

Video Painting with Space-Time-Varying Style Parameters

Mizuki Kagaya, William Brendel, Qingqing Deng, Todd Kesterson, Sinisa Todorovic, Patrick J. Neill, and Eugene Zhang, *Member, IEEE Computer Society*

Abstract—Artists use different means of stylization to control the focus on different objects in the scene. This allows them to portray complex meaning and achieve certain artistic effects. Most prior work on painterly rendering of videos, however, uses only a single painting style, with fixed global parameters, irrespective of objects and their layout in the images. This often leads to inadequate artistic control. Moreover, brush stroke orientation is typically assumed to follow an everywhere continuous directional field. In this paper, we propose a video painting system that accounts for the spatial support of objects in the images or videos, and uses this information to specify style parameters and stroke orientation for painterly rendering. Since objects occupy distinct image locations and move relatively smoothly from one video frame to another, our object-based painterly rendering approach is characterized by style parameters that coherently vary in space and time. Space-time-varying style parameters enable more artistic freedom, such as emphasis/de-emphasis, increase or decrease of contrast, exaggeration or abstraction of different objects in the scene in a temporally coherent fashion.

Index Terms—Nonphotorealistic rendering, video painting, multistyle painting, tensor field design.

1 INTRODUCTION

PAINTERLY rendering of images and videos has received much attention in the past two decades. In this paper, we describe a video painting framework in which style parameters as well as brush stroke orientations can be specified individually for each region (object or background) in some keyframes and propagated to other frames in a temporally coherent fashion. We will refer to this as the problem of *multistyle painterly rendering*, or simply *multistyle painting*. There are a number of benefits in multistyle painting:

1. By spatially varying style parameters such as brush stroke length, width, and opacity, the artist has the freedom to emphasize or de-emphasize certain objects in the scene (Fig. 1c: flower (emphasized) and leaves (de-emphasized)), to control the level of abstraction or realism in the resulting painting (Figs. 2d and 2e), to increase or decrease contrasts between neighboring objects (Fig. 3c: contrast between red/orange peppers and greenish peppers is increased), to exaggerate or trivialize, and certain cinematographic concepts such as calmness or stress.
2. By temporally varying style parameters based on the objects, the artist can maintain the aforementioned

coherence and control, or modify the parameters, such as achieving a rack focus effect (Fig. 9).

3. By allowing region-based brush stroke orientation design, the user has additional tools to achieve the aforementioned effects as well as to create illusions (Fig. 4).

Despite the potential benefits enabled by multistyle painting, there has been relatively little research in this area. Most existing work in image and video painting has focused on mimicking a single painting style where the style parameters such as brush stroke length, width, and opacity are constant over the whole space-time domain [1], [2], [3], [4], [5], [6], [7], [8]. While there has been some work that can enable certain region-based nonphotorealistic effects [9], [10], their focus is typically on automatic stylization for some particular effect. Consequently, there is relatively little exploration on the artistic design of multistyle effects once the segmentation is available. Furthermore, there is little work in systematic design of brush orientations in video painting [11].

In this paper, we address these issues by providing a painterly rendering framework for videos. In this framework, the video is first segmented into regions (objects or background) in a temporally coherent fashion using an existing technique [12]. Next, the user defines style parameters and brush stroke orientation for desired objects by specifying them in some *keyframes*. These user specifications are then automatically propagated to the other frames as well as regions without any specifications through constrained optimization. Lastly, the style parameters and stroke orientations will be used to generate the final rendering.

Our approach facilitates interactive artistic design and experiments once the segmentation becomes available. Style parameters can be specified in a small number of objects in some keyframes and automatically propagated to the whole video. In addition, as part of our approach, we have

- M. Kagaya, W. Brendel, Q. Deng, S. Todorovic, and E. Zhang are with the School of Electrical Engineering and Computer Science, 1148 Kelley Engineering Center, Oregon State University, Corvallis, OR 97331. E-mail: {kagaya, brendel, dengqi, sinisa, zhang}@eeecs.oregonstate.edu.
- T. Kesterson is with the New Media Communications, 207 Gilkey Hall, Oregon State University, Corvallis, OR 97331. E-mail: dengqi@eeecs.oregonstate.edu.
- P.J. Neill is with NVidia, 2701 San Tomas Expressway, Santa Clara, CA 95050. E-mail: pneill@nvidia.com.

Manuscript received 23 June 2009; revised 17 Aug. 2009; accepted 26 Aug. 2009; published online 28 Jan. 2010.

Recommended for acceptance by M. Chen.

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-2009-06-0126. Digital Object Identifier no. 10.1109/TVCG.2010.25.

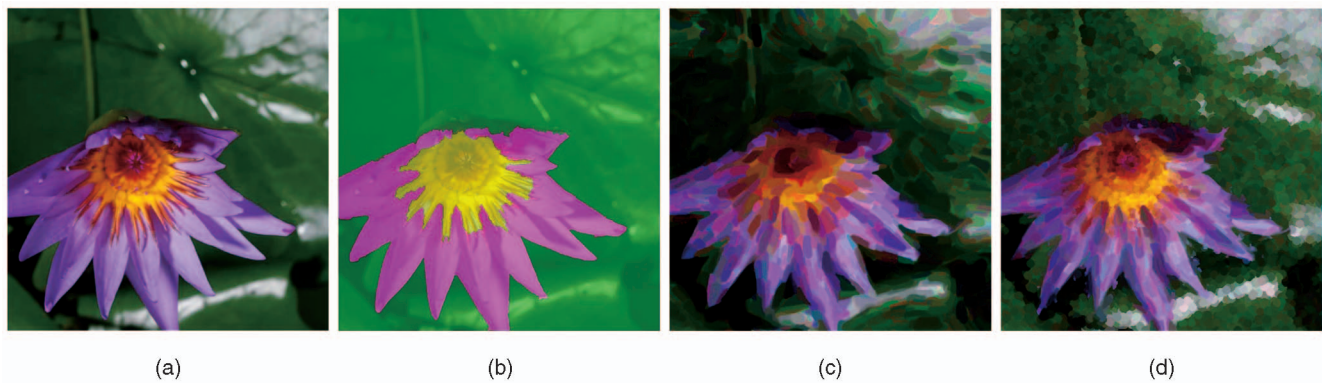


Fig. 1. Two stylized frames from a video showing a blooming flower: (a) the original frame, (b) the segmentation of the frame into three parts: petals, leaves, and stamens, (c) a single van Gogh style is applied to the entire frame, and (d) the same van Gogh style applied to the petals, whereas spatially varying Pointillist settings are applied to the leaves and stamens with different stroke diameters. The use of different styles, and spatially varying style parameters results in de-emphasizing the leaves, portraying the realistic fine granularity of the stamens, and enhancing overall contrast among the three regions. This is possible with the ability to produce temporally coherent segmentation.

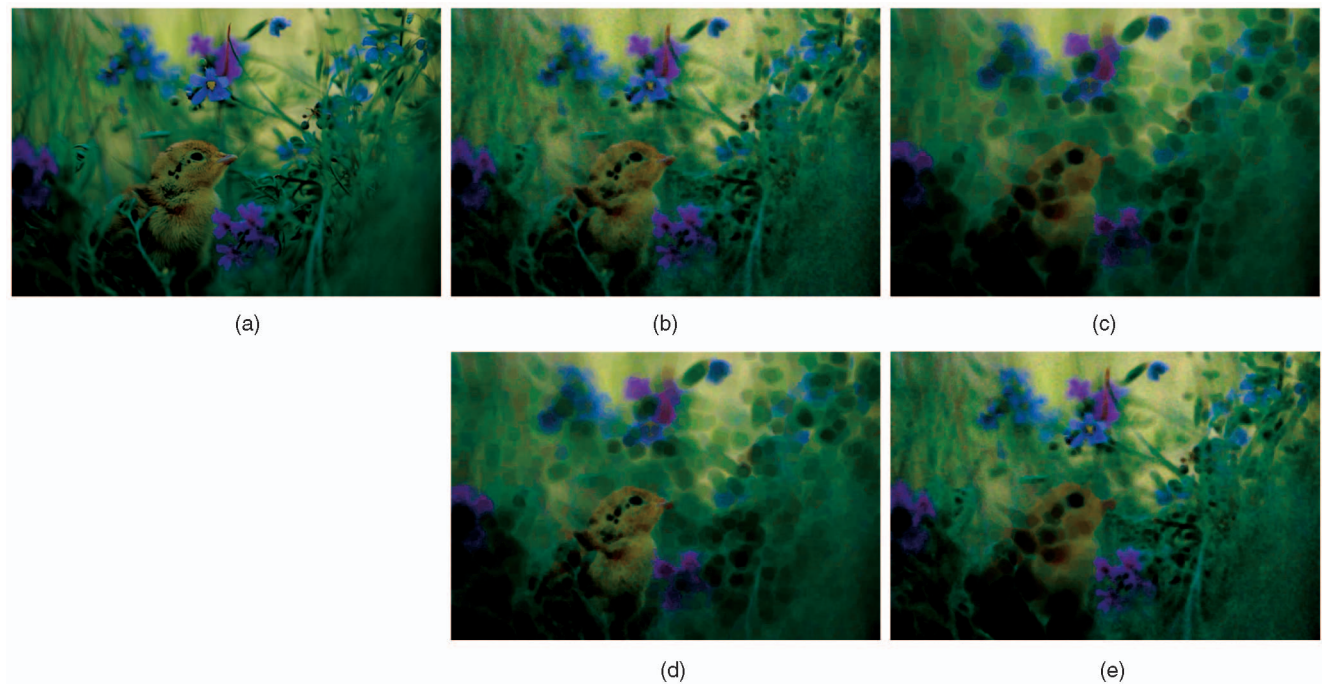


Fig. 2. An image of a bird (a) is painted in constant style parameter values (b-c) and spatially varying values (d-e). The difference between (b) and (c) is the size of the strokes. In (b), details are preserved while in (c), the image is abstracted uniformly. With a binary segmentation (bird and background), the artist controls the amount of abstraction as well as the focus in the painting results (d: focus on the bird; e: emphasis given to the background flowers).

