upon it by employing an iteration-aware function γ_μ that replaces the constant blending weight α. By adopting a linearly decaying function as iteration progresses our results are able to obtain stylizations that allow sharper synthesized patterns. The function $\gamma_\mu(\alpha) = max(\alpha(1 - \frac{m}{\mu}), 0)$ employs a decaying period factor μ that modulates the EMA smoothing weight according with the m-th iteration. We refer to our supplemental video, which better illustrates the effect of the iteration-aware EMA smoothing.

The $\mathcal{T}$ function uses the per-vertex velocities $\mathbf{u}_{t-1}$ to transport quantities defined over the mesh across subsequent frames. We chose the transport function to be the standard Semi-Lagrangian



(a) V = 23.19m^3 (b) V = 10.64m^3 (c) V = 20.98m^3 (d) V = 22.51m^3

Figure 7: Volume conservation with stochastic masking. Style transfer on the input mesh (a) using a high learning rate results (1×10^{-2}, 5×10^{-3}, 5×10^{-4}) in big volume changes without volume conservation (b). When the learning rate is tuned to be lower (1×10^{-3}, 5×10^{-4}, 5×10^{-4}) in (c), volume conservation is better, but the stylization is sub-optimal. When stochastic masking is enabled in (d), even with the learning rate as (b), the volume is conserved better. ©Disney/Pixar.



(a) Original Mesh (b) [Kim et al. 2020] (c) Ours (d) Ours (no mask)

Figure 8: Viscous sheet. LNST [Kim et al. 2020] (b) shows inferior results than ours (c), as it can only modify the mesh structures indirectly during optimization. We show the effect of using no volume conservation in (d): the optimization can generate invalid mesh configurations when stochastic masks are not used.

method, defined as

$$\mathcal{T}(\mathbf{d}_t, \mathbf{u}_t) = \mathcal{I}(\mathcal{P}(\mathbf{x}_t^*, \mathbf{u}_t), \mathbf{d}_{t-1}), \tag{6}$$

where $\mathcal{P}$ and $\mathcal{I}$ represent the position integration (e.g., Runge-Kutta) and interpolation functions, respectively. Differently from previous volumetric approaches, an interpolation function for the displacements is not readily available for animated meshes. We therefore employ a Shepard interpolation to continuously sample mesh displacements in space. For each vertex, a fixed neighborhood size of 50 is used for the interpolations of all our examples.

3.5 Volume Conservation Regularization

In some cases, the stylization procedure might induce prohibitive change of volume (Figure 7(b)), especially in thin regions of the mesh. To avoid this issue, one could enforce a divergence free constraint into the stylization displacements. This can be accomplished by restricting the displacements to follow the tangent space of the mesh [Müller 2009; Zhang et al. 2012], or by solving a Poisson system to project the displacements into their closest divergence free counterpart at each iteration [Da et al. 2016]. We tried both approaches in our early experiments, but the restriction of strictly preserving the volume at each iteration overconstrains the stylization, generating subpar results. Drastically decreasing the learning rate can improve volume conservation (Figure 7(c)), but results in less stylized results that are mostly restricted to the mesh surface.

We implemented a simpler approach that works better in practice (Figure 7(d)). At the start of each optimization scale, we initialize a random mask that covers a user-defined percentage of the vertices. These vertices are defined in the Laplacian parametrization and they are kept from being displaced by the stylization. Due to the Laplacian parametrization, masked vertices have influences on their neighboring vertices, enabling a smooth transition from non-stylized to stylized regions. For the coarser scales, we observe that the mask has to pin down vertices more aggressively to prevent volume loss; for the finest scales, no mask is necessary, since the stylization will mostly focus in creating small scale details that do not incur in significant volume loss. The effect of the proposed masking can be seen in Figure 8(d): naively stylizing thin structures present in the original animated mesh generates invalid mesh configurations; by using a mask covering 20% and 10% of the vertices in the two coarsest levels, the stylization sharply synthesizes patterns present on the style image. Algorithm 1 summarizes all the steps required by our mesh stylization pipeline.

