Interactive Tensor Field Design and Visualization on Surfaces

Eugene Zhang, *Member*, *IEEE*, James Hays, and Greg Turk, *Member*, *IEEE*

Abstract—Designing tensor fields in the plane and on surfaces is a necessary task in many graphics applications, such as painterly rendering, pen-and-ink sketching of smooth surfaces, and anisotropic remeshing. In this article, we present an interactive design system that allows a user to create a wide variety of symmetric tensor fields over 3D surfaces either from scratch or by modifying a meaningful input tensor field such as the curvature tensor. Our system converts each user specification into a basis tensor field and combines them with the input field to make an initial tensor field. However, such a field often contains unwanted degenerate points which cannot always be eliminated due to topological constraints of the underlying surface. To reduce the artifacts caused by these degenerate points, our system allows the user to move a degenerate point or to cancel a pair of degenerate points that have opposite tensor indices. These operations provide control over the number and location of the degenerate points in the field. We observe that a tensor field can be locally converted into a vector field so that there is a one-to-one correspondence between the set of degenerate points in the tensor field and the set of singularities in the vector field. This conversion allows us to effectively perform degenerate point pair cancellation and movement by using similar operations for vector fields. In addition, we adapt the image-based flow visualization technique to tensor fields, therefore allowing interactive display of tensor fields on surfaces. We demonstrate the capabilities of our tensor field design system with painterly rendering, pen-and-ink sketching of surfaces, and anisotropic remeshing.

Index Terms—Tensor field design, tensor field visualization, nonphotorealistic rendering, surfaces, remeshing, tensor field topology.

1 Introduction

Many graphics applications make use of a second-order symmetric tensor field, which is equivalent to a line field that does not distinguish between the forward and backward directions. In painterly rendering, for instance, brushstroke orientations are guided by a line field that is often chosen to be perpendicular to the image gradient field [15], [12], [10]. In hatch-based illustration of smooth surfaces, hatches usually follow one of the principle directions of the curvature tensor [13], [11]. Similarly in anisotropic remeshing, principle curvature directions are used to build a quad-dominant mesh from an input mesh [1], [16]. Both the line field used in painterly rendering and the principle curvature directions for pen-and-ink sketching and anisotropic remeshing can be expressed as the eigenvectors of some symmetric tensor fields. In the remainder of this paper, we will focus on symmetric tensor fields only and will refer to them as tensor fields.

Tensor field design, the main topic of this paper, enables applications such as painterly rendering and hatch-based illustration to achieve different visual effects by using different tensor fields. It also allows a user to modify an existing tensor field to improve its quality. For instance, a numerical estimate of the curvature tensor field on a polygonal surface usually contains excessive *degenerate points*, where anisotropy disappears. Degenerate points often cause visual artifacts in hatch-based sketching [11], and they require special care when performing anisotropic remeshing in surrounding regions [1], [16], [7]. While tensor field smoothing can remove a large percentage of degenerate points, it often "washes away" natural features in the field. Tensor field design provides a user with explicit control over the smoothness of a tensor field as well as the number and location of the degenerate points that it contains. Furthermore, tensor field design enables the creation of synthetic features, such as textures, and real-world semantics that are otherwise difficult to extract through any automated process, such as the eyes in the bunny model (see Fig. 14). A tensor field design system can also be used to test the efficiency of tensor field visualization algorithms. By creating tensor fields with known configurations, it is straightforward to verify whether a visualization algorithm has correctly identified these configurations. Furthermore, tensor field design can be used in teaching tensor analysis in which students learn important concepts in tensor fields by creating example fields and manipulating them.

There are several challenges to tensor field design. First, such a system should enable a user to create a wide variety of tensor fields with relatively little effort. Second, the user needs to have control over tensor field topology, such as the number and location of the degenerate points in the field. Third, the system must support interactive design and display of a tensor field. While there are many high-quality offline visualization methods, interactive techniques have been lacking. Finally, creating a continuous tensor field on a 3D mesh surface requires that we deal with the discontinuities of surface normals at the vertices and across the edges.

To achieve these goals, we develop a two-stage tensor field design system for both planar domains and curved

- *E. Zhang is with the School of Electrical Engineering and Computer Science, Oregon State University, 2111 Kelley Engineering Center, Corvallis, OR 97331. E-mail: zhange@eecs.oregonstate.edu.*
- *J. Hays is with the Computer Science Department, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3891. E-mail: jhhays@cs.cmu.edu.*
- *G. Turk is with the College of Computing, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30332-0280. turk@cc.gatech.edu.*

Manuscript received 3 Oct. 2005; revised 22 Feb. 2006; accepted 23 Mar. 2006; published online 8 Nov. 2006.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0132-1005.

surfaces. In the first stage, a user can quickly produce an initial tensor field through a set of design elements. Every element is used to create a basis tensor field over the domain that has a degenerate point of an arbitrary *tensor index*. All basis fields are then summed along with an input field that is either zero or an application-dependent field, such as a numerical estimate of the curvature tensor of a 3D surface. In the second stage, the user modifies the initial tensor field through a set of predefined editing operations, such as moving a degenerate point to a more desirable location or canceling a pair of degenerate points that have opposite tensor indices. As the user modifies the field, our system quickly analyzes the result and provides visual feedback. The user may perform any number of editing operations before accepting the results.

Our system performs degenerate point pair cancellation and movement by converting a symmetric tensor field into a vector field such that there is a one-to-one correspondence between the set of degenerate points in the tensor field and the singularities in the vector field. This conversion and its inverse operation allow us to use singularity pair cancellation and movement operations for vector fields. These operations provide control over tensor field topology, such as the number and location of the degenerate points.