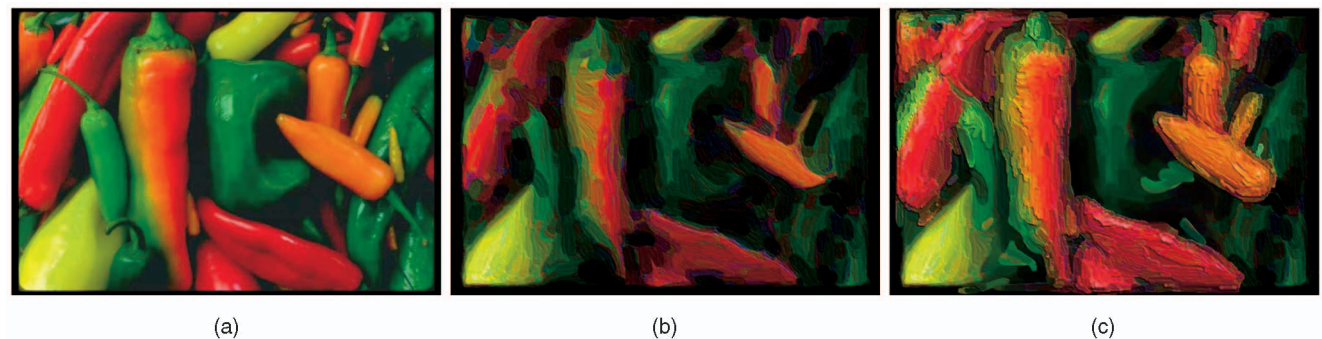


Fig. 3. This figure illustrates the control over the increase or decrease in contrast with spatially varying style parameters. Given an image of peppers of various colors (a), two paintings are produced: (b) single style, and (c) multistyle. Notice that the painting in (c) is obtained from (b) when increasing brightness and the amount of the detail to red and orange peppers *only*.

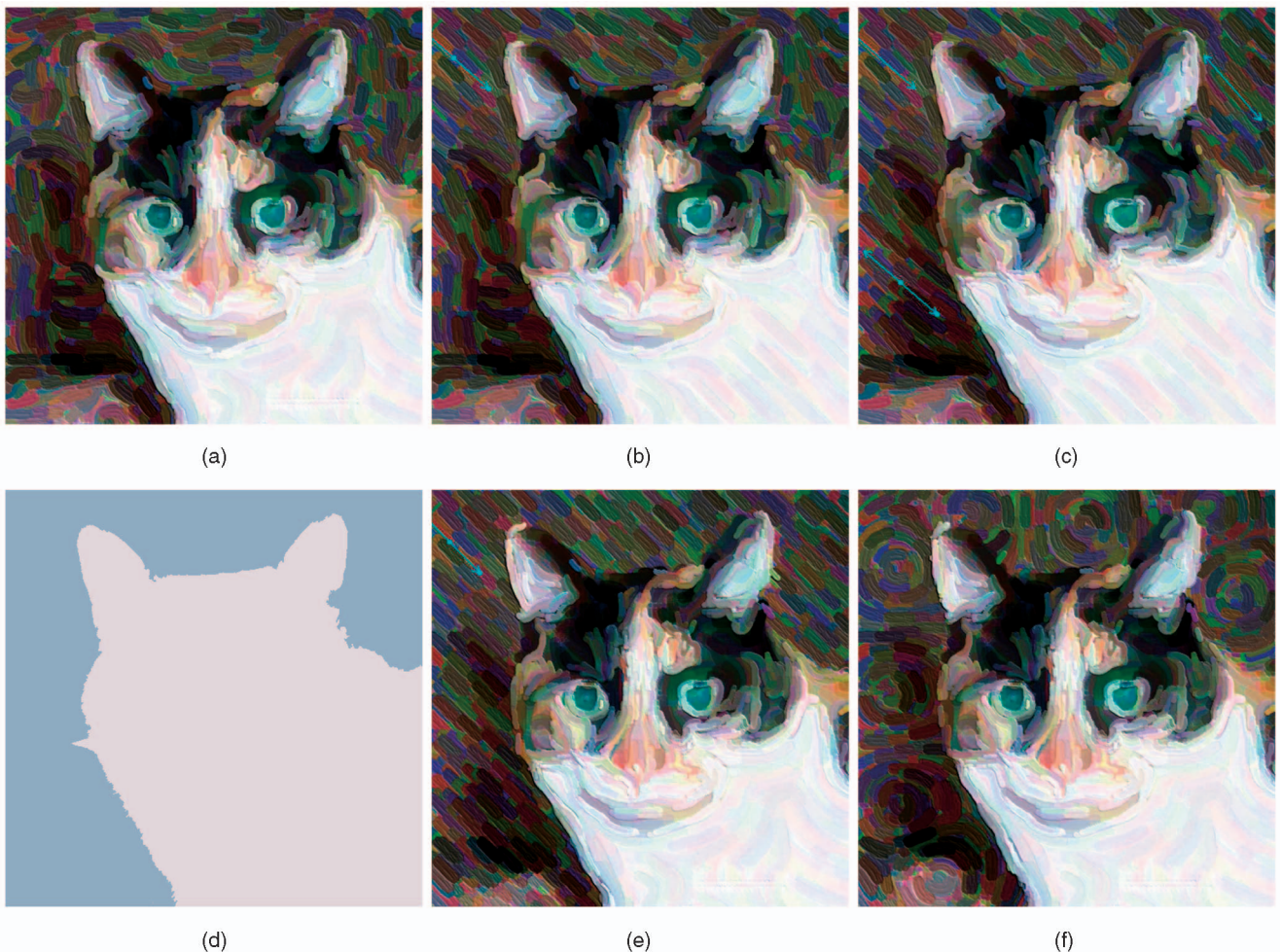


Fig. 4. This figure illustrates the importance of object-based stroke orientation design. Given an input of a cat, brush stroke orientation is based on the strong edges in the image (a). The user desires to assign a constant directional field for the background to achieve contrast between the background and the cat. Without segmentation, a single user-specified constraint (colored arrow in (b)) can only help achieve the artistic goal in a very local region (b). By adding more constraints in (c), the stroke orientations in the background start to conform. However, this has the side effect that these constraints also impact the orientation fields in the cat (white regions on her body). With the segmentation (d), the user can easily achieve the artistic goal with one constraint without modifying the stroke orientations inside the cat (e). Notice the stroke orientations in (e) is the same as in (a). In (f), hallucinational circular patterns are added to the background, again, without impacting the stroke orientations in the cat.

developed an *image-based* painterly rendering algorithm that can lead to more coherent results than existing geometry-based methods in which curved brush strokes are explicitly defined. Moreover, we make use of the color of strokes to sort brush strokes, which has resulted in strokes with softer edges and partially alleviated the flickering issue in the video setting. Finally, we allow the user to easily design brush stroke orientations in a video, which, to the best of our knowledge, is the first of its kind. Note that the brush stroke orientations can be discontinuous across region boundaries, which is often desirable but has not been supported in past stroke orientation field design systems such as [13], [6], [11].

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 describes the style and orientation design capabilities of our system and Section 4 explains our image-based multistyle renderer for images and videos. In Section 5, we describe how we generate a spatiotemporal video segmentation. We present results in Section 6, and summarize and discuss some future work in Section 7.

2 RELATED WORK

We will review two areas that are most relevant to the contributions of this paper: painterly rendering and flow visualization.

2.1 Painterly Rendering

Painterly rendering has been a well-researched area in computer graphics. To review all the work is beyond the scope of this paper. We focus on the most relevant work here.

Haeberli [1] introduces the idea of painterly rendering as a form of visual representation of objects. Litwinowicz [2] presents a system in which Impressionistic effects can be created for images and videos based on user-specified stroke parameters. Hertzmann [13] describes an automatic algorithm that generates a number of styles of painting effects with various stroke sizes. A more physically realistic effect by adding height fields onto brush strokes is used to add shading effect to the strokes [5]. Hertzmann [3] describes another painterly rendering method by formulating brush stroke placement as an energy optimization

problem. Brush strokes can be added and removed as the optimization process iterates. In a rather different approach, Hertzmann et al. [14] augment images with painterly appearance using texture synthesis techniques.

Various techniques can be presented to produce video painting. Hertzmann and Perlin [7] extend the multilayer approach of Hertzmann [13] to videos. In this approach, brush strokes from the previous frame will be reused in the next frame after being moved according to the optical flow. Klein et al. [15] extend the notion of brush strokes to tubes in 3D space-time, leading to more coherent but sometime undesirable appearances. Hays and Essa [6] provide a high-quality painterly renderer for videos by reusing the idea of optical flow and by restraining the speed of brush stroke rotation. DeCarlo and Santella propose a framework for abstracting a video [4], which is later used to produce video watercolorization [8]. In addition, Meier produces coherent rendering results for synthetic 3D scenes using a combination of geometry and image-based techniques [16]. Cunzi et al. [17] address the “shower door” effect in 3D virtual walk-through by moving the background in a plausible manner. These techniques assume a single set of style parameters while our approach assumes space-time-varying style parameters.