4 EXPERIMENTS AND RESULTS

The proposed mesh stylization pipeline was implemented in PyTorch [Paszke et al. 2019] and uses PyTorch3D's Differentiable Renderer [Ravi et al. 2020]. We apply a specular material to the target meshes and render them with a light source attached to the camera position. For each optimization step, an orthographic camera is randomly sampled within a pre-defined range. To efficiently integrate the parametrization in Section 3.2 into the stylization pipeline, we precompute the $(\mathbf{I} + \lambda \mathbf{L})^{-1}$ through a Cholesky decomposition for each mesh that is stylized. We also observed that there were little differences between using combinatorial or cotagent discrete Laplacian formulations; due to efficiency and simplicity we chose the former. Typically, more iterations are used when scales goes finer. We use volume conservation regularizations from Section 3.5 in all our experiments unless otherwise mentioned. The animated meshes for liquid and cloth simulations (Figures 12, 2 and 8) are created in Houdini. We infer vertex velocities for temporal coherency treatment through a first-order Euler step. All results are rendered with Houdini's Mantra renderer. The AdamW optimizer [Loshchilov and Hutter 2019] is used with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ in both color and displacement optimizations. All experiments are conducted on a Nvidia RTX 3090 GPU. We show detailed information as well as the runtime for each example in Table (1). We refer

Algorithm 1: Displacement optimization at frame t

Input: Vertex positions $\mathbf{x}_t$; stylized vertex positions $\hat{\mathbf{x}}_{t-1}$; style image I^s; orientation field $\mathbf{G}$ (optional); Extra Non-stylization Masking $\mathbf{M}$ (optional).

Param: Set of image scales $\mathbf{S}$; set of learning rate η; set of Laplacian coefficients λ; set of number of iterations $\mathbf{N}_{\text{iters}}$; set of stochastic masking ratios $\mathbf{r}$.
▷ Each set has the cardinality of N_{scales}.

Output: Stylized vertex positions $\hat{\mathbf{x}}_t$

Camera Param. $\Theta \leftarrow$ PoissonSample() ▷ $|\Theta|$ = Sum $(\mathbf{N}_{iters})$
$\hat{\mathbf{x}}_t \leftarrow \mathbf{x}_t$ ▷ Initialization
for $\ell \leftarrow 1 : N_{scales}$ **do**
  CholeskyDecomposition$(I + \lambda_\ell L)$
  $\mathbf{M}_\ell \leftarrow$ RandomSample$(r_\ell) \cup \mathbf{M}$ ▷ Sec 3.5
  $\hat{\mathbf{x}}_t^* \leftarrow (I + \lambda_\ell L)\,\hat{\mathbf{x}}_t$
  $\hat{\mathbf{x}}_{t-1}^* \leftarrow (I + \lambda_\ell L)\,\hat{\mathbf{x}}_{t-1}$
  for $m \leftarrow 1 : \mathbf{N}^\ell_{iters}$ **do**
    $\hat{\mathbf{x}}_t^* \leftarrow$ TemporalCoherency$(\hat{\mathbf{x}}_t^*, \hat{\mathbf{x}}_{t-1}^*)$ ▷ Eq. 5
    $\hat{\mathbf{x}}_t \leftarrow$ CholeskySolve $(\hat{\mathbf{x}}_t^*)$ ▷ Eq. 3
    $\mathcal{L} \leftarrow \mathcal{L}\left(\mathcal{R}_{\Theta^\ell_m}(\hat{\mathbf{x}}_t), I^s, \mathbf{S}_\ell, \mathbf{G}\right)$ ▷ Eq. 2, Sec. 3.3
    $\hat{\mathbf{x}}_t^* \leftarrow$ AdamW $(\hat{\mathbf{x}}_t^*, \mathcal{L}, \eta_\ell)$ ▷ Eq. 4
    $\hat{\mathbf{x}}_t^* \leftarrow$ VolumeConservation$(\hat{\mathbf{x}}_t^*, \mathbf{M}_\ell)$ ▷ Sec 3.5
  end
  $\hat{\mathbf{x}}_t \leftarrow$ CholeskySolve $(\hat{\mathbf{x}}_t^*)$ ▷ Eq. 3
end

to our supplemental video, which better illustrates dynamic aspects of our stylization pipeline.

4.1 Color and Displacement Guidance Priors

Color plays an intricate and essential role in 2D image style transfer. While experimenting on how to integrate color stylization in our pipeline, we found that simultaneously optimizing for color and displacements produced results that limited the influence of the latter. This happens since stylizing a set of vertex colors for a given view is less constraining than modifying a set of vertex positions. We thus implement color and displacement optimizations in a sequential order: Figure 11(c) first optimizes displacements and then colors, while (d) first optimizes colors then displacements. Optimizing for colors first tend to bias the synthesized displacements to better follow texture patterns. We leave it as an option for the user to decide the order of the stylization, as the quality of the results is highly example-dependent.

