Stroke Transfer: Example-based Synthesis of Animatable Stroke Styles

Hideki Todo*
Aoyama Gakuin University (AGU)
Kanagawa, Japan
htodo@cs.takushoku-u.ac.jp

Kunihiko Kobayashi*
Aoyama Gakuin University (AGU)
Kanagawa, Japan
kuni.koba.one-piece@outlook.com

Jin Katsuragi
Aoyama Gakuin University (AGU)
Kanagawa, Japan
jincgcg@gmail.com

Haruna Shimotahira
Aoyama Gakuin University (AGU)
Kanagawa, Japan
suansushui@gmail.com

Shizuo Kaji
Kyushu University
Fukuoka, Japan
skaji@imi.kyushu-u.ac.jp

Yonghao Yue
Aoyama Gakuin University (AGU)
Kanagawa, Japan
yonghao@it.aoyama.ac.jp

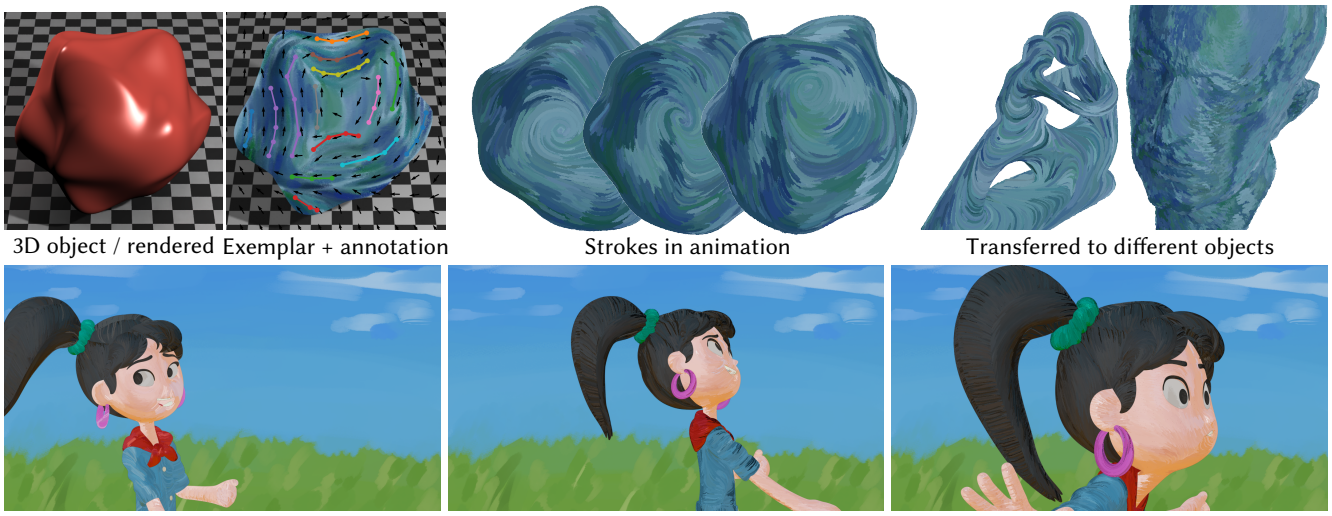


Figure 1: (Top) Taking an object and its simple rendering, as well as a static exemplar image with annotations as input, our method can generate animations with the stroke styles given via the exemplar for changes in the viewing and lighting conditions and object shapes (possibly with a change in topology). (Bottom) Application of our method to a complex animation.

ABSTRACT

We present *stroke transfer*, an example-based synthesis method of brushstrokes for animated scenes under changes in viewpoint, lighting conditions, and object shapes. We introduce *stroke field* for guiding the generation of strokes, consisting of spatially varying attributes of strokes, namely, their orientations, lengths, widths, and colors. Strokes are synthesized as the integral curves of the stroke field. In essence, we separate elements that constitute the artistic stroke into *style-specific* transferable elements and *instance-intrinsic* ones. To generate the stroke field, we first compute a set of vector fields that reflect the instance-intrinsic elements and then combine them using style-specific weight functions learned from exemplars, with the weights computed in a proxy feature space shared among a variety of objects. The rendered animation using our method captures time-varying viewpoint, lighting conditions, and object shapes, as well as the artistic style given in the form of exemplars.

CCS CONCEPTS

• **Computing methodologies** → **Non-photorealistic rendering**.

KEYWORDS

Non-photorealistic rendering, stroke-based rendering, example-based, stroke transfer, vector field generation

*Co-first authors — authors contributed equally.

SIGGRAPH '22 Conference Proceedings, August 7–11, 2022, Vancouver, BC, Canada

ACM ISBN 978-1-4503-9337-9/22/08…$15.00
https://doi.org/10.1145/3528233.3530703

ACM Reference Format:
Hideki Todo, Kunihiko Kobayashi, Jin Katsuragi, Haruna Shimotahira, Shizuo Kaji, and Yonghao Yue. 2022. Stroke Transfer: Example-based Synthesis of Animatable Stroke Styles. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings (SIGGRAPH '22 Conference Proceedings), August 7–11, 2022, Vancouver, BC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3528233.3530703

1 INTRODUCTION

Brushstrokes in paintings reflect artists' style and individuality. For instance, (post-)impressionists, including Vincent van Gogh, make use of expressive brushstrokes to convey vivid and striking impression of the world in their own unique views. Brushstrokes are common in a variety of art forms, and many of them come with well-developed computer-assisted tools, including oil paintings [Baxter et al. 2001; Chen et al. 2015], crayon drawings [Rudolf et al. 2003], watercolor paintings [Curtis et al. 1997], and Indian-ink paintings (also known as moxi or sumie) [Chu and Tai 2005]. Beyond traditional art forms, where brushstrokes exist on the 2D canvas or image plane, digital forms, such as WYSIWYG NPR [Kalnins et al. 2002], Deep Canvas [Katanics and Lappa 2003], and OverCoat [Schmid et al. 2011], allow brushstrokes to act like decal textures tied directly with the target curved surfaces in the 3D space.

When *animated*, the styles of brushstrokes make the films exceptionally stand out from others, with examples being "Loving Vincent" [Kobiela and Welchman 2017] (literally) in the style of Vincent van Gogh throughout the film with nearly 64,000 frames painted manually by more than 50 painters [Mackiewicz and Melendez 2016], and "The Tale of The Princess Kaguya" [Takahata 2014] in the style of Indian-ink painting. The high demand in manual work, however, makes the styles of brushstrokes hard to be applied in animations more broadly.

Example-based synthesis of artistic styles is thus a promising direction to animated expressions of brushstrokes. Typical prior art takes user drawn images as inputs and tries to stitch *2D patches* of the input images together to reconstruct a target drawing with the same artistic style (e.g., [Bénard et al. 2013; Fišer et al. 2016]).

In the painting process in the real world, the artist observes an object in the 3D space and draws it on a 2D canvas. Hence, the resulting brushstrokes usually exhibit a combination of both 2D and 3D effects [Durand 2002]: the strokes may follow the smooth illumination change on the *curved* surface, or may follow the view-dependent silhouette lines apparent in the 2D image plane.

We study example-based synthesis of brushstrokes with these geometric backgrounds in consideration, aiming at relating strokes' attributes with the artist's (usually unconscious) intension in following the silhouette, shading, and/or geometry of the target object, and *transferring* the intended stroke styles given as an *exemplar* to an animation sequence of a new target (Fig. 1).

We view an art form with brushstrokes as a collection of strokes registered on the target 3D surface[1] as in WYSIWYG NPR [Kalnins et al. 2002], Deep Canvas [Katanics and Lappa 2003], and OverCoat [Schmid et al. 2011], though these strokes may vary over time with coherence upon animation. We recognize the stroke styles as their orientations, colors, widths, and lengths. Mathematically, we introduce *stroke field* to represent the collection of strokes, defined as sections of a vector bundle over the surface representing the stroke orientation as well as stroke color, length, and width.

Given a 3D object[2] and a setting for the camera and lighting, we render a static image with simple (diffuse and specular) shading, and let an artist provide an *exemplar* by drawing strokes on the image and *annotations* for the stroke styles. At the core of our method, we learn from the given exemplar a *model* to automatically generate the stroke field for synthesizing the strokes for a given *new* and/or *animated* setting. To decouple the transferable *style-specific* elements from the *instance-intrinsic* elements, we formulate our model as a map from a *proxy feature space*, valid for the new setting, to the weights for combining *canonical sections*, or overly determined bases, of the stroke field. For both the features and canonical sections, we account for the geometry (e.g., curvature and normal), illumination (e.g., diffuse and specular lighting), as well as the viewpoint (e.g., view dependent silhouette lines).