All of this work focuses on the rendering aspect of the painting. In contrast, we wish to provide a system that facilitates and supports the design and rendering process with varying style parameters. Hertzmann’s work [13], [3] in providing level-of-detail control to the user is inspirational to this paper. However, we provide a more systematic approach that includes an effective segmentation tool and a user interface for varying style parameter design. Moreover, in this paper, we introduce the idea of creating a time-dependent tensor field to guide brush stroke orientations in a video. Past work [18], [11] applies tensor field design to image painting only. Based on our literature review, our system is the first that enables the time-dependent design capability.

2.2 Flow Visualization

In this paper, we present a painterly renderer that is inspired by flow visualization techniques (Section 4). To review all related work in flow visualization is beyond the scope of this paper, and we will only mention the most relevant work.

An efficient way of visualizing a vector field is by showing a set of streamlines, i.e., curves that are tangent to the vector field everywhere along their paths. Cabral and Leedom [19] present one such technique by performing *line integral convolution* (LIC) on an initial texture of white noise according to the vector field. For every pixel, they assign a value by extracting the streamline that contain the pixel and computing the average intensity of the initial texture along the streamline. The approach results in a high-quality continuous representation of the vector field. However, it is computationally expensive since it requires tracing a streamline for every pixel. Later, Stalling and Hege describe a faster way of creating LIC images by reducing the number of streamlines that need to be traced (FastLIC) [20]. Van Wijk [21] develops an interactive and high-quality image-based flow visualization technique (IBFV) for planar vector

fields. IBFV enables interactive display of vector fields with the assistance of graphics hardware.

Zhang et al. [11] demonstrate that the edge field extracted from an image is a tensor field. They also point out that treating a tensor field as a vector field by simply removing the ambiguity in the orientation causes visual artifacts. Delmarcelle and Hesselink [22] propose to visualize 2D or 3D tensor fields with hyperstreamlines, which are curves that tangent to the eigenvectors of the tensor fields everywhere along their paths. Zheng and Pang [23] propose a tensor field visualization technique that they call *HyperLIC*. This method makes use of LIC to produce images that resemble visualizations based on hyperstreamlines. Zhang et al. [11] adapt the IBFV approach of van Wijk to tensor fields, which results in an interactive and high-quality tensor field visualization.

Our rendering pipeline adopts the approach of Zhang et al. [11]. However, the application requirement is very different. We will provide details about our advection algorithm in more detail, in Section 4.

3 STYLE AND ORIENTATION FIELD DESIGN

Our multistyle rendering framework consists of three stages:

1. Video analysis and segmentation,
2. Segmentation-enabled style and orientation design, and
3. Rendering.

In the first stage, the input video V is analyzed to compute the following information: the per-frame edge field E and optical flow field f as well as a temporally-coherent segmentation of V into regions R_i ($1 \leq i \leq n$) in the space-time domain.

In the second stage, the user specifies style parameters and designs stroke orientations based on the segmentation. This results in a number of scalar fields s_j ($1 \leq j \leq m$), each of which corresponds to a style parameter such as stroke width or opacity, as well as a tensor field T that guides brush stroke orientations in the video.

In the last step, an output video V' is produced by our painterly rendering algorithm that makes use of the segmentation, the style parameters, the optical flow, as well as the tensor field.

The contributions of this work lie in the second and third stages, which we describe in detail in this section and Section 4, respectively. In Section 5, we will mention existing techniques that we employ for video analysis, such as the segmentation method of Brendel and Todorovic [12].

In the second stage, the user can interactively specify desired style parameters as well as brush stroke orientations. These parameters will then be used in the final stage of rendering the output video (Section 4). The style parameters that our system supports include the following, most of which are inspired by the work of Haeberli [1] and Hertzmann [13].

- *Stroke size*: describes the diameter of a stroke for Pointillism styles and the width of the stroke for other styles such as Impressionism, Expressionism, van Gogh, and Colorist Wash.

- *Stroke length*: is the desirable brush stroke length which allows long, curved brush strokes and can be used to illustrate natural anisotropy in the images such as silhouettes.
- *Stroke opacity*: controls the amount of transparency or *hardness* in a brush stroke and is useful to achieve styles such as the Colorist Wash.
- *Color shift*: describes changes in colors of all brush strokes in the video or a region. It consists of three parameters that describes the shift in hue, saturation, and value, respectively. This parameter can be used to alter the overall tone of the image (Fig. 8).
- *Color jittering*: controls the maximal amount of jittering in the color of each brush stroke. When this parameter is zero, the color of each brush stroke is taken from the pixel in the input video corresponding to the seed location of the stroke. Having a nonzero value allows greater variation and thus the contrast between neighboring brush strokes, which is desirable in Expressionism and the van Gogh style [6].
- *In-stroke texture height and density*: allows a stroke to be textured, as in [5].

In addition to these art-driven parameters, we also support a number of algorithm-specific parameters such as:

- *Number of layers*: which allows layers of brush strokes of different sizes to be overlaid [13].
- *Error threshold*: is used to decide where on the canvas additional strokes are needed [13].

Our system assumes that style parameters are defined at each pixel.

The *orientation* of strokes is an important ingredient in conveying features with directions, such as the silhouette and motions of objects in the scene. Unlike style parameters, the orientation field is not a scalar field and contains directional information. We will discuss the design of scalar-valued style parameters and the orientation field in Sections 3.1 and 3.2, respectively.

Fig. 5 shows the interface of our system. The window consists of four parts: a canvas (upper-left), a control panel (right), a frame browser (middle-left), and a system message box (lower-left). The canvas can be used to show the input (image or video), the segmentation, the visualization of the orientation field, and the painterly result. When the input has been segmented, the user can select any region in the segmentation by clicking on a pixel belonging to that region. When this happens, other regions will be dimmed while the selected region remains with its original colors. Clicking on a selected region will unselect it and undo the dimming of other regions. The user can then use the control panel to assign style parameters or design orientation fields for the selected region. For convenience, we have provided some default styles such as van Gogh, Colorist Wash, Impressionism, Pointillism, and Watercolor. The user can simply apply any of these styles to the designed object by clicking the corresponding button. The style parameters can be modified at any moment and the corresponding result will be shown momentarily. In addition, the user can design brush stroke orientation using the control on the tab titled “Field Design/Display” (not shown in Fig. 5). Note that the controls for

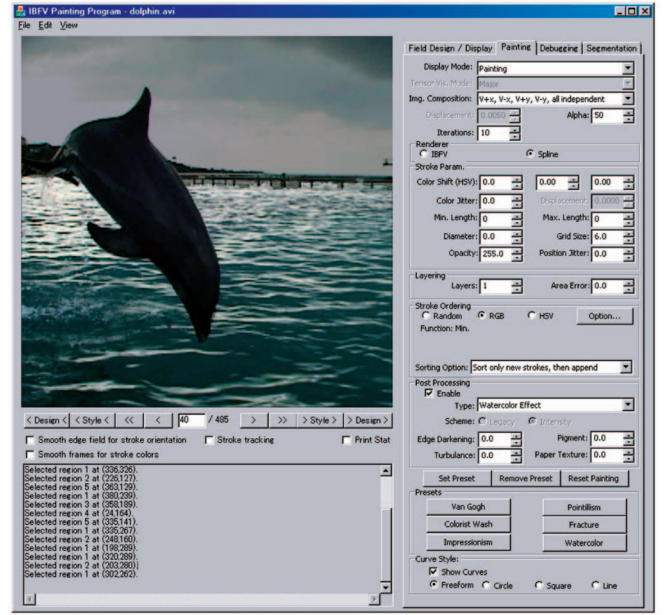


Fig. 5. This figure shows the user interface of our system.

field design is a straightforward adaption from the tensor field design system of Zhang et al. [11]. The user can use the frame browser to jump to any frame or the immediate previous or next keyframe for any given object, with either style parameters or stroke orientations. When this happens, any selected region will remain selected. Finally, the message box is used to provide feedback to the user with useful information. Next, we will describe the mathematical framework behind our design functionalities.

3.1 Style Parameter Design and Propagation

We will first discuss the design of spatially varying style parameters in an image or a frame of a video.

Given an input image I and its segmentation $\mathcal{R} = \{R_1, R_2, \dots, R_n\}$, our system allows a user to select a region R_i ($1 \leq i \leq n$) and associate a set of style parameters $\mathbb{S} = \{s_1, s_2, \dots, s_m\}$ to it. The style parameters are then assigned to every pixel in R_i . We refer to R_i as a *key region*. Regions not receiving any specification are referred to as *free regions*. Our system automatically assigns values to *free regions* by propagating the parameters from the key regions through the following relaxation process. Let R_j be a free region and \mathbf{p} a pixel in R_j . Also, let $s_k \in \mathbb{S}$ be the k th style parameter. Then,

$$s_k(\mathbf{p}) = \sum_r \omega_r s_k(\mathbf{p}_r), \quad (1)$$

where \mathbf{p}_r s are neighboring pixels of \mathbf{p} in the image, and $\omega_r \geq 0$ is the weight of the r th neighboring pixel satisfying $\sum_r \omega_r = 1$. In our setting, we only consider four neighboring pixels and choose $\omega_r = \frac{1}{4}$. When \mathbf{p} is on the boundary of the canvas we adjust the weights accordingly. The collection of these equations for pixels in the free regions gives rise to a system of discrete *Laplace equations* for which the parameter values of the pixels in the key regions serve as the boundary conditions. The discrete Laplace equations result in a sparse

linear system which can be solved efficiently using a multigrid Gauss-Seidel method [24].