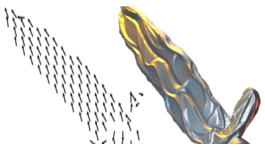
4.2 Comparisons with Previous Approaches

We compare our pipeline to previous approaches in Figure 9. The original code of *Paparazzi* [Liu et al. 2018] did not include their style transfer examples, so simplifications on our codebase were made to emulate their method. Specifically, we employed a single scale Gram matrix style loss, removed masking, and did not include the implicit Laplacian parametrization of mesh vertices. Figure 9(b) demonstrates the results of the *Paparazzi* stylization: synthesized structures are overly constrained to the mesh surface, and do not properly represent the style of the input image. Additional comparisons with *Text2Mesh* [Michel et al. 2021] (Figure 9(c)) and *TextDeformer* [Gao et al. 2023] (Figure 9(d)) are provided. Both works mostly focus on stylizing meshes with input text prompts, but the authors adopt a style loss that measures the cosine distance between the style and rendered image CLIP embeddings [Radford et al. 2021]. The stylized results demonstrate that CLIP embeddings can have difficulties representing the style image accurately. Lastly, our approach (Figure 9(e)) is able to create larger mesh deformations that can faithfully represent the input style.

Figure 8 compares our approach against the Lagrangian Neural Style Transfer (LNST) [Kim et al. 2020]. LNST can only modify mesh structures indirectly, since the stylization displacements are added to the original particle positions that control the simulation. These positions implicitly track the liquid surface changes through a particle-to-grid operation that is implemented through smooth transfer kernels. This assumption results in a liquid surface representation that does not have the necessary degrees of freedom to allow localized sharp stylizations. Thus, LNST modifications are mostly focused on creating holes that poke through the liquid sheet, and therefore fail to capture the discontinuous features required by the triangular style image (Figure 8(b)). Our approach, on the other hand, directly modifies vertex positions at the liquid interface, enabling sharp stylizations (Figure 8(c)). Lastly, we also show the effect of not using vertex masking in Figure 8(d): thin structures present in the original animated mesh can easily generate invalid mesh configurations if vertices are not appropriately pinned.

4.3 Artistic Control Inputs

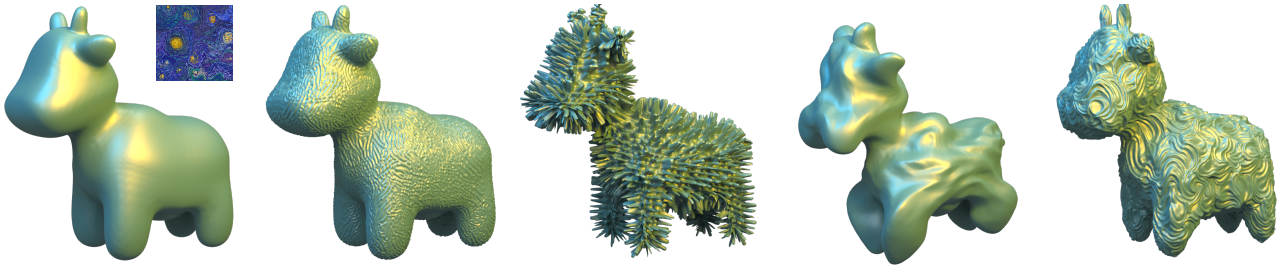
Controllable Masking. Other than the volume preservation stochastic masking in Section 3.5, our implementation also allows the input of a user-specified mask to fix a specific region to be non-stylized. This facilitates not only the artistic control of synthesized style features, but also volume conservation on thin regions of the mesh. Figure 13 shows how one can leverage this functionality by defining a mask in (b) to generate the stylization (d) while maintaining mesh structures around the Manticore's eye, nose and mouth regions. The Laplacian parametrization ensures that the transition between stylized and non-stylized regions are smooth.



Orientation Fields. Combining user-defined orientation fields with rotation augmentation of the style images can better help synthesized structures to align with the input mesh features. The inset image shows the effect of using a prescribed orientation field to align the style patterns with the sword blade in Figure 1. Another example that uses guiding orientation fields is shown in Figure 6: the input orientation field (c) is able control how the line patterns align. When no orientation field is used (a), the stylization follows the horizontal direction for the whole mesh.

4.4 Stylization on Animated Meshes

Stylization results on animated meshes are shown in Figures 2, 12, 14, 8. We tested different parameters for temporal coherency



(a) Original Mesh (b) [Liu et al. 2018] (c) [Michel et al. 2021] (d) [Gao et al. 2023] (e) Ours

Figure 9: Style transfer comparisons for Spot. The original mesh (a) is stylized with the starry night image (inset) with different methods. All experiments are run on the input mesh with 210K vertices, except for (d), which runs on a mesh with 3K vertices as it hits a memory bottleneck. The runtime for each methods are: (b) 72s, (c) 755s (d) 741s and (e) 147s.