Given new settings for a (new) 3D object, view, and lighting, we compute the canonical sections anew and use the learned model with the features at each point on the surface to predict the weights for linearly combining the newly computed canonical sections to generate the new stroke field. We in addition perform filtering via optimization for spatial and temporal coherence. The brushstrokes are finally generated as integral curves of the stroke field.

We evaluate our method quantitatively in terms of the reconstruction error of the regression for the orientations (Fig. 4), and qualitatively by a visual inspection for the colors (Fig. 5), widths, and lengths.

Our novelty mainly lies in our formulation that enables the decomposition of the instant-intrinsic elements and style-specific elements by representing stroke styles by a simple (linear) representation with learnable weights. Our framework enables automatic stroke transfer for painterly drawings with 1) learned stroke styles (unlike many previous work with hard-coded formula (e.g., [Hegde et al. 2013] §3.3) for determining the orientations), 2) only a small amount of user input (entire animation generated using only a single exemplar), and 3) no stitching artifacts by construction as we treat each stroke separately (unlike patch-based approaches). Our approach offers an interpretable representation of brushstrokes since the importance of a certain factor is reflected by the weight in the linear representation. For example, a particular style may be analyzed and described as "near the boundary, the artist tends to align the strokes along the boundary." We show applications of our method (Fig. 1) to animations of object shapes, lighting conditions, and viewing conditions.

2 RELATED WORK

2.1 Expressive Brushstrokes

There have been a vast number of prior art for expressive painterly rendering. We refer to Hertzmann [2003] and Hegde et al. [2013] for comprehensive surveys. A standard pipeline for automatic generation of a painterly rendering is to first construct an orientation field for the alignment of the strokes, and perform placement of

[1] Note that it is also possible to view the brushstrokes as being registered on the 2D canvas. The choice of the target domain is rather a matter of taste.

[2] This is different from Loving Vincent [Mackiewicz and Melendez 2016], where the input to the artists is an image sequence encoding shading, lighting, and silhouettes.

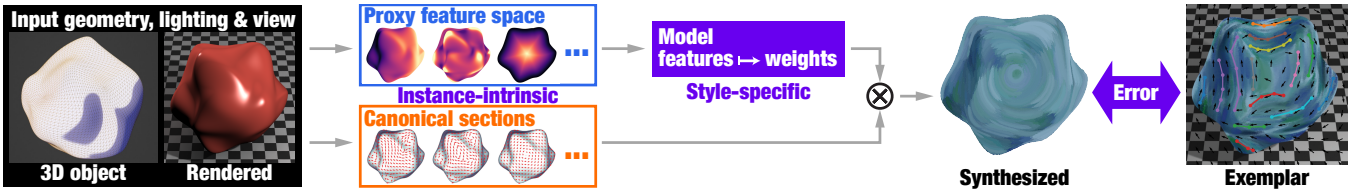


Figure 2: Illustration of our idea. Given a 3D object and its simple rendering (left), we ask an artist to draw an exemplar together with annotations (right). We also extract *instance-intrinsic* features and canonical sections (or overly determined bases of the orientations), and learn a *style-specific* model that converts the set of features into a set of weights, which work as coefficients for the canonical sections to synthesize the orientations, to minimize the error between the synthesized orientations and those from the exemplar. The learned style-specific weights can then be reused for the synthesis for other objects.

individual strokes using the orientations. Our framework builds on this pipeline as well.

For generating *animated* painterly drawings (including fully manual ones and semi-automatic ones), one could let the artist draw frame by frame [Kobiela and Welchman 2017; Mackiewicz and Melendez 2016], provide directional stroke fields [Meier 1996], or paint key frames [Bassett et al. 2013] on 3D geometries via OverCoat [Schmid et al. 2011]. Here, we focus on automatic generation using a few exemplars for reducing the painting workload.

2.2 Example-based approaches

There are different aspects for what an 'example' can mean in expressive painterly rendering. The first is individual stroke (e.g., [Lu et al. 2013; Zheng et al. 2017]), where the way for synthesizing the texture of a single stroke is learned from exemplars. The second is mixing (e.g., [Lu et al. 2013]), where realistic color mixing of pigments (as opposed to the Kubelka-Munk model [Kubelka and Munk 1931]) is learned from exemplars.

The third is the *compositions* of strokes in the image space, where the *patches* of the input 2D image is stitched together to reconstruct a target image via image analogies [Hertzmann et al. 2001] for an animated 3D scene [Bénard et al. 2013], or its extensions to use more advanced features from illumination [Fišer et al. 2016; Sýkora et al. 2019] or photorealistic inputs [Jamriška et al. 2019]. One could also use deep neural networks [Gatys et al. 2016] or a combination of a patch-based approach with deep neural networks [Futschik et al. 2021; Texler et al. 2020] for the reconstruction of the image.

Our focus is the compositions of the individual strokes in the stroke space, i.e., the mechanism to generate a guide (including orientation fields) for placing individual strokes. By synthesizing each stroke in separate, we are free from stitching artifacts that may occur in 2D patch-based approaches. There have been methods for example-based placing of strokes in a part of the object region, including those for stylized silhouettes [Cardona and Saito 2015; Kalnins et al. 2003; Northrup and Markosian 2000; Singh and Schaefer 2010], for representative feature lines of fur, grass and trees [Kowalski et al. 1999; Markosian et al. 2000], and for hatching in pen-and-ink illustration [Haga et al. 2001; Kalogerakis et al. 2012]. We in contrast focus on placing the strokes in the entire visible region for an expressive painterly style drawing. Inspired by prior art on painterly drawings [Hegde et al. 2013; Hertzmann 2003; Meier 1996], we learn an *interpretable model* for determining the colors, flow, and sizes needed to synthesize the brushstrokes.

2.3 Vector and Tensor Fields, Drawing Strokes

Our method constructs vector fields to guide the generation of brushstrokes. There have been methods for designing user-guided vector fields on 3D sufaces [Fisher et al. 2007; Zhang et al. 2006] and its extension to animations [Chen et al. 2011], as well as methods using vector fields for texture synthesis [Liu et al. 2013; Turk 2001; Xu et al. 2009], hair design [Fu et al. 2007], curvature design [Iarussi et al. 2015], hatching [Kratt et al. 2017], generating integral curves [Ray and Sokolov 2014], and remeshing for isotropic triangular or quad-dominant meshes [Jakob et al. 2015]. Building on the discrete differential geometry aspects of these methods, we focus on how to learn the generation of vector fields for synthesizing brushstrokes using exemplars, avoiding having the user to directly design a spatially and temporally coherent vector field.

For those styles, such as a pencil drawing, with which only the trajectory of the strokes matters, the N-RoSy fields and cross fields (e.g., [Kagaya et al. 2011; Zhang et al. 2007]) that do not encode the sign of the orientation are more suitable alternatives than vector fields. Depending on the style, the sign of the orientation is sometimes hard to tell from the exemplar or even cannot be determined in a consistent way. We leave the extension using tensor fields as a future work.

The strokes can be generated on the 2D screen space using an image based technique (e.g., [Huang et al. 2013; Spencer et al. 2009]) or on the 3D object space (e.g., [Kalnins et al. 2002; Katanics and Lappa 2003; Schmid et al. 2011]) like ours. In terms of the styles, the choice is most likely a matter of taste. Implementation-wise, there is no free lunch; working in the object space is simple because the consistent handling of occluded strokes is achieved for free at the cost of processing all strokes including hidden ones, whereas working in the screen space is computationally more efficient if special care is paid for the visibility near a silhouette or occluded region in a temporally coherent way. We choose to work in the object space for simplicity.

3 OVERVIEW

Given a target (possibly animating) 3D object together with lighting (Fig. 2), with its surface expressed as a time-varying manifold $M(t)$, we generate a set of time-varying and view-dependent brushstrokes $\mathcal{S}(t)$ registered on the manifold. For creating plausible and lively brushstrokes, not only the object shape and illumination, but also the 2D view play important roles. For example, if the object is

zoomed in, we would need more strokes to fill in the space, as the widths of the strokes are fixed in the image space. Therefore, our model for generating $\mathcal{S}(t)$ takes the geometry and topology of $M(t)$, the lighting $L(t)$, as well as the view $V(t)$ into account.