In order for our design system to work on curved surfaces, we adapt the surface vector field representation scheme that was developed for vector field design [35] to tensor fields. In this scheme, concepts from differential geometry such as *geodesic polar maps* and *parallel transport* are used to construct initial tensor fields and to perform tensor field analysis and editing.

We have also developed an interactive visualization algorithm for second-order symmetric tensor fields, which is an extension of the *image-based flow visualization* technique [32], [14], [33].

In this paper, we make the following contributions: First, we have identified tensor field design as an important problem in computer graphics and visualization. We will also demonstrate that the edge field in an image is better modeled as a tensor field than as a vector field when it comes to painterly rendering. Second, we present a tensor field design system for mesh surfaces. This system allows a user to create a wide variety of tensor fields in a fast and efficient manner, and it provides the user with control over the degenerate points in the field. To the best of our knowledge, this is the first time a tensor field design system has been proposed and developed. Third, we provide efficient implementations of degenerate point pair cancellation and movement by locally converting the tensor field into a vector field. The conversion is conceptually simple, yet it allows us to reuse algorithms from vector field analysis and design. Fourth, we develop a piecewise interpolation scheme that produces a continuous tensor field on a mesh surface based on tensor values defined at the vertices. This scheme supports fast and efficient tensor field analysis such as separatrix computation, and it removes the need for a global surface parameterization. Finally, we present an interactive and high-quality technique for visualizing surface tensor fields.

The remainder of the article is organized as follows: We first review some relevant background on tensor fields in Section 2. Then, in Section 3, we compare tensor fields and vector fields in terms of image edge extraction. In Section 4, we review relevant work in vector field design and in tensor field analysis and visualization. We present our interactive tensor field visualization technique in Section 5 and describe our tensor field design system in Section 6. Section 7 provides some results of applying our tensor field design system to various graphics applications, such as painterly rendering, pen-and-ink sketching of surfaces, and anisotropic remeshing. Finally, we summarize our contributions and discuss some possible future work in Section 8.

2 Background on Tensor Fields

We first review some relevant facts about tensor fields on surfaces. A tensor field T for a manifold surface $\mathbf{M}$ is a smooth tensor-valued function that associates to every point $\mathbf{p} \in \mathbf{M}$ a second-order tensor

$$T(\mathbf{p}) = \begin{pmatrix} T_{11}(\mathbf{p}) & T_{12}(\mathbf{p}) \\ T_{21}(\mathbf{p}) & T_{22}(\mathbf{p}) \end{pmatrix}.$$

A tensor $[T_{ij}]$ is *symmetric* if and only if $T_{ij} = T_{ji}$. Symmetric tensor fields appear in many graphics applications, such as the metric tensor for surface parameterization, the curvature tensor in remeshing, and the diffusion tensor in medical imaging. A symmetric tensor T can be uniquely decomposed into the sum of its isotropic part S and anisotropic (*deviate*) part A:

$$T = S + A = \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \mu \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{pmatrix}, \quad (1)$$

where $\mu \geq 0$. A has eigenvalues $\pm\mu$, and A and T have the same set of major eigenvectors

$$\left\{ \lambda \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \middle| \ \lambda \neq 0 \right\}$$

and minor eigenvectors

$$\left\{ \lambda \begin{pmatrix} \cos(\theta + \pi/2) \\ \sin(\theta + \pi/2) \end{pmatrix} \middle| \ \lambda \neq 0 \right\}.$$

In this paper, we explore the design of directional fields on 3D surfaces, which is equivalent to designing deviate tensor fields. A more general design system for symmetric tensor fields can be obtained by combining our system and a scalar field design system, such as the one described by Ni et al. [19].

A deviate tensor field $A(\mathbf{p})$ is equivalent to two orthogonal eigenvector fields: $E_1(\mathbf{p}) = \mu(\mathbf{p})e_1(\mathbf{p})$ and $E_2(\mathbf{p}) = \mu(\mathbf{p})e_2(\mathbf{p})$ when $A(\mathbf{p}) \neq 0$. Here, $e_1(\mathbf{p})$ and $e_2(\mathbf{p})$ are unit eigenvectors that correspond to eigenvalues μ and $-\mu$, respectively. E_1 and E_2 are the *major* and *minor* eigenvector fields of A. A point $\mathbf{p}_0$ is *degenerate* for a tensor field T if and only if $A(\mathbf{p}_0) = 0$. A degenerate point for a tensor field often serves the same purpose as a singularity for a vector field. The most basic types of degenerate points are *wedges* and *trisectors* (Fig. 2). Delmarcelle and Hesselink [5] define a *tensor index* for an isolated degenerate point $\mathbf{p}_0$ as follows: Let γ be a small circle around $\mathbf{p}_0$ such that γ contains no additional degenerate points and it encloses only one degenerate point, $\mathbf{p}_0$. Starting from a point on γ and traveling counterclockwise along γ, the major field (after normalization) covers the unit circle S^1 a number of times. This number is the tensor index of