Table 1: Parameters and performance statistics.

Example	Diff Render Resolution	No. Vertices	Image Scales	Laplacian Coefficient λ	Learning Rate ($\times 10^{-3}$)	Runtime (s/frame)
Manticore (Fig. 11)	800×800	700K	[0.5, 0.5, 1.25]	[20.0, 5.0, 1.0]	[8.0, 1.0, 5.0]	230
Manticore (Fig. 1)	700×700	700K	[0.5, 1.0, 1.5]	[20.0, 5.0, 0.5]	[9.0, 4.0, 1.0]	178
Panda (Fig. 7)	800×800	760K	[0.5, 1.0, 1.5]	[20.0, 5.0, 0.5]	[10.0, 5.0, 0.5]	118
Cloth (Fig. 2)	400×400	160K	[0.5, 1.0, 1.5]	[20.0, 5.0, 2.0]	[10.0, 5, 1.5]	18
Liquids (Fig. 12)	400×400	180K	[0.5, 1.0, 1.5]	[20.0, 5.0, 0.5]	[3.5, 0.8, 0.4]	20
Viscous Sheet (Fig. 8)	700×700	200K	[0.5, 0.8, 1.0]	[20.0, 5.0, 3.0]	[5.0, 1.0, 0.25]	14
Spot (Fig. 9)	1000×1000	210K	[0.4, 0.8, 1.5]	[20.0, 5.0, 1.0]	[8.0, 2.0, 0.5]	147
Lost Soul (Fig. 14)	800×800	175K	[0.25, 0.5, 1.0]	[20.0, 5.0, 1.0]	[4.0, 3.0, 1.0]	56

(temporal smoothing coefficient α, and temporal decay period M) for the cloth stylization sequence. These results can be seen in our supplemental video. We stylized the liquid simulations (Figure 12) and the animated character (Figure 14) with a displacement first, color second order. For better temporal consistency, we only sample the stochastic mask once at the first frame and keep it fixed over time on the vertices in the cloth example (Figure 2). The Lost Soul example (Figure 14) includes orientation and velocity fields generated by a curl-noise [Bridson et al. 2007] to intentionally guide and displace patterns over time.

5 CONCLUSIONS

In this paper we present a production-friendly pipeline for mesh neural style transfer (NST) based on 2D images. To provide a better image space style loss gradient, we replace the commonly used Gram Matrix style loss with a neural neighbor formulation. A Laplacian parametrization space is employed to enable a coarse-to-fine optimization procedure that is able to modify large portions of the mesh. To enable artistic control over synthesized pattern directions, we propose to use an input orientation field together with rotation augmented style images in computing the style loss. Furthermore, we improve upon the previous EMA-based temporal coherency treatment by adding an iteration-aware decay term on the blending weight. Finally, for volume and content conservation during the stylization process, a stochastic mask is used to pin down vertices to be non-stylized.

Our pipeline has a few limitations. When color optimization is applied (Figure 11, 14), the final colorized mesh does not properly match the color distribution of the style image. This is a known issue of the neural neighbor formulation [Kolkin et al. 2022]. The authors apply a color matching post-processing on the stylized image in the original paper, which can potentially be integrated in our pipeline. Using stochastic masking does not guarantee a strict volume conservation, and temporal flickering can be observed when a higher percentage of the vertices is pinned down. Other mesh volume conservation techniques, such as tetrahedralizing the mesh and applying a volume conservation step can be further explored to alleviate this issue. Lastly, our stylizations do not guarantee that the resulting meshes will be interpenetration-free.

As future work, diffusion models [Ho et al. 2020; Rombach et al. 2022] can provide a promising venue for exploration. Image features extracted from diffusion models are much more abundant than those from pre-trained image classification networks, and can thus provide more powerful stylization capabilities [Hertz et al. 2024].

REFERENCES

Joshua Aurand, Raphael Ortiz, Silvia Nauer, and Vinicius C. Azevedo. 2022. Efficient Neural Style Transfer for Volumetric Simulations. *ACM Transactions on Graphics* 41, 6 (Nov. 2022). https://doi.org/10.1145/3550454.3555517 Publisher: Association for Computing Machinery.

Robert Bridson, Jim Houriham, and Marcus Nordenstam. 2007. Curl-noise for procedural fluid flow. *ACM Transactions on Graphics* 26, 3 (July 2007), 46–es. https://doi.org/10.1145/1276377.1276435

Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. 2017. StyleBank: An Explicit Representation for Neural Image Style Transfer. https://doi.org/10.48550/arXiv.1703.09210 arXiv:1703.09210 [cs].