Given a video V and a temporally coherent video object segmentation $\mathcal{R} = R_1, R_2, \dots, R_m$, our system allows the user to specify desired style parameters for a region $R_i \in \mathcal{R}$ in a set of keyframes $F_{i,j} : 1 \leq j \leq T(i)$, where $T(i)$ is the number of keyframes for region R_i . Note that $T(i)$ may vary for different regions R_i . Also, two regions R_{i_1} and R_{i_2} may not occur in the same keyframe. The specified values are automatically propagated to the rest of the video, i.e., to all pixels that were not given a style. Similar to the case of images, this can be achieved by using the Laplace equation as follows:

$$s_k(\mathbf{p}^t) = \sum_r \omega_r s_k(\mathbf{p}_r) + \omega_f s_k(\mathcal{F}(\mathbf{p}^t)) + \omega_b s_k(\mathcal{F}^{-1}(\mathbf{p}^t)), \quad (2)$$

where $\mathcal{F}(\mathbf{p}^t)$ and $\mathcal{F}^{-1}(\mathbf{p}^t)$ are the images of \mathbf{p}^t under the forward and backward optical flows, respectively. The total weight $\sum_r \omega_r + \omega_f + \omega_b = 1$. However, this formulation requires the equation be solved over the 3D domain of $M \times N \times L$, where M , N , and L are the number of rows, columns, and frames, respectively. While it is possible to solve this equation, doing so can reduce the interactivity that we wish to provide to the users. Instead of solving with the most general formulation given by (2), we break the problem down into two subproblems. First, given region R_i and a set of corresponding keyframes, how do we estimate the style parameters of the pixels in R_i in a nonkeyframe? Second, how do we estimate the style parameters for a pixel in region R_i when there are no keyframes, i.e., $T(i) = 0$? To address these questions, we consider the following two-step approach:

1. Given a region R_i and a frame number j , we locate the closest previous and next keyframes involving R_i . If both exist, then we assign style parameters to R_i at frame j by performing an interpolation between the style parameters of the two enveloping keyframes. The interpolation function $f(t)$ can be linear or nonlinear. Nonlinear interpolation schemes allow the user to control the speed of transition between styles, which can be used to generate the effects such as calmness and stress. If only one exists, then we copy the style parameters from the keyframe. In the special case that j is a keyframe for region R_i , no action is needed. This step ensures that the user specifications are continuously propagated from keyframes to the entire video.
2. We resolve style parameters of regions for which no keyframe was specified. Note that after the first step, there is at least one region in each frame that has well-defined style parameters. These values are simply propagated to the rest of the pixels in a frame by reusing (1).

This two-step approach does not explicitly take into account the optical flow since the relaxation is done independently per frame. However, assuming that underlying video segmentation is sufficiently temporally coherent, the aforementioned approach approximates well the optimal solution.

3.2 Stroke Orientation Field Design and Propagation

Stroke orientation is an important parameter for providing artistic control. Zhang et al. [11] describe a tensor field design system for the purpose of guiding stroke orientation for image painting. The tensor field can be created from scratch or by modifying an existing field extracted from the image. User specifications are converted into basis tensor fields and summed. While this system has shown great promise in allowing stroke orientation control, it is inefficient for our purposes for two reasons. First, it is designed to work with only images and does not generalize to video in a straightforward fashion. Second, it assumes that the whole image is a single region and always generates a continuous tensor field. However, maintaining discontinuity in stroke orientations across object boundaries is often desirable. Despite these limitations, we feel that the idea of basis tensor fields is intuitive and effective. Consequently, we adapt this approach to construct a tensor field in a segmented video.

We first review relevant concepts of tensor fields and the idea of tensor field design by combining basis tensor fields. A *second-order tensor* T is a 2×2 matrix

$$\begin{pmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{pmatrix}.$$

T is *symmetric* if $t_{12} = t_{21}$ and *traceless* if $t_{11} + t_{22} = 0$. In this paper, we consider second-order symmetric, traceless tensors, referred to as tensors in the sequel, for simplicity. Such a tensor has the following form:

$$T = \begin{pmatrix} u & v \\ v & -u \end{pmatrix} = \rho \begin{pmatrix} \cos \theta & \sin \theta \\ \sin \theta & -\cos \theta \end{pmatrix}, \quad (3)$$

where $\rho = \sqrt{t_{11}^2 + t_{12}^2}$ and $\theta = \tan^{-1}(\frac{t_{12}}{t_{11}})$. T is *degenerate* when $\rho = 0$. A nondegenerate tensor has two eigenvalues $\pm\rho$ and two families of mutually perpendicular eigenvectors corresponding to the eigenvalues. Zhang et al. [11] use the eigenvectors corresponding to the eigenvalue $+\rho$ to guide the stroke orientations. We will follow this convention.

A *tensor field* is a continuous tensor-valued function. Useful tensor patterns for painterly rendering include regular patterns in which the eigenvector directions are the same everywhere and singular patterns such as *wedges*, *trisectors*, *nodes*, and *centers* [11]. A basis tensor field corresponds to a pattern that can be specified by the following radial basis function:

$$T_i(x, y) = e^{-d\sqrt{(x-x_i)^2+(y-y_i)^2}} M(x - x_i, y - y_i) \quad (4)$$

in which d is a decay constant, (x_i, y_i) is the center of the basis function, and $M(s, t)$ is a function describing the tensor patterns. For example, a regular pattern can be characterized by a constant function

$$M = \begin{pmatrix} u_0 & v_0 \\ v_0 & -u_0 \end{pmatrix}$$

while a wedge can be characterized by

$$M(s, t) = \begin{pmatrix} s & t \\ t & -s \end{pmatrix}.$$

In the framework proposed in [11], a tensor field is generated by converting each user specification of a regular or singular pattern into a basis field and combining them. This framework has proven both fast and intuitive. In this paper, we adopt the same framework and adapt it to segmented videos.

Given an image I with segmentation $\mathcal{R} = R_1, R_2, \dots, R_n$, a region R_i is a *key region* with respect to orientation if the brush stroke orientation in R_i is either specified by the user or automatically extracted from the image such as the edge field (Section 5). Otherwise, R_i is a *free region*.

For a region R_i , the brush stroke orientation field is generated as follows: $P_i(\mathbf{p}) = w_i E(\mathbf{p}) + (1 - w_i) \sum_j M_{i,j}(\mathbf{p})$, in which E is the edge field extracted from the image, $M_{i,j}$ is the basis field corresponding to a user specification (design element [11]), and $w_i \in [0, 1]$ is the weight constant assigned to the region. Notice that the region-based field design approach affords greater freedom in generating desired stroke orientations as the user need not be concerned with the impact of a user specification on the pixels not in the intended region. In contrast, the single-region field design method often requires the user to provide additional specifications just to ensure that boundaries between regions are preserved by the field. Note that the center of a design element need not be inside the intended region. This is useful when an object is broken into several connected components such as the sun is partially occluded by the clouds (Fig. 8).

For regions where no user specification is given, our system assigns them values using the same framework in which style parameters are propagated (Section 3.1). Notice that in this case, we do not have a scalar field that represents some style parameter. Instead, we need to propagate directional information into free regions. Recall that the directional information is encoded by a tensor field, i.e., its major eigenvectors. Zhang et al. [11] have demonstrated that such a propagation should be done by solving a pair of Laplace equations similar to (1). Basically, we treat the entries in the tensor field as independent style parameters and reuse the multigrid solver to propagate the tensor field to unknown regions. Note that for the type of tensors that we are concerned with, there are only two independent variables in (3): $u = \rho \cos \theta$ and $v = \rho \sin \theta$. Consequently, the amount of time to propagate stroke orientations is roughly twice the time for a style parameter.

Designing a tensor field for a video poses another challenge. In this setting, a set of constraints have been specified in different keyframes, and we wish to propagate them to the whole video. Similar to style parameters such as stroke length and opacity, we first propagate the basis tensor fields from keyframes to nonkeyframes for regions where some user specifications exist. The only complication here is that we need the ability to update the center location (x_i, y_i) and pattern function $M_{i,j}(x, y)$ of a user specification (4) from a keyframe to a nonkeyframe. This is necessary as the object such as a dolphin can jump and spin, and the user specification for one frame will not be able to achieve its original intention without accounting for the motion of the object. Note that we do not use optical flow for this purpose due to the noise and degeneracy often associated with its estimation. Instead, we obtain the center and orientation of an

object by computing its best fitting ellipse through linear PCA analysis [25]. This allows us to track the motion of the object including translation (center movement), global rotation (axis rotation), and isotropic and anisotropic global scalings. We can now go through a nonkeyframe and compute the tensor field values for pixels in *key regions* using updated center location and pattern functions. Next, values from the key regions in the frame are propagated to *free regions* by the aforementioned tensor-valued relaxation process.