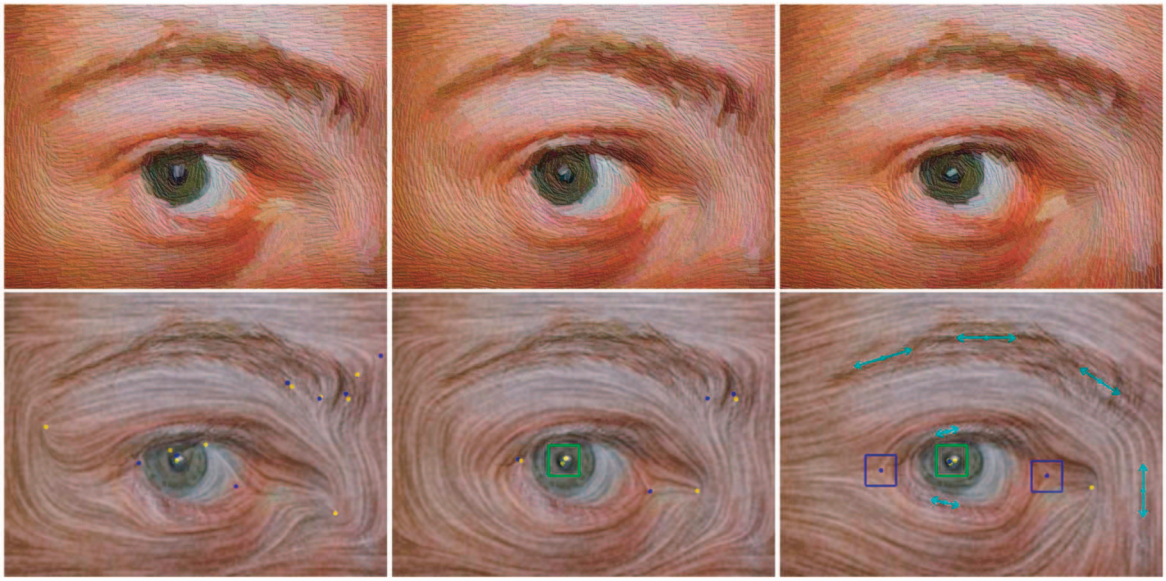

Fig. 1. This figure demonstrates how painterly rendering can benefit from tensor field design. For an input image of a human eye, three different tensor fields (bottom row) were to used to guide brush stroke orientations and produce van Gogh style paintings (top row): a tensor field extracted from the image (left), a combination of the previous field with a user-added center in the middle of eye (middle), and a tensor field designed completely from scratch (right). Notice that both designed fields (middle and right) are smoother in the pupil and near the corners of the eye. Tensor field design allows a user to guide brush stroke orientations in regions where the image gradient is weak. The painterly images shown here were produced by the algorithm of Hays and Essa [10]. The colored dots in the bottom images indicate the location and type of degenerate points in the fields: yellow for wedges and blue for trisectors.

$\mathbf{p}_0$, and it must be a multiple of $1/2$ due to the sign ambiguity associated with tensors. It is $1/2$ for a wedge and $-1/2$ for a trisector. The tensor index for a regular point is zero. Wedges and trisectors are first-order degenerate points. An Nth order degenerate point has a tensor index of $\pm(N/2)$. Second-order degenerate points include centers, nodes, and foci with an index of 1, and saddles with an index of -1 (Fig. 2). As in the case of vector fields, Delmarcelle shows that the total index of a tensor field with only isolated degenerated points is related to the topology of the underlying surface [6]. Let $\mathbf{M}$ be a closed orientable manifold with an Euler characteristic $\chi(\mathbf{M})$, and let T be a continuous tensor field with only isolated degenerate points $\{\mathbf{p}_i : 1 \leq i \leq N\}$. Denote the tensor index of $\mathbf{p}_i$ as $I(\mathbf{p}_i)$. Then,

$$\sum_{i=1}^{N} I(\mathbf{p}_i) = \chi(\mathbf{M}). \tag{2}$$

Delmarcelle and Hesselink [5] suggest visualizing *hyperstreamlines*, which are curves that are tangent to an eigenvector field everywhere along its path. To trace a hyperstreamline from a point, one needs to travel in both directions to obtain two "half" hyperstreamlines. Tracing in one direction results in the loss of sign ambiguity along the path, effectively turning the tensor field into a vector field. Different hyperstreamlines can only meet at degenerate points, and a degenerate point is a hyperstreamline that consists of a single point. Other special hyperstreamlines include *separatrices* and *closed orbits*, which together with degenerate points define the *topology* of a tensor field [5]. In this work, we focus on controlling the degenerate points in a tensor field.

3 Images and Tensors

When it comes to representing natural directions in an image or on a 3D shape, tensor fields provides a larger vocabulary of visual elements than vector fields. For instance, the basic types of degenerate points (wedges and trisectors) do not appear in continuous vector fields. On the other hand, higher-order degenerate points can be used to mimic the visual behavior of a vector field singularity of any order. For instance, a node in a tensor field is visually similar to a source or sink in a vector field, and a fourth-order degenerate point has a similar appearance as a dipole in the vector field. In painterly rendering, brushstroke orientations are often guided by a field F that is perpendicular to the image gradient vector field [15], [12],

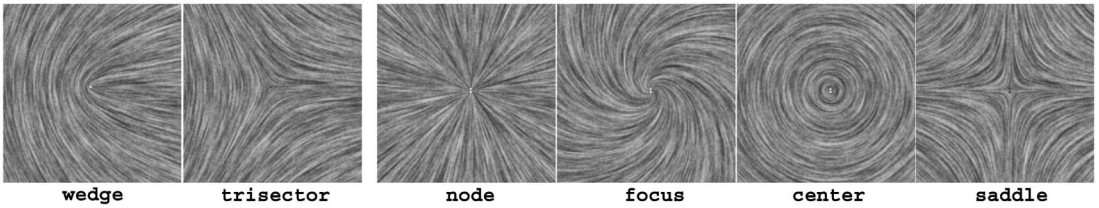

Fig. 2. Some canonical first and second-order degenerate points in a tensor field. Notice that the second-order points (node, focus, center, and saddle) are visually similar to first-order singularities in a vector field.