We first consider one frame at a time, and discuss time evolution later. We model the brushstrokes as the *realization* (§6) of a *stroke field* (§5) to guide the generation. In reality, the brushstrokes drawn by an artist can be thought of as the *superimposition* of realizations of *multiple* stroke fields. We model each of such stroke fields independently.

As depicted in previous work (e.g., [Hegde et al. 2013; Meier 1996]), a stroke is attributed by its position, orientation, length, width, color, and texture. Our stroke field is intended to encode these information, which may either vary smoothly or discretely (i.e., per-stroke) over the manifold. For each attribute, our framework can be configured to flexibly adapt to the smoothness or discreteness[3]. By default, we model the orientations as a smoothly varying vector field, with randomness to each stroke added later on during stroke generation while other attributes, colors, widths, and lengths, are modelled per-stroke. This choice is made since the orientations of two adjacent strokes are likely to be similar in a usual painterly drawing, whereas colors, widths, and lengths vary significantly giving the liveliness or expressiveness to the style.

Mathematically put, a stroke field consists of the following data: at each point $p \in M$, we have $\boldsymbol{u}(p) \in T_pM$, a tangent vector representing (unnormalized) stroke *orientation*; $l(p) \in \mathbb{R}$, a scalar representing stroke *length*[4]; $w(p) \in \mathbb{R}$, a scalar representing stroke *width*; $\boldsymbol{C}(p) \in \mathbb{R}^3$, a tuple of scalars representing stroke *color* in RGB. Scalar fields l, w, and $\boldsymbol{C}$ are nothing but functions on M. The vector field $\boldsymbol{u}$ is a *section* of the tangent bundle $\pi : TM \to M$; a section of π is a map $s : M \to TM$ that satisfies $\pi \circ s = id$, where id is the identity.

Our core idea is to decompose various factors that produce artistically plausible strokes so that strokes can be easily edited and transferred. We first construct $N_A = 6$ *canonical sections*[5] $\hat{A}_i$ of TM, each reflecting a different property of the input; those accounting for the geometry of M, those for the illumination L, as well as those for the view-dependent silhouette V. We discuss the construction of the canonical sections $\hat{A}_i$ in §4.1.

Another key idea is that we map each point $p \in M$ to a *feature space* by the *feature map* $\phi_{L,V} : M \to \mathbb{R}^{N_F}$ that encodes the local features in the vicinity of p in the manifold M as well as in the 2D image Q rendered with the given L and V. We denote by $\Pi : M \to Q$ and $\Pi^{-1} : Q \to M$ the correspondence between the point on the manifold and the rendered image[6]. We consider an $N_F = 9$ dimensional feature space with each coordinate representing (lighting-dependent) illumination, (geometry-dependent) normal and curvature features, and (view-dependent) distances to

[3]Positions are always discrete.

[4]We found that decomposing the length from the orientation this way allows for more faithful adaptation to the given exemplar.

[5]The rank of TM is two, and any section can be written as the sum of two independent sections weighted by two scalar functions on M. However, we construct a set of *overly determined* canonical sections, i.e., more canonical sections (six in our case) than the least degrees of freedom required to represent the stroke field, so that even a simple (for example, linear) regression model as the weight function achieves sufficient expressiveness as we will see later.

[6]They are partial maps as some points of M are occluded, but in what follows, we only consider those points which have a one-to-one correspondence between M and P.

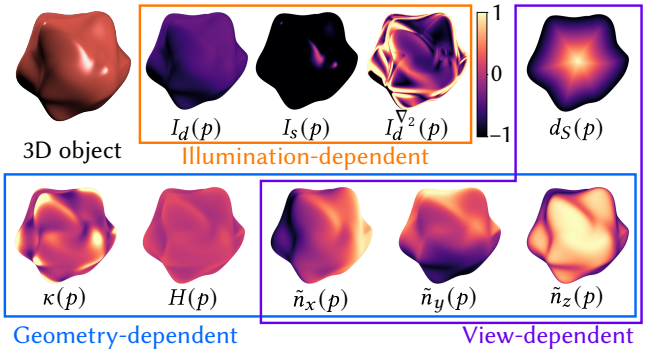


Figure 3: Visualization of features.

feature lines. Specifically, we first define *raw* features $\phi^{\mathrm{raw}}_{L,V}(p) = (I_d(p), I_s(p), I_d^{\nabla_2}(p), \tilde{n}_x(p), \tilde{n}_y(p), \tilde{n}_z(p), \kappa(p), H(p), d_S(p)) \in \mathbb{R}^9$, where $I_d(p) = I_d(\Pi(p))$ (resp. $I_s(p) = I_s(\Pi(p))$) is the tone mapped diffuse (resp. specular) intensity of the rendered image at p, $I_d^{\nabla_2}(p) := \|\nabla_2 I_d(p)\|$ is the magnitude of the apparent gradient[7] (i.e., defined in the 2D image plane) of the tone mapped diffuse intensity, $\tilde{n}_x(p), \tilde{n}_y(p), \tilde{n}_z(p)$ are the apparent normal (i.e., $(\tilde{n}_x(p), \tilde{n}_y(p), \tilde{n}_z(p)) = \mathbf{M}^{\mathrm{MV}}(\boldsymbol{n}(p))$, where $\mathbf{M}^{\mathrm{MV}}$ is the model-view matrix). $\kappa(p)$ is the Gaussian curvature, $H(p)$ is the mean curvature, and $d_S(p)$ is the minimum distance to the silhouette lines. As in our supplementary material, we then encode the raw features into a *relativized* and *normalized* internal representation $\phi_{L,V}(p)$, as in Fig. 3. The key point is that the feature space serves as the proxy space shared among different settings of (M, L, V) for generating the stroke field valid for an *animated* or a *new* setting, allowing for a *transferable* stroke representation. The internal representation improves the transferability by aligning the range of encoded features in the exemplar(s) and in the targets. Our encoding allows for reasonable results even when we use a single exemplar.

We hypothesize that the smoothly varying (unnormalized) stroke orientations in TM can be modeled by the weighted sum of the canonical sections:

$$\boldsymbol{u}(p) = W_{\boldsymbol{u}}(\phi_{L,V}(p)) \cdot \hat{A}(p), \tag{1}$$

where $W_{\boldsymbol{u}} : \mathbb{R}^{N_F} \to \mathbb{R}^{N_A}$ is a weight function. Similarly, each of the other smoothly varying scalar attributes B_s are modeled by

$$B_s(p) = W_{B_s}(\phi_{L,V}(p)), \tag{2}$$

where $W_{B_s} : \mathbb{R}^{N_F} \to \mathbb{R}$. This way, the smoothly varying elements in a stroke field at any point on the object are determined by combining the canonical and constant fields weighted using local information around p. We learn the weight functions W_* from user drawn exemplars by fitting a *regression model* (§4).

For each of the (discrete) per-stroke elements B_d, we found that simply using a nearest neighbor query to the exemplar gives plausible results:

$$B_d(p) = W_{B_d}(\Lambda(\phi_{L,V}(p))), \tag{3}$$

[7]We use ∇_2 to mean the spatial derivative in the 2D image plane.

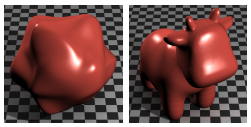
where $\Lambda(\phi_{L,V}(p))$ returns the data point in the feature space collected from the exemplar closest to the given feature $\phi_{L,V}(p)$ with respect to the Euclidean metric in the encoded feature space, and $W_{B_d}(\Lambda(\phi_{L,V}(p)))$ returns its value.

Given a new geometry M', illumination L', and view V', we reuse the weight functions W_* fitted to the original triple (M, L, V) but with the new canonical sections $\hat{A}_i'$ computed from (M', L', V') to generate a candidate stroke field for the new setting. In this way, we separate elements that constitute the artistic stroke into "style-specific" transferable elements encoded in W_* and "instance-intrinsic" elements encoded in $\hat{A}_i'$.

To further improve spatial and temporal coherency, we formulate a simple optimization problem for obtaining the final stroke field (§5), which is in turn used to guide the generation of the brushstrokes as integral curves of the final stroke field (§6).