Yongwei Chen, Rui Chen, Jiabao Lei, Yabin Zhang, and Kui Jia. 2022. *TANGO: Text-driven Photorealistic and Robust 3D Stylization via Lighting Decomposition.* Technical Report. https://cyw-3d.github.io/tango.

Yingshu Chen, Guocheng Shao, Ka Chun Shum, Binh-Son Hua, and Sai-Kit Yeung. 2023. Advances in 3D Neural Stylization: A Survey. https://doi.org/10.48550/arXiv.2311.18328 arXiv:2311.18328 [cs].

Fang Da, David Hahn, Christopher Batty, Chris Wojtan, and Eitan Grinspun. 2016. Surface-only liquids. *ACM Transactions on Graphics* 35, 4 (July 2016), 1–12. https://doi.org/10.1145/2897824.2925899

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. 2017. A Learned Representation For Artistic Style. https://doi.org/10.48550/arXiv.1610.07629 arXiv:1610.07629 [cs].

Anna Frühstück, Ibraheem Alhashim, and Peter Wonka. 2019. TileGAN: synthesis of large-scale non-homogeneous textures. *ACM Transactions on Graphics* 38, 4 (Aug. 2019), 1–11. https://doi.org/10.1145/3306346.3322993

William Gao, Noam Aigerman, Thibault Groueix, Vova Kim, and Rana Hanocka. 2023. TextDeformer: Geometry Manipulation using Text Guidance. In *ACM SIGGRAPH 2023 Conference Proceedings (SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3588432.3591552

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2414–2423. https://doi.org/10.1109/CVPR.2016.265

Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. 2017. Exploring the structure of a real-time, arbitrary neural artistic stylization network. In *Procedings of the British Machine Vision Conference 2017*. British Machine Vision Association, London, UK, 114. https://doi.org/10.5244/C.31.114

Pav Grochola, Filippo Maccari, Young Joon Lee, and Edmond Boulet-Gilly. 2023. Linework in Spider-Man Across the Spider-Verse: An artistic driven approach to linework generation. In *ACM SIGGRAPH 2023 Talks*. ACM, Los Angeles CA USA, 1–2. https://doi.org/10.1145/3587421.3595456

Jie Guo, Mengtian Li, Zijing Zong, Yuntao Liu, Jingwu He, Yanwen Guo, and Ling-Qi Yan. 2021. Volumetric appearance stylization with stylizing kernel prediction network. *ACM Transactions on Graphics* 40, 4 (July 2021), 162:1–162:15. https://doi.org/10.1145/3450626.3459799

Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. 2020. Deep geometric texture synthesis. *ACM Transactions on Graphics* 39, 4 (July 2020). https://doi.org/10.1145/3386569.3392471 arXiv: 2007.00074 Publisher: Association for Computing Machinery.

Amir Hertz, Andrey Voynov, Shlomi Fruchter, and Daniel Cohen-Or. 2024. Style Aligned Image Generation via Shared Attention. https://doi.org/10.48550/arXiv.2312.02133 arXiv:2312.02133 [cs].

Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. 2001. Image Analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 327–340. https://doi.org/10.1145/383259.383295

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems* 2020-Decem, NeurIPS 2020 (2020), 1–25. arXiv: 2006.11239.

Hsin-Ping Huang, Hung-Yu Tseng, Saurabh Saini, Maneesh Singh, and Ming-Hsuan Yang. 2021. Learning to Stylize Novel Views. https://doi.org/10.48550/arXiv.2105.13509 arXiv:2105.13509 [cs].

Xun Huang and Serge Belongie. 2017. Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Venice, 1510–1519. https://doi.org/10.1109/ICCV.2017.167

Lukas Höllein, Justin Johnson, and Matthias Nießner. 2022. *StyleMesh: Style Transfer for Indoor 3D Scene Reconstructions*. Technical Report. https://lukashoel.github.io/stylemesh/

Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao, and Mingli Song. 2018. Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*. Springer-Verlag, Berlin, Heidelberg, 244–260. https://doi.org/10.1007/978-3-030-01261-8_15

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Computer Vision – ECCV 2016*. Vol. 9906. Springer International Publishing, Cham, 694–711. https://doi.org/10.1007/978-3-319-46475-6_43 Series Title: Lecture Notes in Computer Science.