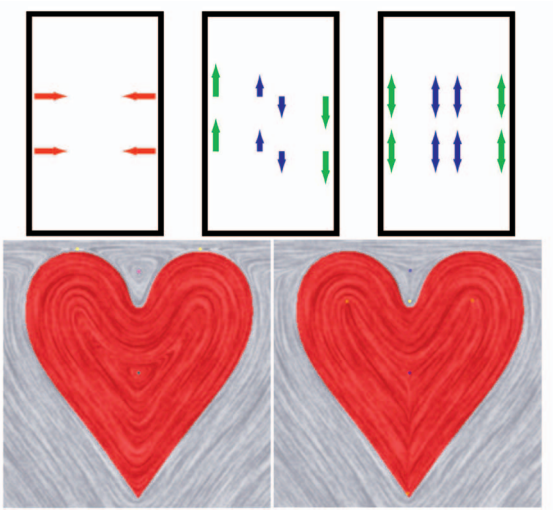

Fig. 3. This figure illustrates the difference between vector-based image edge fields (VIEF) and tensor-based image edge fields (TIEF). For the rectangle (top row), the image gradient vector field along the walls points to the other side (left, red arrows). This causes VIEF to point in opposite directions, and extrapolating values from the wall to the interior of the rectangle cause singularities (middle, green and blue arrows). TIEF does not suffer from this problem due to the sign ambiguity of directions (right). For the heart, TIEF (right) is much smoother than VIEF (left) in the interior region due to the richer vocabulary in tensor fields than vector fields. Inside the heart region, VIEF can only have an elongated center as the singularity. On the other hand, TIEF contains a trisector and two wedges, which are more natural here.

[10]. There are two ways of representing F: vector-based image edge field (VIEF) and tensor-based image edge field (TIEF). VIEF is obtained by rotating the image gradient by $\pi/2$ counterclockwise, and TIEF is the tensor field whose minor eigenvector field is colinear with the image gradient field everywhere in the domain. Often, the image edge field is computed where the image gradient is strong. Then, values in these regions are propagated to other regions where the image gradient is weak [15], [12], [10]. Under this scenario, however, TIEF provides a smoother representation than VIEF. Fig. 3 illustrates this with two examples: a rectangle and a heart. In the rectangle example (top), the values of the image gradient vector field are strong on the inner walls (left, red arrows), and they point toward the other side. This causes VIEF (middle) to point upward along the left wall and downward along the right wall (green arrows). Propagating these values to the interior of the rectangle leads to singularities. On the other hand, TIEF does not suffer from this problem due to the sign ambiguity (right). In the heart example, both VIEF and TIEF capture the boundary of the shape. However, TIEF is smoother and more uniform elsewhere than VIEF, for example, inside the heart region. This is due to the fact that the vector field can only have vortices as singularities while a tensor field can have wedges and trisectors that are more natural in this example. Fig. 4 illustrates the difference between VIEF and TIEF in painterly rendering with an example image of a duck. Notice VIEF (left) contains more noise than TIEF (right), which causes artifacts in the painterly results (compare the region near the beak). Here, the noise in VIEF is again due to the fact that a vector field can only have vortices as the basic singularities. For LIC-style rendering, a vector field can be treated as a tensor field by tracing a trajectory in both the forward and reverse directions. When tracing a hyperstreamline, Interrante et al. use the current direction to remove the sign of ambiguity [13]. Such an approach has also been used in painterly rendering. For example, Hertzmann compute long, curved brushstrokes by tracing streamlines according to a vector-based edge field [12]. In this paper, we also adopt this approach when computing separatrices in tensor field analysis (Section 6.1) and when computing streamlines for pen-and-ink sketching (Section 7.2) and anisotropic remeshing (Section 7.3).

A vector field can be treated as a tensor field if one ignores the direction. In contrast, treating a tensor field as a vector field requires that sign ambiguity in the tensor field be removed, which in general will create discontinuities in the resulting vector field. This issue has two implications that we have to deal with. First, the image-based flow visualization technique of van Wijk [32] does not directly apply to tensor fields. Second, using a vector field design system to modify a tensor field is likely to be unsuccessful due to its lack of ability to create and control tensor-specific features such as wedges and trisectors. We address these problems and provide solutions in Sections 5 and 6, respectively.

4 Previous Work

Tensor field analysis and visualization have been well-researched by the scientific visualization community. To review all of this work is beyond the scope of our article. We will only refer to the work that are most relevant to ours. Tensor field design, on the other hand, have received relatively little attention. To the best of our knowledge, there are no published tensor field design systems. Next, we review past work in vector and tensor field visualization and analysis as well as existing design systems for vector fields.

Delmarcelle and Hesselink [4] propose to visualize 2D or 3D tensor fields with hyperstreamlines, which has proven very efficient in revealing features in a tensor field. Around the same time, Cabral and Leedom [2] present a texture-based technique for visualizing planar vector fields with the use of *line integral convolution* (LIC). Given an initial texture of white noises and a vector field, they assign a gray value to every pixel by performing line interval convolution along the streamline that contains the pixel. The LIC method 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) [23]. Zheng and Pang [36] 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. Van Wijk [32] 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. Later, van Wijk [33] and Laramee et al. [14] extend IBFV to 3D surfaces. IBFV is at the core of our visualization technique, which we will describe in Section 5.