4 REGRESSION

To gather an input exemplar, we prepare a rendered image of a 3D object as reference, and let the artist provide drawings, as in StyLit [Fišer et al. 2016]. In our prototype, we consider simple illumination conditions. We show examples of reference images in the right inset, which are rendered using a few point light sources accounting for diffuse and specular reflections. When drawing, we ask the artist to group strokes into layers, each forming a realization of a single stroke field. Ideally, each stroke in a layer could be identified from the drawing history or via a time-lapse video [Tan et al. 2015]. In our prototype, we built a simple tool to identify the strokes via manual annotation: for a set of representative strokes visible in the exemplar image, their orientations, lengths, and widths are manually specified, where the lengths and widths are measured in the image space. Finally, we use the radial basis function (with the distance measured on the 2D canvas) to interpolate the orientations, length, and width (Fig. 1).



4.1 Smoothly Varying Stroke Attributes

We construct the unnormalized stroke orientation (i.e., tangent vectors) as a linear combination of deterministic canonical sections of the tangent bundle of M. We define the following $N_A = 6$ canonical sections $M \to TM$: 1) the apparent intensity gradient $\boldsymbol{I}^{\nabla_2}{}_{d\|} := \Pi^{-1}(\mathcal{N}(\nabla_2 I_d))$, where I_d is the (tone mapped) diffuse intensity calculated from the lighting L, surface normal of M, and the view V, $\mathcal{N}(\cdot)$ is the normalization operator (i.e., $\mathcal{N}(\boldsymbol{x}) := \boldsymbol{x}/\|\boldsymbol{x}\|$), and $\Pi^{-1}(\cdot)$ is the push forward operator for lifting a vector in the 2D image plane to the corresponding tangent vector on the surface in the 3D object space. We then include 2) the lift $\boldsymbol{o}_{s\|} = \Pi^{-1}(\mathcal{N}(\boldsymbol{o}_{2,s\|}))$ of directions interpolated from silhouette lines $\boldsymbol{o}_{2,s\|}$ via radial basis functions. We also include 3) the apparent gradient of the tilt of the surface normal from the image plane $\boldsymbol{n}_\| := \Pi^{-1}(\mathcal{N}(\nabla_2 \tilde{n}_z))$. Finally, we include their $\pi/2$-rotations in the image plane 4) $\boldsymbol{I}^{\nabla_2}{}_{d\perp} := \Pi^{-1}(\mathrm{Rot}_{\pi/2}(\mathcal{N}(\nabla_2 I_d)))$, 5) $\boldsymbol{o}_{s\perp} := \Pi^{-1}(\mathrm{Rot}_{\pi/2}(\mathcal{N}(\boldsymbol{o}_{2,s\|})))$, and 6) $\boldsymbol{n}_\perp := \Pi^{-1}(\mathrm{Rot}_{\pi/2}(\mathcal{N}(\nabla_2 \tilde{n}_z)))$.

For each exemplar, we know its underlying 3D geometric features, the resulting lighting conditions, as well as the stroke attributes drawn by the artist in the 2D image plane Q. For the ease of computation, we learn the weight functions by minimizing the following reconstruction error functional defined in the image plane, measuring the differences in the projected orientations, as:

$$\mathcal{E}_{\boldsymbol{u}}(W_{\boldsymbol{u}}) = \int_{\Pi(M)} \|\boldsymbol{u}_0(q) - \Pi(W_{\boldsymbol{u}}(\phi_{L,V}(\Pi^{-1}(q))) \cdot \hat{A}(\Pi^{-1}(q)))\|^2 dQ, \quad (4)$$

where $\Pi(M)$ is the visible region of M in the exemplar, $\boldsymbol{u}_0(q)$ is the stroke orientation, a 2D tangent vector, at $q \in \Pi(M)$ of the exemplar. As for the function space of the weight functions, we simply consider a linear expression (including a constant bias term), and hence use linear regression. To evaluate its effectiveness, we compare the exemplar and prediction in Fig. 4. Whereas choosing a globally constant weight function (i.e., irrespective of the features) (Fig. 4 (a, d)) would fail to reconstruct the exemplar orientations, our weight function accounting for local features (Fig. 4 (b, e)) provides sufficient reconstruction quality, and thus, just enough degree-of-freedom. We have also tried other models such as a higher-order polynomial regression for the weight functions to observe better fitting but less stability. See our supplementary material for the details.

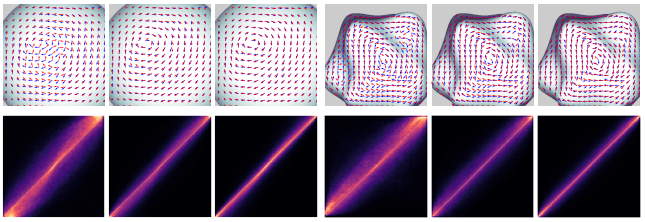


Figure 4: Evaluation of our regression. We compare exemplar orientations (blue arrows) with reconstructed orientations (red arrows) for the (a-c) sphere and (d-f) blobby examples, using globally constant weight functions (a, d), our weight functions accounting for local features in the first (b, e) and second (c, f) order representations. In the bottom, we visualize the alignment between exemplar and reconstructed orientations, where we compare their magnitudes in randomly sampled test directions of unit length in the 3D object space. The use of our functions allows narrower spread from the perfect alignment corresponding to the diagonal line, indicating better agreement.

If necessary, smoothly varying scalar attributes can be learned similarly by minimizing

$$\mathcal{E}_{Q_s}(w_{Q_s}) = \int_{\Pi(M)} \|Q_{s,0}(q) - W_{Q_s}(\phi_{L,V}(\Pi^{-1}(q)))\|^2 dQ. \quad (5)$$

4.2 Per-Stroke Attributes

To preserve the color variation in the synthesized results, we use 1 nearest neighbor regression of the local features $\phi_{L,V}(p)$, instead of linear regression. As shown in Fig. 5, nearest neighbor regression exhibits higher and live color variation similar to the StyLit [Fišer et al. 2016] result. Even for the back side of object, for which the exemplar seemingly does not provide information, our regression result maintains temporal stability. Likewise, we use 1 nearest neighbor regression for stroke lengths and widths.

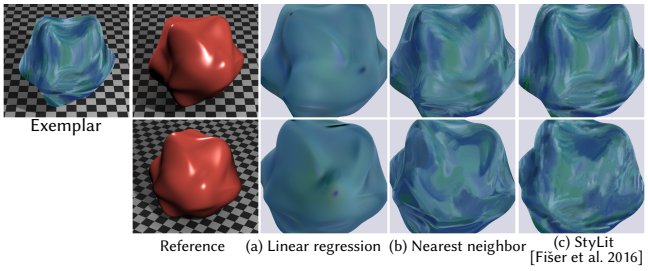


Figure 5: Comparison of predicted color attributes. (a) Linear regression from local features $\phi_{L,V}(p)$ exhibits spatially smooth results, but leads to suppression in the color variation. (b) Nearest neighbor regression from the same features $\phi_{L,V}(p)$ has a higher color variation, suited for a painterly rendering, similar to the patch-based synthesis using StyLit [Fišer et al. 2016]. In (a) and (b), we are only showing the color attributes, not the synthesized strokes.

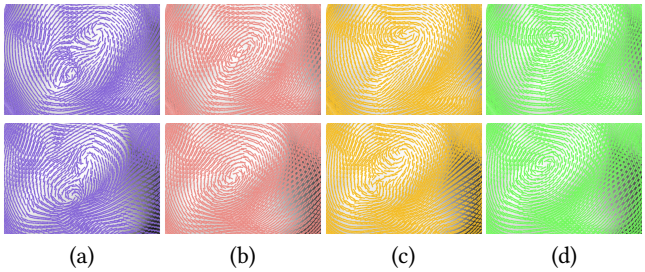


Figure 6: Comparison of the optimization settings for spatial and temporal filtering. (a) Input random orientation field. (b) Spatially smoothed orientation field with $\lambda_s = 0.07$, $\lambda_t = 10^{-10}$. (c) Temporally smoothed orientation field with $\lambda_s = 10^{-10}$, $\lambda_t = 0.5$. (d) Our setting for making results with $\lambda_s = 0.07$, $\lambda_t = 0.5$.

5 SYNTHESIZING STROKE FIELD

For per-stroke attributes, we simply use (3) per frame and do not perform any spatial or temporal filtering.