Hongyuan Kang, Xiao Dong, Juan Cao, and Zhonggui Chen. 2023. Neural style transfer for 3D meshes. *Graphical Models* 129 (Oct. 2023), 101198. https://doi.org/10.1016/j.gmod.2023.101198

Paul Kanyuk, Vinicius Azevedo, Raphael Ortiz, and Jingwei Tang. 2023. Singed Silhouettes and Feed Forward Flames: Volumetric Neural Style Transfer for Expressive Fire Simulation. In *ACM SIGGRAPH 2023 Talks*. ACM, Los Angeles CA USA, 1–2. https://doi.org/10.1145/3587421.3595435

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3D Mesh Renderer. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Salt Lake City, UT, 3907–3916. https://doi.org/10.1109/CVPR.2018.00411

Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. 2019. Transport-based neural style transfer for smoke simulations. *ACM Transactions on Graphics* 38, 6 (Nov. 2019). https://doi.org/10.1145/3355089.3356560 arXiv: 1905.07442 Publisher: Association for Computing Machinery.

Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. 2020. Lagrangian neural style transfer for fluids. *ACM Transactions on Graphics* 39, 4 (July 2020). https://doi.org/10.1145/3386569.3392473 arXiv: 2005.00803 Publisher: Association for Computing Machinery.

M. Kohlbrenner, U. Finnendahl, T. Djuren, and M. Alexa. 2021. Gauss Stylization: Interactive Artistic Mesh Modeling based on Preferred Surface Normals. *Computer Graphics Forum* 40, 5 (2021), 33–43. https://doi.org/10.1111/cgf.14355 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14355.

Nicholas Kolkin, Michal Kucera, Sylvain Paris, Daniel Sykora, Eli Shechtman, and Greg Shakhnarovich. 2022. Neural Neighbor Style Transfer. (March 2022). http://arxiv.org/abs/2203.13215 arXiv: 2203.13215.

Chuan Li and Michael Wand. 2016. Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Las Vegas, NV, USA, 2479–2486. https://doi.org/10.1109/CVPR.2016.272

Shaohua Li, Xinxing Xu, Liqiang Nie, and Tat-Seng Chua. 2017d. Laplacian-Steered Neural Style Transfer. In *Proceedings of the 25th ACM international conference on Multimedia*. ACM, Mountain View California USA, 1716–1724. https://doi.org/10.1145/3123266.3123425

Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. 2019. Learning Linear Transformations for Fast Image and Video Style Transfer. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Long Beach, CA, USA, 3804–3812. https://doi.org/10.1109/CVPR.2019.00393

Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. 2017a. Diversified Texture Synthesis with Feed-Forward Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Honolulu, HI, 266–274. https://doi.org/10.1109/CVPR.2017.36

Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. 2017b. Universal Style Transfer via Feature Transforms. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/hash/49182f81e6a13cf5eaa496d51fea6406-Abstract.html

Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. 2017c. Demystifying Neural Style Transfer. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Melbourne, Australia, 2230–2236. https://doi.org/10.24963/ijcai.2017/310

Hsueh Ti Derek Liu and Alec Jacobson. 2019. Cubic stylization. *ACM Transactions on Graphics* 38, 6 (Nov. 2019). https://doi.org/10.1145/3355089.3356495 arXiv: 1910.02926 Publisher: Association for Computing Machinery.

Hsueh-Ti Derek Liu and Alec Jacobson. 2021. Normal-Driven Spherical Shape Analogies. *Computer Graphics Forum* 40, 5 (2021), 45–55. https://doi.org/10.1111/cgf.14356 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14356.

Hsueh-Ti Derek Liu, Vladimir G. Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. 2020. Neural subdivision. *ACM Transactions on Graphics* 39, 4 (Aug. 2020). https://doi.org/10.1145/3386569.3392418

Hsueh-ti Derek Liu, Michael Tao, Alec Jacobson, and Hsueh-Ti Derek Liu. 2018. Paparazzi: Surface Editing by way of Multi-View Image Processing. 1, 1 (2018), 11. https://doi.org/10.1145/8888888.7777777

Kunhao Liu, Fangneng Zhan, Yiwen Chen, Jiahui Zhang, Yingchen Yu, Abdulmotaleb El Saddik, Shijian Lu, and Eric Xing. 2023c. StyleRF: Zero-shot 3D Style Transfer of Neural Radiance Fields. (March 2023). http://arxiv.org/abs/2303.10598 arXiv: 2303.10598.

Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. 2023b. One-2-3-45: Any Single Image to 3D Mesh in 45 Seconds without Per-Shape Optimization. http://arxiv.org/abs/2306.16928 arXiv:2306.16928 [cs].

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. 2023a. Zero-1-to-3: Zero-shot One Image to 3D Object. https://doi.org/10.48550/arXiv.2303.11328 arXiv:2303.11328 [cs].

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. https://doi.org/10.48550/arXiv.1711.05101 arXiv:1711.05101 [cs, math].