Delmarcelle and Hesselink demonstrate the importance of topological analysis for tensor field visualization. They also provide theories and algorithms for computing tensor field topology, such as degenerate points and separatrices [5]. Tensor fields from scientific data sets often contain noise, which makes visualization difficult. Tricoche and

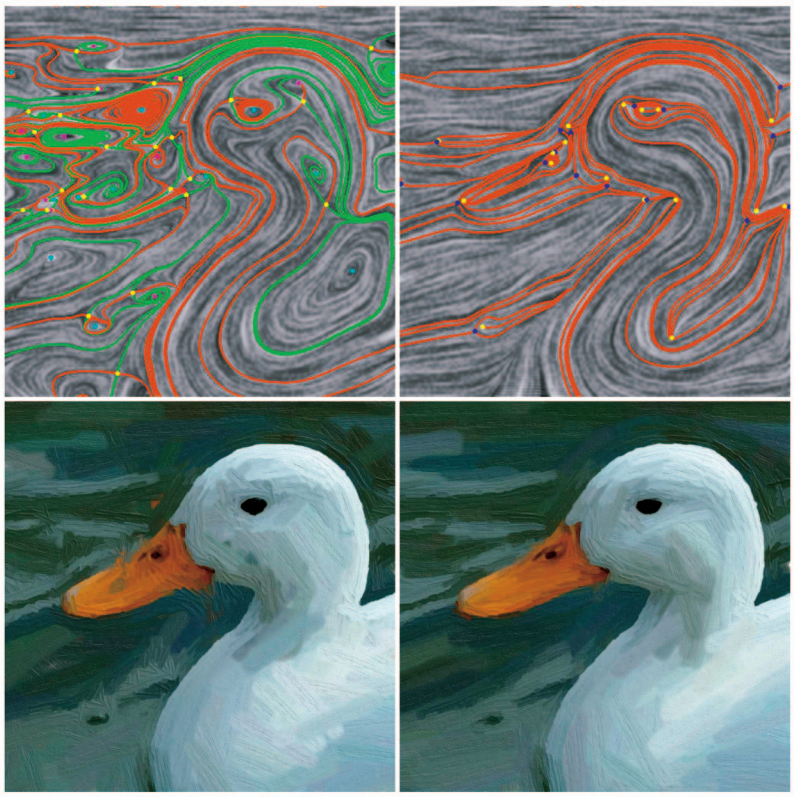

Fig. 4. Comparison between the vector-based image edge field (VIEF, left) and the tensor-based image edge field (TIEF, right) for painterly rendering of an image of a duck. Notice that TIEF is much smoother than VIEF (top row), and their impact on the painterly results are clearly visible near the beak of the duck.

Scheuermann [29] simplify the topology of a tensor field by performing "pair annihilation" on degenerate point pairs that are spatially close. They also cluster nearby first-order degenerate points into higher-order ones [27]. Alliez et al. [1] perform tensor field smoothing to remove noise in the curvature tensor, which also tends to reduce the number of degenerate points. Our system provides operations for both types of tensor field simplification (Section 6.2).

While tensor field design systems are lacking, there have been published work on scalar and vector field design. Ni et al. [19] allow a user to design *fair Morse functions* (scalar fields) on 3D surfaces for a number of graphics applications, such as parameterization and remeshing [7]. Vector field design has been used in texture synthesis [20], [30], [34], fluid simulation [24], and vector field visualization [32], [33]. These algorithms were developed in a quick manner to generate vector fields for a particular application, and the details of these systems were not published. Rockwood and Bunderwala [21] develop a vector field design system based on geometric algebra. All of these design systems lack control over vector field topology, such as singularities. The design system of Theisel [25] allows a user to control vector field topology, but it requires the user to provide the complete topological skeleton, which is cumbersome. Zhang et al. [35] introduce an interactive vector field design system that provides users with control over the number and location of the singularities in the field. In addition, their system works for both planar domains and curved mesh surfaces. Our tensor field design system is reminiscent of their system in terms of the functionalities. However, their system cannot be used to modify tensor fields, such as the tensor-based image edge fields and the curvature tensor fields.

5 Image-Based Tensor Visualization

In this section, we present our interactive visualization technique for planar and surface tensor fields. This technique is an extension of the image-based flow visualization techniques [32], [33], [14]. To visualize vector fields, IBFV produces streaks in the direction of the flow starting from an initial image (usually white noise). This initial image is warped in the flow direction by texturing a coarse 2D mesh with the image and then moving the mesh vertices along the flow. The warped image is blended with the old image, and the process is repeated.

To visualize a tensor field T, we find it sufficient to show only the major eigenvector field E_1. No information is lost by omitting a view of the minor eigenvector field E_2 since it is simply the major field rotated by $\frac{\pi}{2}$. To visualize E_1, it is desirable to convert E_1 into a continuous vector field V so that we can apply vector field visualization techniques, such as IBFV. One obvious way to perform this task is to choose a direction for every point in the domain. However, V will contain discontinuities that cannot always be eliminated. For instance, the tensor field

$$T(x,y) = \begin{pmatrix} x & y \\ y & -x \end{pmatrix}$$

contains a wedge at (0, 0). Assume there is a way to assign directions to every point such that the sign ambiguity is

removed. Then, (0, 0) becomes a singularity in V. However, the Poincaré index of a first-order singularity is ± 1, which is impossible to achieve for the wedge. This is because the total Poincaré index of a region for a vector field must be an integer, and the total index of T for the same region is $1/2$. In fact, a necessary condition for the existence of a consistent assignment is that the tensor field contains no degenerate points of an odd order (first, third, fifth, etc).