For smoothly varying attributes, we first generate tentative attributes for each frame separately, then synthesize the spatially and temporally coherent ones via optimization. For the orientations, we first extract canonical sections in the same way as in Section 4 per frame. Then, we use our model (1) to predict the orientations $\boldsymbol{u}(t,p)$ per frame at all points on the 3D surface M including both visible and invisible ones. We process orientations at invisible points for allowing (partially) invisible strokes to turn into visible smoothly and coherently. Then, for better coherence, we solve the optimization problem of the following objective functional with the regularization parameters $\lambda_s \geq 0$ and $\lambda_t \geq 0$ to control spatial and temporal coherence, respectively:

$$\check{\boldsymbol{u}}(t,p) = \operatorname{argmin}_{\tilde{\boldsymbol{u}}(t,p)} \int_0^T \int_{M(t)} \left(\|\tilde{\boldsymbol{u}}(t,p) - \boldsymbol{u}(t,p)\|^2 + \lambda_s \|\nabla \tilde{\boldsymbol{u}}(t,p)\|^2 + \lambda_t \left\| \frac{\partial}{\partial t} \tilde{\boldsymbol{u}}(t,p) \right\|^2 \right) dM dt, \quad (6)$$

where $\nabla : \Gamma(TM) \to \Gamma(T^*M \otimes TM)$ is the Levi-Civita connection, and all the norms are induced by the metric of M.

The first term in (6) penalizes the deviation from the predicted orientations $\boldsymbol{u}(t,p)$, while the second and third terms penalize spatial and temporal variations, respectively. The effect of varying the influence of the second and third terms is discussed in Fig. 6. The process for smoothly varying scalar attributes follows similarly. In our implementation, M is represented by a triangulated manifold mesh, and a vector field is assumed to be piecewise linear. We discretize the above functional via discrete exterior calculus [Crane et al. 2013; Desbrun et al. 2006; Fisher et al. 2007; Hirani 2003]. Through the integration of the inner product of the field and each oriented edge e of the mesh along e, we convert the vector field $\boldsymbol{u}(t,p)$ to the discrete 1-form $\boldsymbol{c}^k \in C^1(M)$ according to Fisher et al. [2007], where $k \geq 1$ is the frame index. Here, a discrete 1-form $\boldsymbol{c}^k$ is represented as a vector whose components are indexed by the oriented edges $\mathcal{E}$. Then, the optimization reduces to

$$\{\check{\boldsymbol{c}}^k\} = \operatorname{argmin}_{\{\tilde{\boldsymbol{c}}^k\}} \sum_k \left(\star_1 \left\| \tilde{\boldsymbol{c}}^k - \boldsymbol{c}^k \right\|^2 + \lambda_s (\tilde{\boldsymbol{c}}^k)^\top \boldsymbol{L}_1 \tilde{\boldsymbol{c}}^k + \lambda_t \frac{\star_1}{\Delta t} \left\| \tilde{\boldsymbol{c}}^k - \tilde{\boldsymbol{c}}^{k-1} \right\|^2 \right), \quad (7)$$

where Δt is the time interval between frames $k-1$ and k, $\star_1$ is the discrete Hodge star, and $\boldsymbol{L}_1$ is the discrete covariant Laplacian. We further approximate the optimization by solving it sequentially: we first perform spatial filtering, followed by temporal filtering using the spatially filtered 1-forms as the input. The resulting optimization problem is a linear system. We applied frame-by-frame smoothing because this was implementation-wise simpler and much better in terms of memory use, and did not show noticeable artifacts. Finally, the orientation field is reconstructed from the 1-form $\check{\boldsymbol{c}}^k$ using the Whitney elements [Whitney 1957], as discussed in Fisher et al. [2007, §2.4]. While we learn a style-specific model in regression, the smoothing is an instance-intrinsic operation, hence performed separately. Smoothly varying scalar attributes follow similarly.

6 SYNTHESIZING STROKES

We synthesize each stroke as an integral curve of the stroke field. Specifically, we need to decide an anchor point $q \in M$ (start point), initial orientation (tangent vector) $\boldsymbol{u}(q) \in T_qM$, length l, width w, and color $\boldsymbol{C}$ of the curve. We use a user-specified texture to draw each stroke. Inspired by "Loving Vincent" [Kobiela and Welchman 2017; Mackiewicz and Melendez 2016], where only necessary strokes are *overdrawn* atop the strokes from the previous frame, we generate coherent animation as follows.

First, we *predetermine* a *hierarchy* of anchor points. The hierarchy is for handling possible changes in the zoom level during the animation. Starting from the first frame, we use Poisson disk sampling [Bowers et al. 2010] with an initial radius r_0 (5% of the diagonal length of the bounding box of the object) to generate points on the object surface, forming the anchor points belonging to the 0-th level. To generate anchor points in the i-th level, we perform Poisson disk sampling *incrementally*, atop all anchor points in the $0, \ldots, (i-1)$-th levels, using a reduced radius (i.e., $r_i = 0.5 r_{i-1}$) applied to already generated points as well. In our current implementation, we use 4 levels in total. Then, we propagate the anchor points in all levels to the next frame. Note that gaps may appear in the second frame due to the deformation of the object. Thus, we generate additional anchor points to fill in such gaps starting from the coarsest level (i.e., 0-th level). All anchor points in the second

frame are then propagated to the next frame. We repeat this process for the rest of the frames.

Next, we synthesize the strokes starting from the first frame. We begin by setting the entire *visible* surface region of the first frame as 'active.' Starting from the coarsest level, we accept predetermined anchor points if they are within the active region. Each time we accept an anchor point q, we generate a stroke starting from the anchor point. We determine its length l_q and width w_q, both measured in the screen space, and color C_q, by querying the values predicted from our model[8]. To trace along the tangent vector field, we first choose a random angular offset ξ_q according to a uniform distribution $U(-\sigma_{\boldsymbol{u}}, \sigma_{\boldsymbol{u}})$, and solve the following differential equation with the initial condition $p(0) = \Pi^{-1}(q)$ until the stroke length measured in the screen space becomes l_q:

$$\frac{d}{dt}p(t) = \mathrm{Rot}(\xi_q, p(t), \boldsymbol{u}(p(t))), \tag{8}$$

where $\mathrm{Rot}(\xi_q, p(t), \boldsymbol{u}(p(t)))$ means a rotation of $\boldsymbol{u}(p(t))$ in the tangent space at $p(t)$ in angle ξ_q. Random angular offsets within $\sigma_{\boldsymbol{u}} = 5^\circ$ allow the emulation of natural variation seen in hand-drawn styles of artists (as in our supplementary video).

We 'deactivate' the region that would be painted by the generated stroke[9]. If all the anchor points in the current level are used but there are still active regions, we proceed to the next level. To draw the strokes, we first render a base layer serving as 'undercoat,' where the color of each point is queried using our model. This undercoat is an extension of Lit-Sphere [Sloan et al. 2001] and works to cover remaining gaps even when we have used up all anchor points[10] (see Fig. 7). Atop of this base layer, the chosen strokes are drawn in an order sorted across all the 4 levels according to their luminance, darker one first, which gives nicely looking painterly expression as shown in §7 and the supplementary video.

When proceeding on to the next frame, we propagate the strokes from the previous frame. For each anchor point in the current frame, we check whether there is a corresponding point in the previous frame from which a stroke is drawn. It yes, we propagate the *random number* used to alter its orientation for a coherent alternation. We generate the stroke using the tangent vector, length, width, and color, queried from the stroke field at the current frame together with the propagated random number. If the resulting drawing of the current frame has gaps, we mark those regions as 'active' and follow the process used for the first frame to accept additional anchor points. For the order of drawing, we first draw all of the propagated strokes. Atop of them, we draw the newly generated strokes sorted across all the 4 levels in the order of their luminance.

7 RESULTS

7.1 On the Stroke Fields and Generated Strokes

In our supplementary video, we show the influence of canonical sections on the resulting orientations. With only the view-dependent canonical sections $\boldsymbol{o}_{s\parallel}$ and $\boldsymbol{o}_{s\perp}$, the resulting orientations are stable but too boring. On the other hand, with the other canonical sections ${\boldsymbol{I}^{\nabla_2}}_{d\parallel}$, ${\boldsymbol{I}^{\nabla_2}}_{d\perp}$, $\boldsymbol{n}_{\parallel}$, and $\boldsymbol{n}_{\perp}$, the resulting orientations can become unstable when lighting conditions or normals vary significantly. The use of the combination of these canonical sections leads to a stable yet lively motion in the orientations.