4 MULTISTYLE PAINTERLY RENDERING

Once all style parameters and stroke orientations have been populated to each pixel in every frame, we enter the third stage of our pipeline in which the video is processed, according to the style parameters. We provide two algorithms.

4.1 Explicit Stroke Method

The first one is a straightforward adaptation from the renderer of Hertzmann [13], and Hays and Essa [6]. For images, this approach first places the brush stroke seeds (points) on a jittered 2D grid. Then every seed is traced along a directional field, which results in a *streamline*. The streamline is then fattened into a curved region (stroke) by a uniformly defined stroke width. Once the canvas has been populated with these strokes, it is compared to a blurred version of the input image. Additional brush strokes with a smaller brush stroke width will be placed where the difference between the painting and the blurred image is larger than some threshold. This process then repeats. The resulting images can be further enhanced by embossing stroke textures to achieve a 3D effect [5]. It is straightforward to use this approach to support varying style parameters. The properties of a stroke are determined by the pixel for which it is centered. This way we do not have to recompute and reevaluate the strokes during streamline tracing. Fig. 6 illustrates this process with the painting of an eye image using varying style parameters. For videos, we follow the approach of Hays and Essa [6] by using the optical flow to move the centers of strokes from the previous frame and by adding and removing strokes when necessary.

There are a number of complications. For example, the number of layers, an integer, is a style parameter in our system that can be propagated and interpolated. Yet, after the relaxation process, this number must become a rational number. To overcome this problem, we snap the number of layers to the nearest integer while maintaining a continuously transitioned area error threshold. For a style that requires only one layer, we will set the corresponding area error threshold to the maximum number, which means no upper layer strokes are needed. Then, gradually lowering this number as one travels into multilayer regions in which the area threshold is also lower, the number of upper layer strokes gradually increases.

Another problem is the brush stroke density which leads to a strong sense of discontinuity. We do not address the density problem as it requires the handling of discontinuity in the number of strokes placed over the canvas. Instead, we assume a constant density for all styles. While this seems overly constraining, in practice, we have found that the variety of styles our approach can simulate is not significantly affected by this limitation.

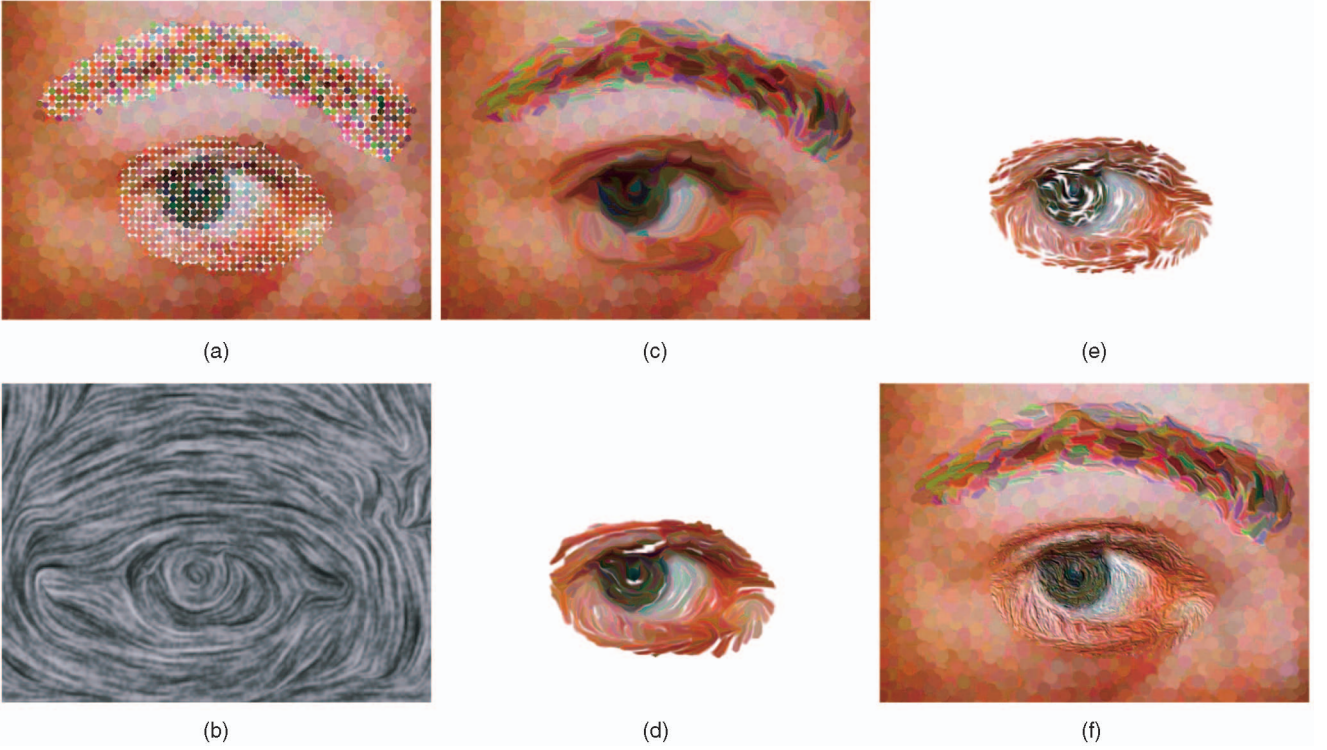


Fig. 6. An example workflow of our system: given the input image of an eye, the initial seeds are assembled in an image (a) which is advected according to the edge field shown in (b) to obtain the first-layer painting (c). Additional layers (d and e) are generated in a similar fashion, which are then combined with the first layer to form the final multistyle rendering (f). The eye is rendered in a three-layer Impressionist style, the eyebrow in a one-layer van Gogh style, and the face in a one-layer Pointillist style.

We have observed that this rendering technique seems to have a strong emphasis on individual strokes, especially for van Gogh and Impressionist styles. While this may be desired for image rendering, we have noticed much flickering in painterly processed videos due to the relative movement and visibility changes among overlapping strokes. Though we can use in-stroke opacity textures, it does not alleviate the problem due to the relatively small size of the strokes. To deal with this, we have developed a second renderer which we refer to as an *implicit stroke method*. For convenience, we will also refer to the renderer of Hertzmann [13] and Hays and Essa [6] as an *explicit stroke method*.

4.2 Implicit Stroke Method

Our implicit stroke method differs from the explicit stroke method mainly in one regard: stroke generation. To generate brush strokes on a layer, we first compute a set of seeds each of which is the starting point of a brush stroke on this layer. Next, every seed is drawn onto a temporary canvas C_0 as a colored disk where the color is from the pixel in the input I or a blurred version of it. Finally, we advect the canvas as an image, according to the stroke orientation field. The advection is performed by iteratively warping the current canvas C_i and compositing the result with the original canvas C_0 to obtain the new image C_{i+1} . Fig. 7 demonstrates this process. When composing the images at the pixel level, we face a challenge that the ID of the strokes are not well maintained during the warping of the canvas (as an image). Consequently, this can lead to inconsistent composition of pixels in overlapping strokes, which in turn, leads to color bleeding between them. To overcome this

problem, we consider the following composite function f for two images A and B :

$$f(\mathbf{p}) = \begin{cases} A(\mathbf{p}), & b_B(\mathbf{p}) = 1, \\ B(\mathbf{p}), & b_A(\mathbf{p}) = 1, \\ \min(A(\mathbf{p}), B(\mathbf{p})), & b_A(\mathbf{p}) = b_B(\mathbf{p}) = 0, \end{cases} \quad (5)$$

where $b_R(\mathbf{p})$ is a binary-valued function that takes a value of 0 if the pixel \mathbf{p} is covered by a stroke in image R . Otherwise, it is given a value of 1. Basically, if \mathbf{p} is not covered by any brush strokes in one of the images in composition, we use the color of \mathbf{p} from the other image. This ensures consistent colors within a single stroke as it is being advected over the canvas. Note that this includes the case when $b_A(\mathbf{p}) = b_B(\mathbf{p}) = 1$, i.e., the pixel is not covered by brush strokes in either image. In this case, the pixel belongs to background in both images and will remain so after the composition.

When a pixel is covered in both images, i.e., there are at least two brush strokes that cover it, we choose the color which is smaller, according to a total order on the space of

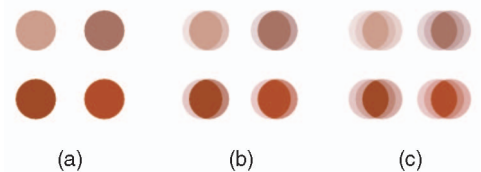


Fig. 7. This figure illustrates our process of advecting a source image (a) in the direction of an underlying vector field V . The image in (b) is obtained by warping (a) based on V and composed with (a). Warping (b) and composing it with (a) results in (c).

colors. Given two colors $C_1(R_1, G_1, B_1)$ and $C_2(R_2, G_2, B_2)$, we choose C_1 if

1. $R_1^2 + G_1^2 + B_1^2 < R_2^2 + G_2^2 + B_2^2$, or
2. $R_1^2 + G_1^2 + B_1^2 = R_2^2 + G_2^2 + B_2^2$ and $G_1 < G_2$, or
3. $R_1^2 + G_1^2 + B_1^2 = R_2^2 + G_2^2 + B_2^2$ and $G_1 = G_2$ and $R_1 < R_2$, or
4. $R_1^2 + G_1^2 + B_1^2 = R_2^2 + G_2^2 + B_2^2$ and $G_1 = G_2$ and $R_1 = R_2$ and $B_1 < B_2$.