Let D denote the domain and $S(V) \subset D$ be the set of points where V is discontinuous. While it is not always possible to construct a vector field V from T such that $S(V) = \emptyset$, we build two vector fields V_1 and V_2 such that $\bigcap_i S(V_i)$ contains only the degenerate points of T, and every regular point in the domain belongs to $D \setminus S(V_i)$ for some i. The major eigenvector field E_1 can be represented in terms of two spatially varying scalar fields ρ and θ, which are the magnitude and direction, respectively. Specifically,

$$E_1 = \pm\rho \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \ (\rho \geq 0).$$

We define the following two vector fields from E_1:

$$V_1 = V_x = \begin{cases} \rho \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} & \text{if } \cos\theta \geq 0 \\ \rho \begin{pmatrix} -\cos\theta \\ -\sin\theta \end{pmatrix} & \text{otherwise,} \end{cases} \tag{3}$$

$$V_2 = V_y = \begin{cases} \rho \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} & \text{if } \sin\theta \geq 0 \\ \rho \begin{pmatrix} -\cos\theta \\ -\sin\theta \end{pmatrix} & \text{otherwise.} \end{cases} \tag{4}$$

Basically, V_x is obtained from E_1 by choosing directions so that the x-component of V_x is nonnegative everywhere. Therefore, $S(V_x) = \{(x, y) | \cos(\theta(x, y)) = 0\}$. Similarly, V_y is obtained by choosing directions so that the y-component of V_y is nonnegative and $S(V_y) = \{(x, y) | \sin(\theta(x, y)) = 0\}$. $S(V_x) \bigcap S(V_y)$ is the set of degenerate points. Let I_x and I_y be the images produced using IBFV with V_x and V_y, respectively. Let $W_X = \cos^2\theta$ and $W_y = \sin^2\theta = 1 - W_x$ be the blending functions. Then, the final image $I = W_x \times I_x + W_y \times I_y$ produces the desired result. Fig. 5 illustrates this process for a tensor field T (Fig. 5d). We compute the IBFV images based on V_x (Fig. 5a) and V_y (Fig. 5b). Notice the visual artifacts caused by the discontinuities in these images. The weight function W_x (Fig. 5c) is shown according to the following color coding: From 0 to 1 in increasing order, the colors are dark, red, yellow, and green. To extend this technique to visualizing a surface tensor field T, we project E_1 onto the image space and apply the two-image blending technique to the projection. Notice our visualization technique does not provide information on eigenvalues. This information may be added through color coding as was done by Urness et al. [31].

6 Tensor Field Design

In this section, we describe our two-stage tensor field design system for planar domains.

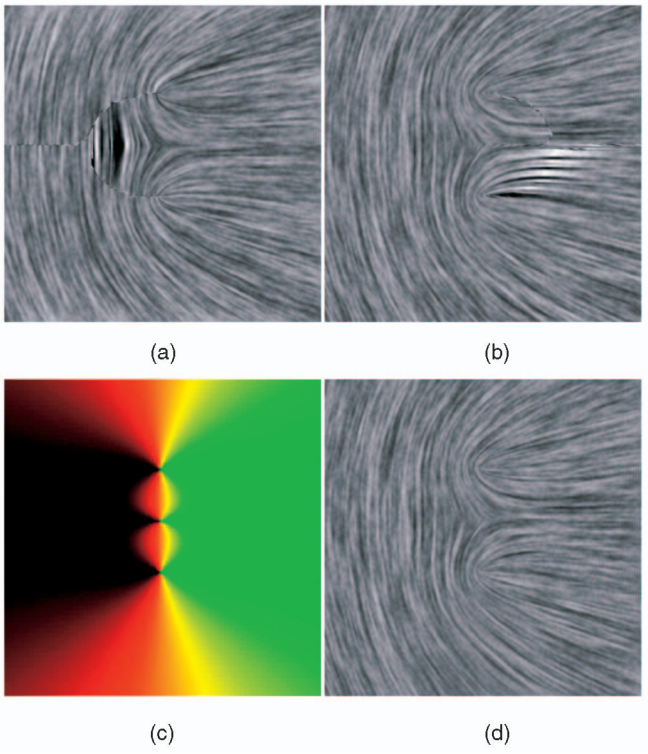

Fig. 5. This figure illustrates our visualization technique with a planar tensor field. The system first produces images according to two direction assignments: ((a), in the positive x-direction) V_x and ((b), in the positive y-direction) V_y. The images are then blended according to weight functions W_x (a color coding shown in (c)) and $W_y = 1 - W_x$. (d) The resulting image no longer contains the visual artifacts from V_x and V_y.

6.1 Initialization and Analysis

During the initialization stage, our system allows a user to quickly create an initial tensor field through a set of *design elements*. An element can be either *regular* if a desired tensor value is specified, or *singular* if a particular type of degenerate point is needed. For our applications, we have found that it is usually sufficient to provide specifications up to second-order degenerate points (first-order: wedges and trisectors; second-order: nodes, centers, and saddles; see Fig. 2). Every design element is extended to a globally defined *basis field*, and the user-defined tensor field is a sum of these basis fields. Similar ideas have been used by van Wijk to create vector fields with desired behaviors [32]. With a carefully chosen set of weight functions, the resulting tensor field will have desired tensor values or degenerate points as specified by individual elements. Next, we provide details on how to build a basis field from a regular or singular element.

Given a regular element (S_0, T_0) defined at $\mathbf{p}_0$, we compute $\rho_0 = \sqrt{S_0^2 + T_0^2}$ and $\theta_0 = \arctan\left(\frac{T_0}{S_0}\right)$ and define the following basis field:

$$T(\mathbf{p}) = e^{-d\|\mathbf{p}-\mathbf{p}_0\|^2} \rho_0 \begin{pmatrix} \cos 2\theta_0 & \sin 2\theta_0 \\ \sin 2\theta_0 & -\cos 2\theta_0 \end{pmatrix}, \tag{5}$$

where d is a decay constant that is used to control the amount of influence of the basis field. The weight function $e^{-d\|\mathbf{p}-\mathbf{p}_0\|^2}$ is strong near the center of the element and grows weaker for points farther away, which allows us to combine basis tensor fields by summing them and still maintain