Chongyang Ma, Haibin Huang, Alla Sheffer, Evangelos Kalogerakis, and Rui Wang. 2014. *Analogy-Driven 3D Style Transfer*. Technical Report. Volume: 33 Issue: 2.

Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. 2021. Text2Mesh: Text-Driven Neural Stylization for Meshes. (Dec. 2021). arXiv: 2112.03221.

Shailesh Mishra and Jonathan Granskog. 2022. CLIP-based Neural Neighbor Style Transfer for 3D Assets. https://doi.org/10.48550/arXiv.2208.04370 arXiv:2208.04370 [cs].

Matthias Müller. 2009. Fast and robust tracking of fluid surfaces. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, New Orleans Louisiana, 237–245. https://doi.org/10.1145/1599470.1599501

Thu Nguyen-Phuoc, Feng Liu, and Lei Xiao. 2022. SNeRF: Stylized Neural Implicit Representations for 3D Scenes. *ACM Transactions on Graphics* 41, 4 (July 2022). https://doi.org/10.1145/3528223.3530107 arXiv: 2207.02363 Publisher: Association for Computing Machinery.

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large steps in inverse rendering of geometry. *ACM Transactions on Graphics* 40, 6 (Dec. 2021). https://doi.org/10.1145/3478513.3480501 Publisher: Association for Computing Machinery.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani,

Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. http://arxiv.org/abs/1912.01703 arXiv:1912.01703 [cs, stat].

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2022. DreamFusion: Text-to-3D using 2D Diffusion. (Sept. 2022). http://arxiv.org/abs/2209.14988 _eprint: 2209.14988.

Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. 2023. Magic123: One Image to High-Quality 3D Object Generation Using Both 2D and 3D Diffusion Priors. http://arxiv.org/abs/2306.17843 arXiv:2306.17843 [cs].

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. https://doi.org/10.48550/arXiv.2103.00020 arXiv:2103.00020 [cs].

Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. 2020. Accelerating 3D Deep Learning with PyTorch3D. https://doi.org/10.48550/arXiv.2007.08501 arXiv:2007.08501 [cs].

Elad Richardson, Gal Metzer, Yuval Alaluf, Raja Giryes, and Daniel Cohen-Or. 2023. TEXTure: Text-Guided Texturing of 3D Shapes. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*. ACM, Los Angeles CA USA, 1–11. https://doi.org/10.1145/3588432.3591503

Eric Risser, Pierre Wilmot, and Connelly Barnes. 2017. Stable and Controllable Neural Texture Synthesis and Style Transfer Using Histogram Losses. http://arxiv.org/abs/1701.08893 arXiv:1701.08893 [cs].

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2022-June (Dec. 2022), 10674–10685. https://doi.org/10.1109/CVPR52688.2022.01042 ISBN: 9781665469463 _eprint: 2112.10752.

Omry Sendik and Daniel Cohen-Or. 2017. Deep Correlations for Texture Synthesis. *ACM Transactions on Graphics* 36, 5 (Oct. 2017), 1–15. https://doi.org/10.1145/3015461

Falong Shen, Shuicheng Yan, and Gang Zeng. 2018. Neural Style Transfer via Meta Networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Salt Lake City, UT, 8061–8069. https://doi.org/10.1109/CVPR.2018.00841

Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. 2023. MVDream: Multi-view Diffusion for 3D Generation. 2 (2023), 1–18. http://arxiv.org/abs/2308.16512 arXiv: 2308.16512.

Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. (2014). https://doi.org/10.48550/ARXIV.1409.1556 Publisher: arXiv Version Number: 6.

Arunachalam Somasundaram, Levi Biasco, and Damon Riesberg. 2023. Dynamic Mesh Sharpening. In *ACM SIGGRAPH 2023 Talks*. ACM, Los Angeles CA USA, 1–2. https://doi.org/10.1145/3587421.3595428

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Proceedings of the fifth Eurographics symposium on Geometry processing (SGP '07)*. Eurographics Association, Goslar, DEU, 109–116.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going Deeper with Convolutions. https://doi.org/10.48550/arXiv.1409.4842 arXiv:1409.4842 [cs].

Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky. 2016. Texture networks: feed-forward synthesis of textures and stylized images. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. JMLR.org, New York, NY, USA, 1349–1357.

Xin Wang, Geoffrey Oxholm, Da Zhang, and Yuan-Fang Wang. 2017. Multimodal Transfer: A Hierarchical Deep Convolutional Neural Network for Fast Artistic Style Transfer. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Honolulu, HI, 7178–7186. https://doi.org/10.1109/CVPR.2017.759

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. 2023. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation. (2023), 1–34. http://arxiv.org/abs/2305.16213 arXiv:2305.16213.

Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 2021. 3DStyleNet: Creating 3D Shapes with Geometric and Texture Style Variations. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Montreal, QC, Canada, 12436–12445. https://doi.org/10.1109/ICCV48922.2021.01223

Hang Zhang and Kristin Dana. 2019. Multi-style Generative Network for Real-Time Transfer. In *Computer Vision – ECCV 2018 Workshops*. Vol. 11132. Springer International Publishing, Cham, 349–365. https://doi.org/10.1007/978-3-030-11018-5_32 Series Title: Lecture Notes in Computer Science.

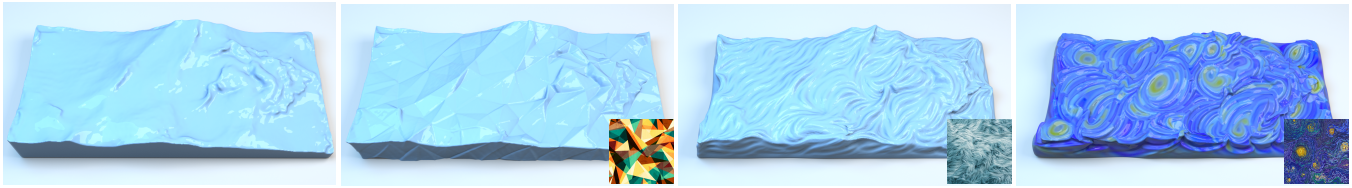
Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. 2022. ARF: Artistic Radiance Fields. https://doi.org/10.48550/arXiv.2206.06360 arXiv:2206.06360 [cs].

Yizhong Zhang, Huamin Wang, Shuai Wang, Yiying Tong, and Kun Zhou. 2012. A Deformable Surface Model for Real-Time Water Drop Animation. *IEEE Transactions on Visualization and Computer Graphics* 18, 8 (Aug. 2012), 1281–1289. https://doi.org/10.1109/TVCG.2011.141

Yang Zhou, Zhen Zhu, Xiang Bai, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. 2018. Non-stationary texture synthesis by adversarial expansion. *ACM Transactions on Graphics* 37, 4 (Aug. 2018), 1–13. https://doi.org/10.1145/3197517.3201285

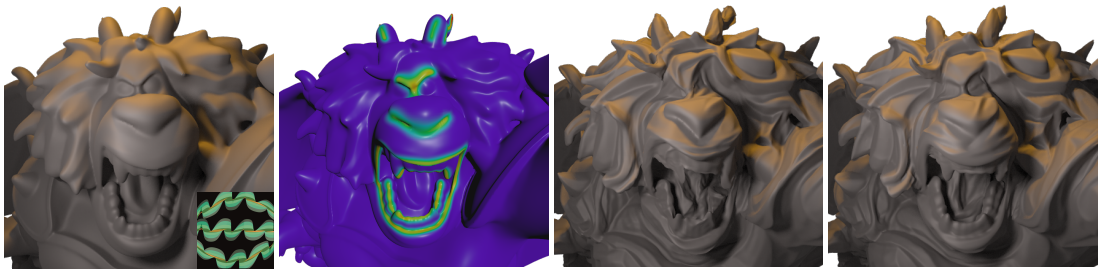
(a) Original Mesh (b) Displacement (c) Displacement → Color (d) Color → Displacement

Figure 11: Displacement and color optimizations. (b) only runs vertex displacement optimization on the original mesh (a) with the angular spiral style image (inset). (c) runs color optimization with (b) as initialization. (d) runs color optimization, and use it as initialization for the displacement optimization. ©Disney/Pixar.



(a) Original Mesh (b) Triangle Style (c) Fur Style (d) Starry Night Style (w/ Color)

Figure 12: Liquid stylization. Our method is able to stylize animated liquid meshes without any special treatment. Since it directly modifies the mesh surface vertices, it can create sharper results than previous liquid stylization techniques.



(a) Original Mesh (b) User-defined Mask (c) Unmasked Stylization (d) Masked Stylization

Figure 13: Controllable masking examples on Manticore. Masking the facial features of the original mesh (a) allows us to maintain the sculptor's desired look. The unmasked result (c) shows clear volume loss on the teeth and background horns, while the masked version (d) enables proper stylization with smooth transitions between deformations. ©Disney/Pixar.

(a) Original Mesh (b) Angular Spiral Style (c) Fire Style (d) Foam Style (e) Triangle Style

Figure 14: Style transfers on Lost Soul. All experiments run our mesh style transfer on the animated character. The displacement optimization is performed first, and color optimization later. ©Disney/Pixar.