Otherwise, we choose C_2 . In practice, we have found less than one percent of total pixels ever require the second test. Notice that any total ordering can reduce bleeding between neighboring strokes. Finding the optimal ordering that achieves the best artistic looks is beyond the scope of this paper and left for future exploration. The idea of a total color ordering has been used in adding security to images [26]. Notice our definition is slightly different from theirs.

While the implicit stroke method can also be accomplished through techniques such as LIC [19], we choose to adapt the technique of texture-based tensor field visualization technique of Zhang et al. [11], which is based on the IBFV technique of van Wijk [21].

The explicit and implicit methods can both support interactive design, with the explicit method being slightly faster (under 0.5 second per frame for one layer). On the other hand, the implicit method appears to focus less on individual strokes, and thus, is less sensitive to sudden change in strokes and more coherent in the video setting. All the example videos in this paper were generated using the implicit method.

5 VIDEO ANALYSIS AND SEGMENTATION

In this section, we describe how we compute the optical flow and perform video segmentation. None of this work is novel as we reuse existing methods.

5.1 Edge Field and Optical Flow Estimation

An automatic video painting algorithm typically extracts the following information from the video: a directional field that orients brush strokes in each frame, and an optical flow field that moves brush strokes to follow the underlying objects that they depict [7], [6].

Our estimation of the directional field is based on [13], [11]. First, we estimate the image gradient vector field G by using a Sobel filter. Next, we generate a second-order symmetric tensor field whose minor eigenvectors align with the image gradient vectors. The major eigenvectors of this tensor field are the directional field. Finally, we perform tensor field smoothing [11] in both space and time. This leads to more spatiotemporally coherent stroke orientations. Note that this directional field will be modified during the second phase of our framework: style and orientation design (Section 3.2). Moreover, it can be computed using other edge detection filters.

Optical flow estimation is done using the Lucas-Kanade method [27]. In practice, we compute both the forward and backward optical flow fields which allows us to transfer user specification in both style and orientation design from keyframes to other frames (Section 3).

5.2 Spatiotemporal Video Segmentation

Obtaining spatiotemporal video segmentation is necessary for any region-based video processing operations, such as video matting [28], rotoscoping [29], and video tooning [9]. In our framework, it enables region-based style and orientation design. There are several approaches to achieving a spatiotemporal video segmentation [30], [31], [10], [32], [9]. For our purposes, any of these spatiotemporal video segmentation methods can be used. Specifically, we use the method presented in [12], because it does not require any models of, or prior knowledge about objects and their motions in the scene. Also, it is relatively fast and user-friendly (see more detailed specifications in Section 6).

We wish to emphasize that it is not the goal of this paper to develop a new video segmentation method. Therefore, below, we will only briefly review the segmentation method of [12] that we use in this paper, for completeness. The algorithm initially performs frame-by-frame 2D segmentation, and then tracks similar regions across the frames, such that the resulting tracks are locally smooth. Tracking is conducted by many-to-many matching of groups of spatially adjacent regions in one frame with groups of adjacent regions in the next frame. This partitions the spatiotemporal video volume into tubes that are coherent in space and time. Since region boundaries coincide with object boundaries, a cross section of the tubes and any video frame delineates all objects present in the frame. The extensive experimental evaluation of this method, presented in [12], suggests that the proposed approach compares favorably with the state of the art.

6 RESULTS

We have applied our system to a large number of example images or videos. Figs. 2, 3, and 4 provide examples in which an input image is processed using multistyle rendering to achieve various artistic goals. Next, we show frames from processed videos.

As the first example, we show the power of spatially varying styles with a video of a blooming flower that was segmented into three regions: stamens, petals, and leaves (Fig. 1b). The stamens contain the highest frequency details and the petals have strongest anisotropy. There is also motion in the leaves due to the movement of the highlight (upper-right). Rendering the video, using the same style parameters such as van Gogh (Fig. 1c), often cannot adequately maintain the contrast between these characteristics. Using spatially varying style parameters, the artist made the following style assignments: stamens (a Pointillist style), petals (van Gogh style), and leaves (a Pointillist style with a diameter twice as large as that for the stamens) (Fig. 1d). With this setting, the detail in the stamens is maintained without sacrificing the contrast between the three regions.

The second example video shows a setting sun moving behind clouds (Fig. 8a). The artist increased the contrast between the clouds and sun by gradually shifting the hue of the background (clouds and sky) from yellow to purplish blue through controlled color modification and by shifting the saturation of the strokes representing the sun toward pure colors (Fig. 8b). This allows stroke movement in the dark sky region to become more visible, thus shifting the focus from the sun to the sky and clouds to reinforce that

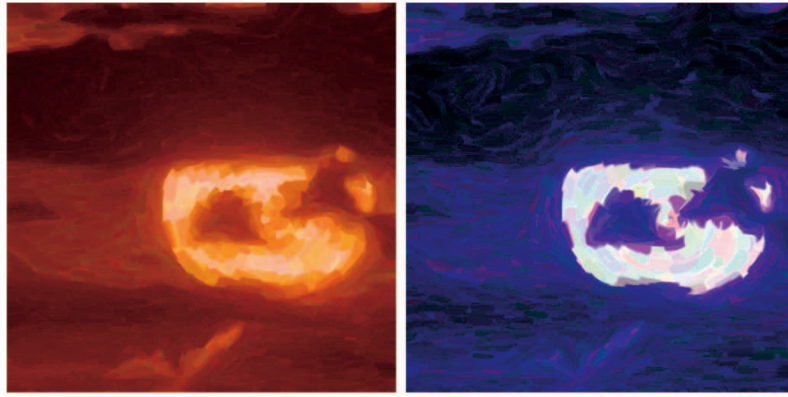


Fig. 8. Two frames from a video of sunset. Notice the change of hue in the background color allows the additional focus on the sky and clouds.



Fig. 9. Rack focus is shown on three frames from a video with space-time-varying stroke diameters: (a: Frame 161) mother in focus and girl out of focus, (b: Frame 429) mother out of focus and girl coming into focus, and (c: Frame 600) both women in focus.

the night is coming. The design was inspired by van Gogh's *Starry Night*.

The third example highlights the benefits of time-varying styles in achieving an effect similar to a rack focus used in cinema. In a rack focus the lens focus is changed during a shot so that one object remains in focus while the other goes out of focus. This is used to direct viewer attention to different regions of the frame. In our example, we are using variation in Pointillist brush size as a means of emphasizing or de-emphasizing regions of an image. The video shows a woman, her child, and their interaction. The video is segmented into the woman, her hat and hair, the child, her hair, and the background. There are a number of motions in the video such as the woman smiling, pointing, and kissing the child as well as the child turning her head and smiling. Fig. 9 shows three frames from the output video in which the artist initially assigns a style to both faces with high details. Then, the style is gradually varied to a low-detailed one for the child to reduce the attention on her as the mother points, turns her head, and smiles (A frame in which this has occurred is shown in Fig. 9a). Later, the style for the child transitions back to the original high-detail style as her mother kisses her. Meanwhile the style for the mother is switched to the same low-detail one used on the girl (Fig. 9b). Toward the end, the styles for both women are returned to the same original high-detailed one (Fig. 9c). Notice the rack focus effect generated by varying style parameters over time.

In the next example, we show temporal transition of stress to calmness with a video showing a dolphin repeatedly jumping out of water (Fig. 10). Starting with a sense of stress in which the artist shifts the stroke hues. This effect then smoothly transitions into a sense of calmness as the degree of color shifts is reduced.

In the last example, we demonstrate the power of combining varying style parameters with stroke orientation design with the dolphin video. In Fig. 11a, the frame was rendered using the same style parameters for every object in the scene: the dolphin, the water, the sky, a mountain, and a bridge. The artist then assigned different style parameters to these objects (Fig. 11b) to add more details to the dolphin. The interaction between the dolphin and the water was further enhanced through stroke orientation design to create the illusion of ripples (Fig. 11c: where the dolphin touches the water).

The rendered clips corresponding to these images can be found in the supplementary video.¹ The runtime of the employed video segmentation method [12] depends on the number of objects of interest and the frame size. For the dolphin video, it took about 30 seconds to process 100 frames without any user intervention. However, this process is slowed down by changes that the user may want to make in every frame. The design of style parameters and orientation fields depends on the number of objects, the number of

1. http://web.engr.orst.edu/~zhange/multistyle_painting.html.