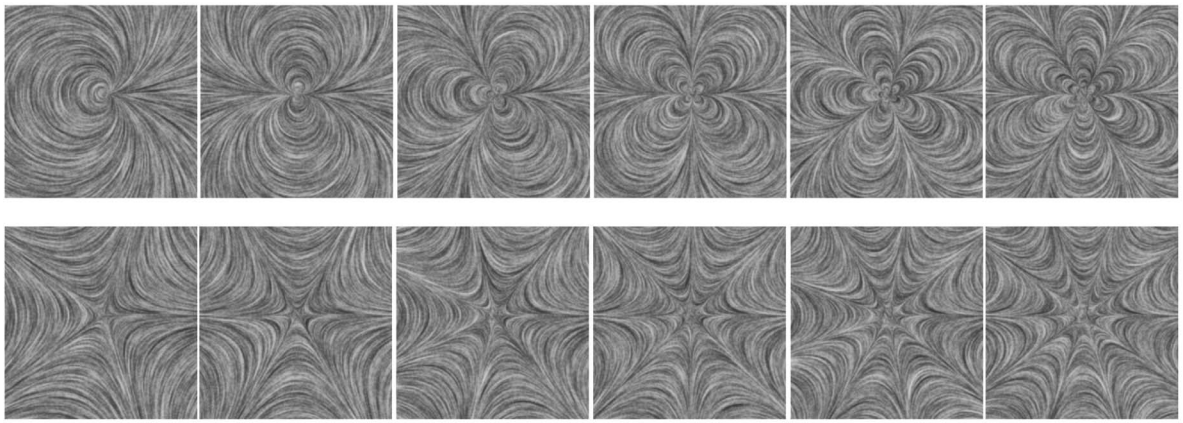

Fig. 6. Example basis tensor fields corresponding to higher-order design elements with a positive tensor index (top row) or a negative tensor index (bottom row). From left to right are third through eighth-order elements.

desired values at specified locations. Notice that other weight functions can be used instead as long as they also satisfy this property.

Singular elements can be extended to create basis tensor fields in a similar fashion. For example, to create a basis field with a wedge point at $\mathbf{p}_0 = (x_0, y_0)$ such that its only separatrix is extended in the positive X-axis, we use the following formula:

$$T(\mathbf{p}) = e^{-d\|\mathbf{p}-\mathbf{p}_0\|^2} \begin{pmatrix} x & y \\ y & -x \end{pmatrix}, \tag{6}$$

where $x = x_{\mathbf{p}} - x_0$ and $y = y_{\mathbf{p}} - y_0$. The following matrices produce a trisector, a node, a center, and a saddle, respectively.

$$\begin{pmatrix} x & -y \\ -y & -x \end{pmatrix}, \quad \begin{pmatrix} x^2 - y^2 & 2xy \\ 2xy & -(x^2 - y^2) \end{pmatrix}$$
$$\begin{pmatrix} y^2 - x^2 & -2xy \\ -2xy & -(y^2 - x^2) \end{pmatrix}, \quad \begin{pmatrix} x^2 - y^2 & -2xy \\ -2xy & -(x^2 - y^2) \end{pmatrix}.$$

Our system allows a user to modify the location, orientation, and scale of a singular element as well as to remove an existing element. Modifications to a singular element will result in more complicated matrices.

To allow an arbitrary tensor field to be created, our system allows the use of a design element of any order. Recall that an Nth-order element has a tensor index of $\pm(\frac{N}{2})$. Such an element can be created by using the following matrix:

$$D^N \begin{pmatrix} a\cos(N\theta) + b\sin(N\theta) & c\cos(N\theta) + d\sin(N\theta) \\ c\cos(N\theta) + d\sin(N\theta) & -(a\cos(N\theta) + b\sin(N\theta)) \end{pmatrix}, \tag{7}$$

where

$$D = \sqrt{(x - x_0)^2 + (y - y_0)^2},$$

$$\theta = \arctan\left(\frac{y - y_0}{x - x_0}\right),$$

and the matrix $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ has a full rank. The sign of tensor index of a degenerate point is the same as the sign of $ad - bc$ (see the Appendix for more details). The values of a, b, c, and d determine the geometric characteristics of the tensor field near the degenerate point, such as anisotropy and skewness. Note that the matrices that we use to create first- and second-order elements are special instances of (7) in that

$$x = \sqrt{x^2 + y^2}\cos\theta,$$
$$y = \sqrt{x^2 + y^2}\sin\theta,$$
$$x^2 - y^2 = (x^2 + y^2)\cos 2\theta,$$
$$2xy = (x^2 + y^2)\sin 2\theta.$$

Fig. 6 shows some third through eighth-order elements (from left to right), where the elements in the top row have a positive tensor index and the elements in the bottom row have a negative index.

The resulting tensor field is interactively updated and displayed as the user continues to make adjustment to the set of regular and singular elements. The tensor fields in the middle and right of Fig. 1 and in Fig. 12c show examples of designed fields. Colored line segments with arrows indicate the location and orientation of regular elements, and colored boxes indicate the type and location of singular elements. Our implementation of field initialization is similar to the vector field design system of Zhang et al. [35]. Notice this is not the only way to create an initial tensor field. Other methods such as constrained optimization could also be used. The initial tensor field often contains unspecified degenerate points, and our system handles them through topological editing operations that we will describe in the next section. The initial tensor field is then sampled at the vertices and linearly interpolated inside the triangles.

For tensor field analysis, we detect the location and type of degenerate points as well as compute separatrices emanated from wedges and trisectors. For planar tensor fields, we follow closely the algorithms described by Delmarcelle and Hesselink [4] and by Tricoche [28].