We also compare the effectiveness of the use of the anchor points and sorting in terms of temporal coherence in our supplementary video. The use of the coherent anchor points and sorting greatly reduce temporal flickering due to the shift in stroke positions and the change in drawing orders when there is no inter-frame coherence.

(a) Undercoat (b) Strokes only (c) Composition

Figure 7: Stroke rendering over the undercoat. (a) Our predicted color attributes as the undercoat. (b) Integral curves only. (c) The composition of (b) atop (a) as the output.

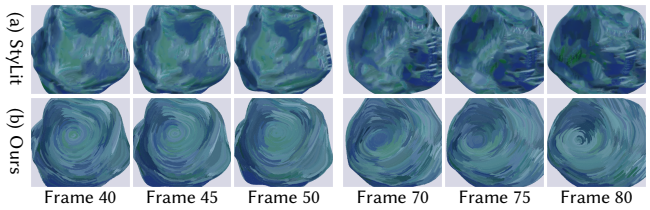


Figure 8: Comparison with patch-based stylization. (a) StyLit [Fišer et al. 2016]. (b) Our approach.

7.2 Comparison to Patch-based Approach

Unlike photorealistic inputs, our inputs are simple renderings with much fewer visual details, making it hard to apply recent neural network based approaches using VGG features, as discussed by Fišer et al. [2016] and Hauptfleisch et al. [2020]. Here, we compare our method to the state-of-the-art guided stylization framework of StyLit [Fišer et al. 2016]. To run their algorithm, we provide illumination effects (diffuse and specular images in RGB colors, computed using the same lighting and shading models as ours) as the guiding channels. StyLit tends to produce patch orientation drift, causing temporal artifacts on stroke motion. Even with the optimization for patch-consistency, maintaining coherent orientation motions is hard in the image space. Our method offers much better temporal stability, while still retaining the expressive stroke styles, as in Fig. 8 and our supplementary video.

7.3 Applications

First, we show our application to animations in the viewpoint in Fig. 9 (top), as well as simultaneous lighting and view animation in Fig. 9 (middle). With the change in the view, part of the object previously unseen will be drawn anew. Even though this occurs

[8]We also leave knobs for the user to adjust the lengths and widths, by allowing the user to multiply globally constant scalars for scaling. If necessary, the user could also apply color adjustments as a post-processing step.

[9]When computing the region for deactivation, we *reduce* lengths and widths by a factor of 0.5 to accept more anchor points for a denser stroke coverage, similar to the use of the reduced radius in avoid-a-void [Yue et al. 2015].

[10]Deeper hierarchy would result in an overly long computation time.

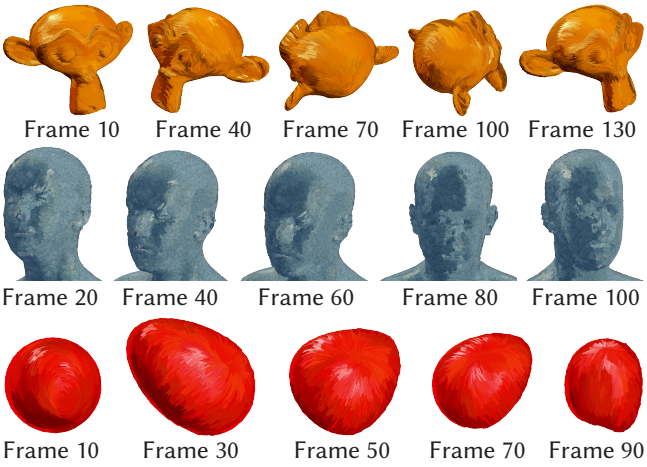


Figure 9: Top: view animation. Middle: combined view and lighting animation. Bottom: stroke animation for a deforming object.

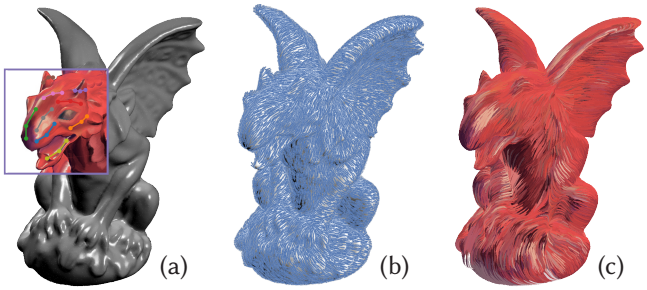


Figure 10: Partial exemplar for a complex scene. (a) We only use the exemplar and annotation given in the focused area. (b) Generated orientation field on the complex model. (c) Expressive strokes are synthesized over the entire region.

every frame, we have a stable animation (please see our supplementary video). This is because we first generate tentative stroke field even for the unseen region and then perform smoothing for the entire region. Even when the viewpoint moves to the back side of the object, for which seemingly information only in the front side has been provided through the exemplar, our generated animation is still plausible. Note the natural motion of the highlights as the viewpoint changes, as well as the expressive stroke styles for each frame.

Encouraged by the fact that our learned model can plausibly generate stroke field for the unseen area, we test another ambitious setting where the artist provides drawing only for a small fraction of the object (the focused area in Fig. 10 (a)). The decomposed style-specific elements and instance-intrinsic elements allow us to nicely propagate the stroke orientations, length, width, and colors.

Next, we show our application to a deforming object in Fig. 9 (bottom). The object starts in a spherical shape and gets stretched over time, resulting in the total surface area to vary as well. We can apply our method even to a complex, deforming character

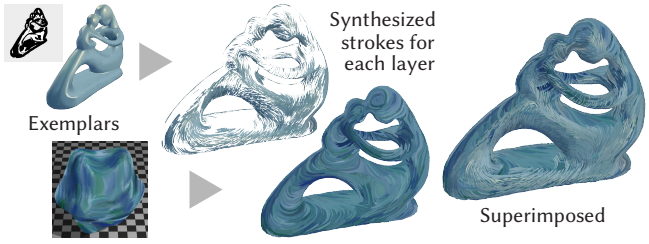


Figure 11: Superimposition of two styles.

animation shown in Fig. 1 with coherent and expressive stroke styles.

In Fig. 11, we transfer the stroke styles drawn for the genus-0 blobby object to the bottom layer of the genus-4 fertility statue. The transfer works for objects with different topology. We also superimpose another layer of strokes generated using a different style. For the upper layer, we have another transparency exemplar (applied via 1 nearest neighbor regression, like colors) for specifying which part of the style to apply. Because we are generating and registering strokes on the manifold, the animation is smooth even when the holes appear or disappear in the image space.

8 CONCLUSIONS AND FUTURE WORK

We have presented a method for example-based synthesis of animatable stroke styles. From the exemplars drawn by the artist, our method learns a model that takes a set of features in the proxy space as input and returns scalar attributes or weights for combining canonical sections for generating the orientations. The use of the proxy space decouples the dependency on style-specific elements from the instance-intrinsic elements, and the use of the overly determined bases enables the use of a very simple (in fact linear) regression. The generated stroke animations are more temporally stable and contain less artifacts compared to the patch-based approaches, while retaining the vivid and expressive stroke styles.

There are several fascinating future work (the details are in our supplementary material): 1) an automatic approach that can tell if an enough variation of samples are collected during the annotation process, 2) learning the *correlation* (with a generalized randomness) among neighboring strokes for the colors, lengths, widths, and orientations, 3) offering interactive artistic control, 4) incorporating transparent media with the consideration of removing already generated strokes as well as of their mixing, 5) handling atmospheric participating media, and 6) bridging more advanced geometry processing tools, such the N-RoSy field or topological optimization, for better control of the singular points or for incorporating more sophisticated view-dependent canonical sections based on, e.g., the suggestive contours [DeCarlo et al. 2003].

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful suggestions and discussions. This work was supported in part by a grant from JST FOREST Program, JPMJFR206R, Japan, a JSPS Grant-in-Aid for Scientific Research (A) 18H04106, Japan, and a JSPS Grant-in-Aid for Scientific Research (C) 22K12051, Japan.