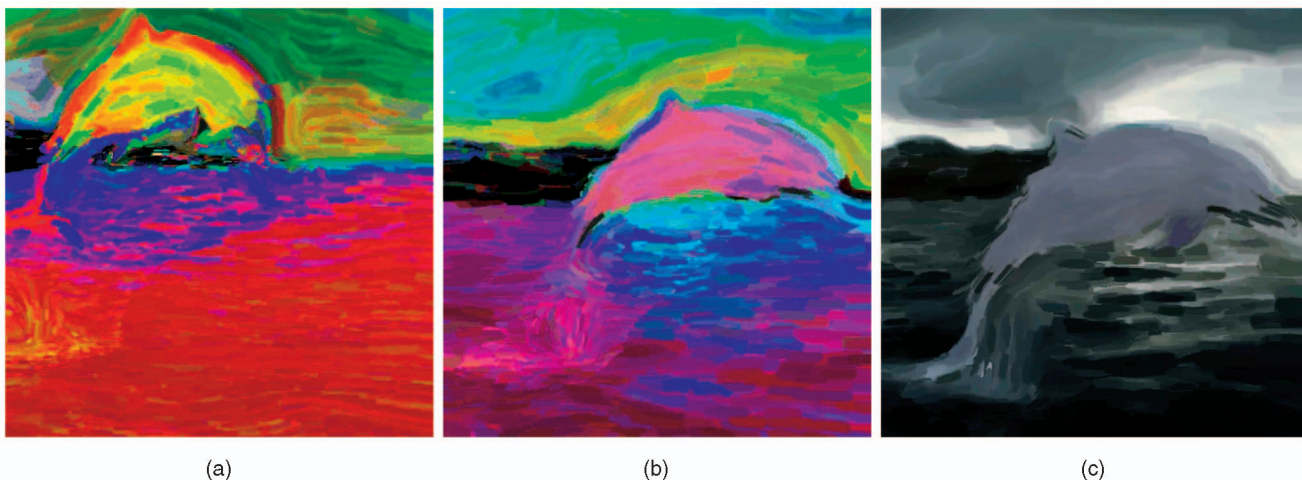


Fig. 10. Using space-time-varying style parameters, we achieve a transition of a stress effect (a) to a calm effect (c). In (b), the transition is shown.

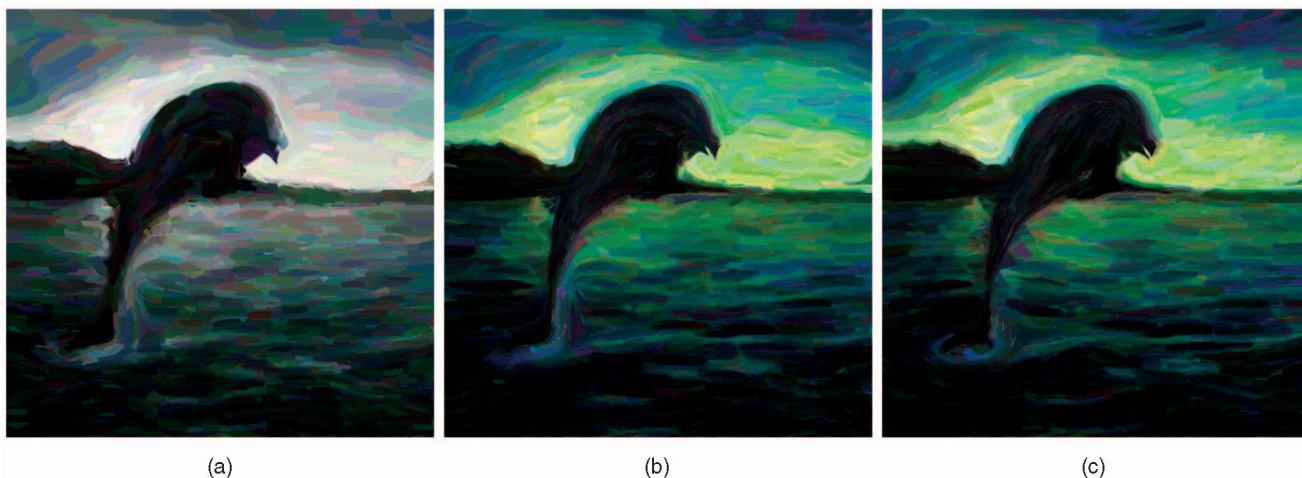


Fig. 11. Corresponding frames from three different renderings of a video of a swimming dolphin: (a) single style, (b) multiple styles, and (c) same as (b) with additional design of stroke orientations. Notice that with control over stroke orientations (c), an illusion of rippling effect was added while the dolphin was coming out of the water.

keyframes, and the number of design changes to be made. For the flower video, where there are three regions, five keyframes, and the total design time was 230 seconds for 311 frames. Please note that the parameters for the supplementary video were designed by the artist on our team, who was a novice to our system and needed only a small period of time to be productive with our system—the period that is comparable to learning to use any other commercially available painterly rendering system. The design process is made easier with our system by allowing the user to start from some existing styles and make a small number of adjustments. The orientation field design for the dolphin took approximately 10 minutes. Furthermore, the rendering time for these videos is on the average of 2.5 seconds/frame. The segmentation was performed on a computer of 1 GB RAM and a 1.3 GHz CPU. The design and rendering was done on a computer that has an NVIDIA 8800 GTX card with 512 MB video memory and Pentium IV with a speed of 3.80 GHz.

7 CONCLUSION

In this paper, we describe a system for the design and painterly rendering of videos with style parameters varying

in space and time. Our system enables a wide range of artistic controls including brush stroke colors, widths, lengths, and opacity as well as brush stroke orientations. To our knowledge, the design of stroke orientation field in a temporally coherent fashion is the first of this kind. Our design tool is both interactive and intuitive. It can automatically propagate rendering parameters to the video thus reducing the amount of labor work. We provide two painterly renderers, explicit and implicit methods, by adapting existing painting methods and by applying flow visualization techniques, respectively. The implicit stroke method requires less focus on individual strokes, and is, thus, less sensitive to sudden changes in stroke visibility. We have demonstrated the effectiveness of our system with several examples.

The efficiency of our system greatly depends on the quality of the employed segmentation method. For example, when there is semitransparent object such as water in the dolphin video, we have noticed that it is difficult to always obtain a clean segmentation (Fig. 12). As the dolphin jumps in and out of water, it is not always clear how to classify pixels that represent the part of dolphin underwater



Fig. 12. A frame from the dolphin video along with its segmentation. Notice that part of the dolphin is underwater but visible due to the transparency of water. This makes it more challenging to obtain a coherent segmentation.

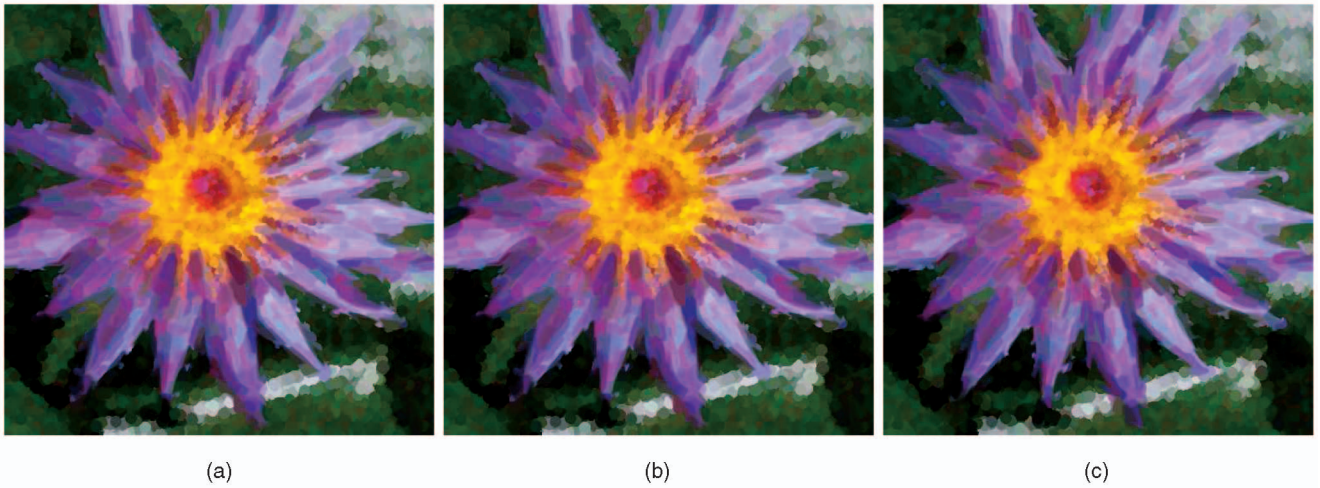


Fig. 13. An illustration of the flickering effect with the flower example. (b) is one frame after (a) while (c) is ten frames after (b). Notice that there is relatively little change in stroke visibility between (a) and (b), but significant changes between (b) and (c).

in a temporally coherent fashion. The vision researchers on this team plan to further investigate the issue as part of their future work.

Maintaining temporal coherence is still a great challenge. This is perhaps the most clear in the flower example, where there is stronger flickering effect toward the end when the flower has fully bloomed. Fig. 13 illustrates this with three frames. Notice that there is little difference between the first frame in the sequence (Fig. 13a) and the immediate next frame (Fig. 13b). However, when comparing Fig. 13b with Fig. 13c, that is, 10 frames after, it is quite clear that the order of neighboring strokes have changed in many part of the image. The relatively sudden visibility change between neighboring strokes is the source of the problem and requires further investigation. On the other hand, we note that the orientations of these strokes do not change significantly over time, indicating temporal coherence of our tensor fields.

There are a number of future research avenues for this work. First, we plan to investigate a more rigorous handling of time-dependent tensor field design. In particular, we wish to understand how to improve the quality of tensor field with explicit control over its smoothness and topology, such as singularities and bifurcations. We also plan to study how to edit the optical flow field using vector field design

techniques. Second, the idea of using color orders to sort brush strokes is interesting, and we plan to pursue this direction in the near future. Finally, we are interested in means to propagate style parameters and orientation fields directly in the space-time domain rather than the two-step approach we use in this paper. We believe that solving in 3D can lead to smoother results.