REFERENCES

Katie Bassett, Ilya Baran, Johannes Schmid, Markus Gross, and Robert W. Sumner. 2013. Authoring and Animating Painterly Characters. *ACM Transactions on Graphics* 32, 5, Article 156 (oct 2013), 12 pages. https://doi.org/10.1145/2484238

Bill Baxter, Vincent Scheib, Ming C. Lin, and Dinesh Manocha. 2001. DAB: Interactive Haptic Painting with 3D Virtual Brushes. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, 461––468. https://doi.org/10.1145/383259.383313

Pierre Bénard, Forrester Cole, Michael Kass, Igor Mordatch, James Hegarty, Martin Sebastian Senn, Kurt Fleischer, Davide Pesare, and Katherine Breeden. 2013. Stylizing Animation by Example. *ACM Transactions on Graphics* 32, 4 (Proc. of SIGGRAPH 2013), Article 119 (jul 2013), 12 pages. https://doi.org/10.1145/2461912.2461929

John Bowers, Rui Wang, Li-Yi Wei, and David Maletz. 2010. Parallel Poisson Disk Sampling with Spectrum Analysis on Surfaces. *ACM Transactions on Graphics* 29, 6 (Proc. of SIGGRAPH ASIA 2010), Article 166 (dec 2010), 10 pages. https://doi.org/10.1145/1882261.1866188

Luis Cardona and Suguru Saito. 2015. Hybrid-Space Localized Stylization Method for View-Dependent Lines Extracted from 3D Models. In *Proceedings of the Workshop on Non-Photorealistic Animation and Rendering (NPAR '15)*. Eurographics Association, 79––89.

Guoning Chen, Vivek Kwatra, Li-Yi Wei, Charles D Hansen, and Eugene Zhang. 2011. Design of 2D Time-Varying Vector Fields. *IEEE Transactions on Visualization and Computer Graphics* 18, 10 (2011), 1717–1730.

Zhili Chen, Byungmoon Kim, Daichi Ito, and Huamin Wang. 2015. Wetbrush: GPU-Based 3D Painting Simulation at the Bristle Level. *ACM Transactions on Graphics* 34, 6 (Proc. of SIGGRAPH ASIA 2015), Article 200 (oct 2015), 11 pages. https://doi.org/10.1145/2816795.2818066

Nelson S.-H. Chu and Chiew-Lan Tai. 2005. MoXi: Real-Time Ink Dispersion in Absorbent Paper. *ACM Transactions on Graphics* 24, 3 (Proc. of SIGGRAPH 2005) (jul 2005), 504––511. https://doi.org/10.1145/1073204.1073221

Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. 2013. Digital Geometry Processing with Discrete Exterior Calculus. In *ACM SIGGRAPH 2013 courses* (Anaheim, California) *(SIGGRAPH '13)*. ACM, New York, NY, USA, 126 pages.

Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. 1997. Computer-Generated Watercolor. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., 421––430. https://doi.org/10.1145/258734.258896

Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. 2003. Suggestive Contours for Conveying Shape. *ACM Transactions on Graphics* 22, 3 (Proc. of SIGGRAPH 2003) (jul 2003), 848––855. https://doi.org/10.1145/882262.882354

Mathieu Desbrun, Eva Kanso, and Yiying Tong. 2006. Discrete Differential Forms for Computational Modeling. In *ACM SIGGRAPH 2006 Courses* (Boston, Massachusetts) *(SIGGRAPH '06)*. Association for Computing Machinery, New York, NY, USA, 39––54. https://doi.org/10.1145/1185657.1185665

Frédo Durand. 2002. An Invitation to Discuss Computer Depiction. In *Proceedings of the 2nd International Symposium on Non-Photorealistic Animation and Rendering* (Annecy, France) *(NPAR '02)*. Association for Computing Machinery, New York, NY, USA, 111––124. https://doi.org/10.1145/508530.508550

Matthew Fisher, Peter Schröder, Mathieu Desbrun, and Hugues Hoppe. 2007. Design of Tangent Vector Fields. *ACM Transactions on Graphics* 26, 3 (Proc. of SIGGRAPH 2007) (jul 2007), 56––65. https://doi.org/10.1145/1276377.1276447

Jakub Fišer, Ondřej Jamriška, Michal Lukáč, Eli Shechtman, Paul Asente, Jingwan Lu, and Daniel Sýkora. 2016. StyLit: Illumination-Guided Example-Based Stylization of 3D Renderings. *ACM Transactions on Graphics* 35, 4 (Proc. of SIGGRAPH 2016), Article 92 (jul 2016), 11 pages. https://doi.org/10.1145/2897824.2925948

Hongbo Fu, Yichen Wei, Chiew-Lan Tai, and Long Quan. 2007. Sketching Hairstyles. In *Proceedings of the 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM '07)*. Association for Computing Machinery, New York, NY, USA, 31––36. https://doi.org/10.1145/1384429.1384439

David Futschik, Michal Kučera, Michal Lukáč, Zhaowen Wang, Eli Shechtman, and Daniel Sýkora. 2021. STALP: Style Transfer with Auxiliary Limited Pairing. *Computer Graphics Forum* 40, 2 (Proc. of EUROGRAPHICS 2021) (2021), 563–573.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. (2016), 2414–2423. https://doi.org/10.1109/CVPR.2016.265

T. Haga, Henry Johan, and Tomoyuki Nishita. 2001. Animation Method for Pen-and-Ink Illustrations Using Stroke Coherency. In *Proc. of CAD & Graphics 2001*. 333–343.

Filip Hauptfleisch, Ondřej Texler, Aneta Texler, Jaroslav Křivánek, and Daniel Sýkora. 2020. StyleProp: Real-Time Example-Based Stylization of 3D Models. *Computer Graphics Forum* 39, 7 (Proc. of Pacific Graphics 20+21) (2020), 575–586.

Siddharth Hegde, Christos Gatzidis, and Feng Tian. 2013. Painterly Rendering Techniques: A State-of-The-Art Review of Current Approaches. *Computer Animation and Virtual Worlds* 24, 1 (2013), 43–64. https://doi.org/10.1002/cav.1435

Aaron Hertzmann. 2003. Tutorial: A Survey of Stroke-Based Rendering. *IEEE Computer Graphics and Applications* 23, 4 (jul 2003), 70––81. https://doi.org/10.1109/MCG.2003.1210867

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 (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 327––340. https://doi.org/10.1145/383259.383295

Anil Nirmal Hirani. 2003. *Discrete Exterior Calculus*. Ph. D. Dissertation. USA. Advisor(s) Marsden, Jerrold E. https://doi.org/10.7907/ZHY8-V329

Jin Huang, Zherong Pan, Guoning Chen, Wei Chen, and Hujun Bao. 2013. Image-Space Texture-Based Output-Coherent Surface Flow Visualization. *IEEE Transactions on Visualization and Computer Graphics* 19, 9 (2013), 1476–1487. https://doi.org/10.1109/TVCG.2013.62

Emmanuel Iarussi, David Bommes, and Adrien Bousseau. 2015. BendFields: Regularized Curvature Fields from Rough Concept Sketches. *ACM Transactions on Graphics* 34, 3, Article 24 (apr 2015), 16 pages. https://doi.org/10.1145/2710026

Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant Field-Aligned Meshes. *ACM Transactions on Graphics* 34, 6 (Proc. of SIGGRAPH ASIA 2015), Article 189 (nov 2015), 15 pages. https://doi.org/10.1145/2816795.2818078

Ondřej Jamriška, Šárka Sochorová, Ondřej Texler, Michal Lukáč, Jakub Fišer, Jingwan Lu, Eli Shechtman, and Daniel Sýkora. 2019. Stylizing Video by Example. *ACM Transactions on Graphics* 38, 4 (Proc. of SIGGRAPH 2019), Article 107 (aug 2019), 11 pages. https://doi.org/10.1145/3306346.3323006

Mizuki Kagaya, William Brendel, Qingqing Deng, Todd Kesterson, Sinisa Todorovic, Patrick J. Neill, and Eugene Zhang. 2011. Video Painting with Space-Time-Varying Style Parameters. *IEEE Transactions on Visualization and Computer Graphics* 17, 1 (2011), 74–87. https://doi.org/10.1109/TVCG.2010.25

Robert D. Kalnins, Philip L. Davidson, Lee Markosian, and Adam Finkelstein. 2003. Coherent Stylized Silhouettes. *ACM Transactions on Graphics* 22, 3 (Proc. of SIGGRAPH 2003) (jul 2003), 856––861. https://doi.org/10.1145/882262.882355

Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, and Adam Finkelstein. 2002. WYSIWYG NPR: Drawing Strokes Directly on 3D Models. *ACM Transactions on Graphics* 21, 3 (Proc. of SIGGRAPH 2002) (jul 2002), 755––762. https://doi.org/10.1145/566654.566648

Evangelos Kalogerakis, Derek Nowrouzezahrai, Simon Breslav, and Aaron Hertzmann. 2012. Learning Hatching for Pen-and-Ink Illustration of Surfaces. *ACM Trans. Graph.* 31, 1, Article 1 (jan 2012), 17 pages. https://doi.org/10.1145/2077341.2077342

George Katanics and Tasso Lappa. 2003. Deep Canvas: Integrating 3D Painting and Painterly Rendering. In *Theory and Practice of Non-Photorealistic Graphics: Algorithms, Methods, and Production Systems (ACM SIGGRAPH 2003 Course Notes)*.

Dorota Kobiela and Hugh Welchman. 2017. Loving Vincent. BreakThru Productions, Trademark Films.

Michael A. Kowalski, Lee Markosian, J. D. Northrup, Lubomir Bourdev, Ronen Barzel, Loring S. Holden, and John F. Hughes. 1999. Art-Based Rendering of Fur, Grass, and Trees. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 433––438. https://doi.org/10.1145/311535.311607

Julian Kratt, Ferdinand Eisenkeil, Marc Spicker, Yunhai Wang, Daniel Weiskopf, and Oliver Deussen. 2017. Structure-Aware Stylization of Mountainous Terrains. In *Proceedings of the Conference on Vision, Modeling and Visualization* (Bonn, Germany) *(VMV '17)*. Eurographics Association, 17––27. https://doi.org/10.2312/vmv.20171255

Paul Kubelka and Franz Munk. 1931. Ein Beitrag zur Optik der Farbanstriche. *Zeitschrift für Technishen Physik* 12 (1931), 593––601.

Bei-Bei Liu, Yan-Lin Weng, Jian-Nan Wang, and Yi-Ying Tong. 2013. Orientation Field Guided Texture Synthesis. *Journal of Computer Science and Technology* 28, 5 (2013), 827–835.

Jingwan Lu, Connelly Barnes, Stephen DiVerdi, and Adam Finkelstein. 2013. RealBrush: Painting with Examples of Physical Media. *ACM Transactions on Graphics* 32, 4 (Proc. of SIGGRAPH 2013), Article 117 (jul 2013), 12 pages. https://doi.org/10.1145/2461912.2461998

Lukasz Mackiewicz and Francho Melendez. 2016. Loving Vincent: Guiding Painters through 64.000 Frames. In *ACM SIGGRAPH 2016 Talks (SIGGRAPH '16)*. Association for Computing Machinery, Article 6, 2 pages. https://doi.org/10.1145/2897839.2927394

Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Loring S. Holden, J. D. Northrup, and John F. Hughes. 2000. Art-Based Rendering with Continuous Levels of Detail. In *Proceedings of the 1st International Symposium on Non-Photorealistic Animation and Rendering (NPAR '00)*. Association for Computing Machinery, 59––66. https://doi.org/10.1145/340916.340924

Barbara J. Meier. 1996. Painterly Rendering for Animation. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. Association for Computing Machinery, 477––484. https://doi.org/10.1145/237170.237288

J. D. Northrup and Lee Markosian. 2000. Artistic Silhouettes: A Hybrid Approach. In *Proceedings of the 1st International Symposium on Non-Photorealistic Animation and Rendering (NPAR '00)*. Association for Computing Machinery, 31––37. https://doi.org/10.1145/340916.340920

Nicolas Ray and Dmitry Sokolov. 2014. Robust Polylines Tracing for N-Symmetry Direction Field on Triangulated Surfaces. *ACM Transactions on Graphics* 33, 3,

Article 30 (may 2014), 11 pages. https://doi.org/10.1145/2602145

Dave Rudolf, David Mould, and Eric Neufeld. 2003. Simulating Wax Crayons. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications (PG '03)*. IEEE Computer Society, 163–172. https://doi.org/10.5555/946250.946956

Johannes Schmid, Martin Sebastian Senn, Markus Gross, and Robert W. Sumner. 2011. OverCoat: An Implicit Canvas for 3D Painting. *ACM Transactions on Graphics* 30, 4 (Proc. of SIGGRAPH 2011), Article 28 (jul 2011), 10 pages. https://doi.org/10.1145/2010324.1964923

Mayank Singh and Scott Schaefer. 2010. Suggestive Hatching. In *Proceedings of the Sixth International Conference on Computational Aesthetics in Graphics, Visualization and Imaging* (London, United Kingdom) *(Computational Aesthetics'10)*. Eurographics Association, Goslar, DEU, 25—-32.

Peter-Pike J. Sloan, William Martin, Amy Ashurst Gooch, and Bruce Gooch. 2001. The Lit Sphere: A Model for Capturing NPR Shading from Art. In *Proceedings of Graphics Interface 2001* (Ottawa, Ontario, Canada) *(GI '01)*. Canadian Information Processing Society, CAN, 143—-150.

Benjamin Spencer, Robert S. Laramee, Guoning Chen, and Eugene Zhang. 2009. Evenly Spaced Streamlines for Surfaces: An Image-Based Approach. *Computer Graphics Forum* 28, 6 (2009), 1618–1631. https://doi.org/10.1111/j.1467-8659.2009.01352.x

Daniel Sýkora, Ondřej Jamriška, Ondřej Texler, Jakub Fišer, Michal Lukáč, Jingwan Lu, and Eli Shechtman. 2019. StyleBlit: Fast Example-Based Stylization with Local Guidance. *Computer Graphics Forum* 38, 2 (Proc. of EUGRAPHICS 2019) (2019), 83–91.

Isao Takahata. 2014. The Tale of The Princess Kaguya. Studio Ghibli.

Jianchao Tan, Marek Dvorožňák, Daniel Sýkora, and Yotam Gingold. 2015. Decomposing Time-Lapse Paintings into Layers. *ACM Transactions on Graphics* 34, 4 (Proc. of SIGGRAPH 2015), Article 61 (aug 2015), 10 pages. https://doi.org/10.1145/2766960

Ondřej Texler, David Futschik, Michal kučera, Ondřej jamriška, Šárka Sochorová, Menclei Chai, Sergey Tulyakov, and Daniel SÝkora. 2020. Interactive Video Stylization Using Few-Shot Patch-Based Training. *ACM Transactions on Graphics* 39, 4 (Proc. of SIGGRAPH 2020), Article 73 (jul 2020), 11 pages. https://doi.org/10.1145/3386569.3392453

Greg Turk. 2001. Texture Synthesis on Surfaces. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 347––354. https://doi.org/10.1145/383259.383297

Hassler Whitney. 1957. *Geometric Integration Theory*. Princeton University Press.

Kai Xu, Daniel Cohen-Or, Tao Ju, Ligang Liu, Hao Zhang, Shizhe Zhou, and Yueshan Xiong. 2009. Feature-Aligned Shape Texturing. *ACM Transactions on Graphics* 28, 5 (Proc. of SIGGRAPH ASIA 2009) (dec 2009), 1—-7. https://doi.org/10.1145/1618452.1618454

Yonghao Yue, Breannan Smith, Christopher Batty, Changxi Zheng, and Eitan Grinspun. 2015. Continuum Foam: A Material Point Method for Shear-Dependent Flows. *ACM Transactions on Graphics* 34, 5 (2015), 160:1–20.

Eugene Zhang, James Hays, and Greg Turk. 2007. Interactive Tensor Field Design and Visualization on Surfaces. *IEEE Transactions on Visualization and Computer Graphics* 13, 1 (2007), 94–107. https://doi.org/10.1109/TVCG.2007.16

Eugene Zhang, Konstantin Mischaikow, and Greg Turk. 2006. Vector Field Design on Surfaces. *ACM Transactions on Graphics* 25, 4 (oct 2006), 1294––1326. https://doi.org/10.1145/1183287.1183290

Ming Zheng, Antoine Milliez, Markus Gross, and Robert W. Sumner. 2017. Example-Based Brushes for Coherent Stylized Renderings. In *Proceedings of NPAR'17*. 3:1–3:10. https://doi.org/10.1145/3092919.3092929 Los Angeles, CA, USA, July 28-29.