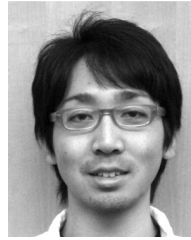
ACKNOWLEDGMENTS

The authors would like to thank all the reviewers whose constructive comments and suggestions have made this paper much stronger. The authors also wish to thank Artbeats.com for the videos used in this research as well as freefotos.com, pics4learning.com, US Department of Agriculture, and Greg Turk for the source images. This work is sponsored by the US National Science Foundation (NSF) Grant CCF-0546881.

REFERENCES

- [1] P. Haeberli, "Paint by Numbers: Abstract Image Representations," *Proc. ACM SIGGRAPH '90*, pp. 207-214, 1990.
- [2] P. Litwinowicz, "Processing Images and Video for an Impressionist Effect," *Proc. ACM SIGGRAPH '97*, pp. 407-414, 1997.

- [3] A. Hertzmann, "Paint by Relaxation," *Proc. Computer Graphics Int'l (CGI '01)*, pp. 47-54, 2001.
- [4] D. DeCarlo and A. Santella, "Stylization, Abstraction of Photographs," *Proc. ACM SIGGRAPH '02*, pp. 769-776, 2002.
- [5] A. Hertzmann, "Fast Paint Texture," *Proc. Second Int'l Symp. Non-Photorealistic Animation and Rendering (NPAR '02)*, pp. 91-97, 2002.
- [6] J. Hays and I. Essa, "Image, Video Based Painterly Animation," *Proc. Third Int'l Symp. Non-Photorealistic Animation and Rendering (NPAR '04)*, pp. 113-120, 2004.
- [7] A. Hertzmann and K. Perlin, "Painterly Rendering for Video Interaction," *Proc. First Int'l Symp. Non-Photorealistic Animation and Rendering (NPAR '00)*, pp. 7-12, 2000.
- [8] A. Bousseau, F. Neyret, J. Thollot, and D. Salesin, "Video Watercolorization Using Bidirectional Texture Advection," *Proc. SIGGRAPH '07, ACM SIGGRAPH 2007 Papers*, p. 104, 2007.
- [9] J. Wang, Y. Xu, H.-Y. Shum, and M.F. Cohen, "Video Tooning," *Proc. SIGGRAPH '04, ACM SIGGRAPH 2004 Papers*, pp. 574-583, 2004.
- [10] J.P. Collomosse, D. Rowntree, and P.M. Hall, "Stroke Surfaces: Temporally Coherent Artistic Animations from Video," *IEEE Trans. Visualization Computer Graphics*, vol. 11, no. 5, pp. 540-549, Sept. 2005.
- [11] E. Zhang, J. Hays, and G. Turk, "Interactive Tensor Field Design and Visualization on Surfaces," *IEEE Trans. Visualization and Computer Graphics*, vol. 13, no. 1, pp. 94-107, Jan. 2007.
- [12] W. Brendel and S. Todorovic, "Video Object Segmentation by Tracking Regions," *Proc. IEEE Int'l Conf. Computer Vision*, pp. 1-8, 2009.
- [13] A. Hertzmann, "Painterly Rendering with Curved Brush Strokes of Multiple Sizes," *Proc. ACM SIGGRAPH '98*, pp. 453-460, 1998.
- [14] A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, and D.H. Salesin, "Image Analogies," *Proc. ACM SIGGRAPH '01*, pp. 327-340, 2001.
- [15] A.W. Klein, P.-P.J. Sloan, A. Finkelstein, and M.F. Cohen, "Stylized Video Cubes," *Proc. ACM SIGGRAPH/Eurographics Symp. Computer Animation (SCA '02)*, pp. 15-22, 2002.
- [16] B.J. Meier, "Painterly Rendering for Animation," *Proc. ACM SIGGRAPH '96*, pp. 477-484, 1996.
- [17] M. Cunzi, J. Thollot, S. Paris, G. DeBunne, J.-D. Gascuel, and F. Durand, "Dynamic Canvas for Non-Photorealistic Walk-throughs," *Proc. Graphics Interface Conf.*, pp. 121-130, 2003.
- [18] E. Zhang, K. Mischaikow, and G. Turk, "Vector Field Design on Surfaces," *ACM Trans. Graphics*, vol. 25, no. 4, pp. 1294-1326, 2006.
- [19] B. Cabral and L.C. Leedom, "Imaging Vector Fields Using Line Integral Convolution," *Proc. ACM SIGGRAPH '93*, pp. 263-270, 1993.
- [20] D. Stalling and H.-C. Hege, "Fast, Resolution Independent Line Integral Convolution," *Proc. ACM SIGGRAPH '95*, pp. 249-256, 1995.
- [21] J.J. van Wijk, "Image Based Flow Visualization," *Proc. ACM SIGGRAPH '02*, pp. 745-754, 2002.
- [22] T. Delmarcelle and L. Hesselink, "Visualizing Second-Order Tensor Fields with Hyperstream Lines," *IEEE Computer Graphics and Applications*, vol. 13, no. 4, pp. 25-33, July 1993.
- [23] X. Zheng and A. Pang, "Hyperlic," *Proc. IEEE Visualization Conf.*, pp. 249-256, 2003.
- [24] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge Univ. Press, 1992.
- [25] N.J. Foster and A.C. Sanderson, "Determining Object Orientation Using Ellipse Fitting," *Proc. IEEE Int'l Conf. Computer Vision*, 2009.
- [26] C.-H. Tzeng, Z.-F. Yang, and W.-H. Tsai, "Adaptive Data Hiding in Palette Images by Color Ordering and Mapping with Security Protection," *IEEE Trans. Comm.*, vol. 52, no. 5, pp. 791-800, May 2004.
- [27] B.D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI)*, pp. 674-679, 1981.
- [28] Y.-Y. Chuang, A. Agarwala, B. Curless, D.H. Salesin, and R. Szeliski, "Video Matting of Complex Scenes," *Proc. ACM SIGGRAPH '02*, pp. 243-248, 2002.
- [29] A. Agarwala, A. Hertzmann, D.H. Salesin, and S.M. Seitz, "Keyframe-Based Tracking for Rotoscoping Animation," *Proc. ACM SIGGRAPH '04, ACM SIGGRAPH 2004 Papers*, pp. 584-591, 2004.
- [30] R. Megret and D. DeMenthon, "A Survey of Spatio-Temporal Grouping Techniques," Technical Report CS-TR-4403, Univ. of Maryland, 2002.
- [31] A. Agarwala, "Snaketoonz: A Semi-Automatic Approach to Creating Cel Animation from Video," *Proc. Second Int'l Symp. Non-Photorealistic Animation and Rendering (NPAR '02)*, pp. 139-148, 2002.
- [32] Y. Park and K. Yoon, "Painterly Animation Using Motion Maps," *Graphical Models*, vol. 70, nos. 1-2, pp. 1-15, 2008.



Mizuki Kagaya received the bachelor's degree from Oregon State University, and he is currently continuing on study for the master's degree in computer science at Oregon State University. He is working with Eugene Zhang, and his interested area in computer graphics research includes painterly rendering and geometric modeling.



William Brendel received the engineering master's degree from CPE Lyon, France, in 2005. He worked for Infineon Technologies in Munich, Germany, between 2003 and 2004, and for Laika Studios—House in Portland, OR, in 2007. Currently, he is a PhD student in the School of Electrical Engineering and Computer Science at the Oregon State University. His research mostly focuses on graphics, computer vision and machine learning problems, including 3D reconstruction, feature detectors and descriptors, texture gradients, and spatiotemporal video segmentation.



Qingqing Deng received the bachelor's degree and master's degree in computer science from Chongqing University, China, in 2004 and 2007, respectively. She is currently working toward a PhD degree in computer science at Oregon State University. She is working with Eugene Zhang on nonphotorealistic rendering and vector field visualization.



Todd Kesterson received the MS degree in environmental education from Southern Oregon University and the MFA degree in interdisciplinary arts from Goddard College in Vermont. He has worked professionally in the field of computer animation since 1990 on a wide range of projects for advertising, entertainment, and educational purposes. His work has been featured at ACM SIGGRAPH, at Sea World (Florida and San Diego), and on National Geographic television. He currently teaches courses in media aesthetics, conceptualization, and production in the New Media Communications Program at Oregon State University.



Sinisa Todorovic received the PhD degree in electrical and computer engineering from the University of Florida in 2005. He was postdoctoral research associate in the Beckman Institute at the University of Illinois Urbana-Champaign, between 2005 and 2008. Currently, he is an assistant professor in the School of Electrical Engineering and Computer Science at the Oregon State University. His research mostly focuses on computer vision and machine learning problems, including object recognition, texture analysis, and spatiotemporal video segmentation. He has published more than 30 journals and refereed conference papers. He is associate editor of *Image and Vision Computing Journal*, and program chair of first International Workshop on Stochastic Image Grammars (at CVPR 2009).



Patrick J. Neill received the BS and MS degrees in computer science, both from Oregon State University, in 2006 and 2008, respectively. His research interests include rendering and physical-based simulation. As a graduate student, he has worked with Eugene Zhang on performing fluid simulation on interacting, deformable surfaces as well as painterly rendering of videos. He is working at NVidia.



Eugene Zhang received the PhD degree in computer science from Georgia Institute of Technology in 2004. He is currently an assistant professor at Oregon State University, where he is a member of the School of Electrical Engineering and Computer Science. His research interests include computer graphics, scientific visualization, and geometric modeling. He received an NSF CAREER award in 2006. He is a member of the IEEE Computer Society and the ACM.

▷ **For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.**