6.2 Editing

Our tensor field design system provides three types of editing operations: matrix actions on tensor fields, smoothing, and topological editing. These operations are natural adaptations of the editing operations provided in the vector

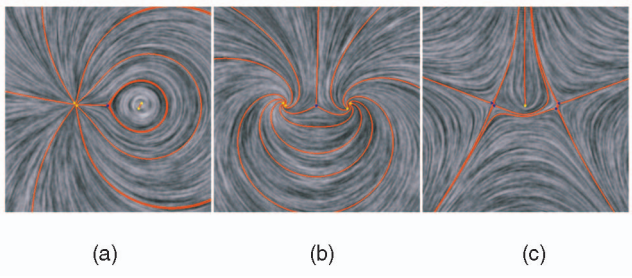

Fig. 7. (a) A tensor field is first (b) rotated by $\pi/4$ and then (c) reflected with respect to the Y-axis.

field design system of Zhang et al. [35]. While the functionalities of our tensor editing operations are similar to their counterpart for vector fields, the implementations are rather different due to the sign ambiguity in tensor fields. One of our major contributions in this article is the use of local conversions between vector fields and tensor fields, which allows us to adapt editing operations for vector fields to tensor fields. We will now describe these editing operations in more detail.

6.2.1 Matrix Actions on Tensor Fields

We consider the action of a nondegenerate 2×2 matrix M on a tensor field $T : (\widetilde{M}(T))(\mathbf{p}) = M^T T(\mathbf{p}) M$. It is straightforward to verify that $\widetilde{M}$ is a group action on the set of deviate matrices if and only if

$$M = \rho \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

or

$$M = \rho \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix},$$

for some $\rho \in \mathbb{R}$ and $\theta \in [0, 2\pi)$. Ignoring scales, we consider the following sets:

$$R = \left\{ \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \right\}$$

and

$$F = \left\{ \begin{pmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{pmatrix} \right\}.$$

Let R_θ be an element in R. R_θ rotates the major (minor) eigenvector field of T by an angle of θ. R_θ does not change the number or location of the degenerate points in T. Furthermore, it maintains the tensor index of any isolated degenerate point, and it can be used to turn a center into a node or a focus with an appropriate rotation θ.

Any element F_θ in F induces a reflection of the major (minor) eigenvector fields of T with respect to $\sin(\frac{\theta}{2})X + \cos(\frac{\theta}{2})Y = 0$ and $\cos(\frac{\theta}{2})X - \sin(\frac{\theta}{2})Y = 0$. F_θ also maintains the number and location of the degenerate points in T. The signs of tensor indices of degenerate points are negated, however, by the operators in F. Performing F_θ twice results in the original field.

Fig. 7 illustrates this on a tensor field shown in the left. It is first rotated by $\pi/4$ to obtain the tensor field shown in the middle, which is then reflected with respect to the Y-axis to obtain the field in the right. Notice that tensor rotations and reflections do not change the number or location of the degenerate points. Rotations maintain tensor indices while reflections negate them.

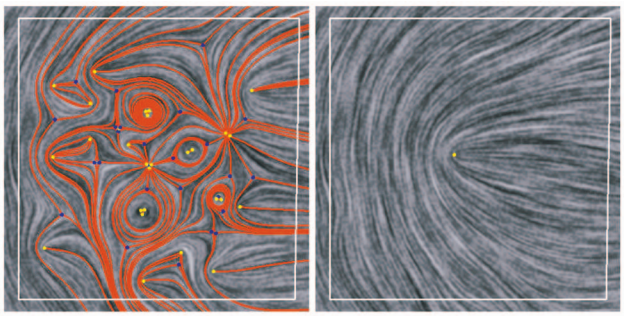

Fig. 8. This figure shows a tensor field before and after user-guided smoothing. (a) The original field has many degenerate points, while (b) the smoothed one has only one. Notice that tensor values outside the smoothing region (the white loop) do not change.

6.2.2 Smoothing

Our system allows tensor field smoothing inside a user-specified region R. By holding tensor values fixed on the boundary of R, the system performs a componentwise discrete Laplacian smoothing to obtain the new tensor values for the interior vertices of R. To be more specific, let

$$\begin{pmatrix} F(\mathbf{p}) & G(\mathbf{p}) \\ G(\mathbf{p}) & -F(\mathbf{p}) \end{pmatrix}$$

and

$$\begin{pmatrix} \overline{F}(\mathbf{p}) & \overline{G}(\mathbf{p}) \\ \overline{G}(\mathbf{p}) & -\overline{F}(\mathbf{p}) \end{pmatrix}$$

be the tensor values before and after smoothing, respectively. Then, $\overline{F} = F$, $\overline{G} = G$ for vertices on the boundary of R. The new values at the interior vertices are determined by

$$\begin{pmatrix} \overline{F}(v_i) \\ \overline{G}(v_i) \end{pmatrix} = \sum_{j \in J} \omega_{ij} \begin{pmatrix} \overline{F}(v_j) \\ \overline{G}(v_j) \end{pmatrix}, \tag{8}$$

where J is the set of index js such that (v_i, v_j) is an edge in the mesh. The weights ω_{ij}s are defined according to the mean-value coordinates of Floater [8] since this choice of weights guarantees ω_{ij} to be nonnegative. This leads to a pair of sparse linear systems, which we solve through an implicit biconjugate solver. Similar smoothing operations have been used in tensor field smoothing [1], [16] and vector field smoothing [26], [35]. Tensor field smoothing allows a user to reduce the geometric complexity of a field as well as the number of degenerate points that it contains. Fig. 8 compares a tensor field (Fig. 8a) with its smoothed version (Fig. 8b). Note the tensor values on and outside the region's boundary (the white loop) do not change.

6.2.3 Topological Editing

Our system provides two topological editing operations: degenerate point pair cancellation and degenerate point

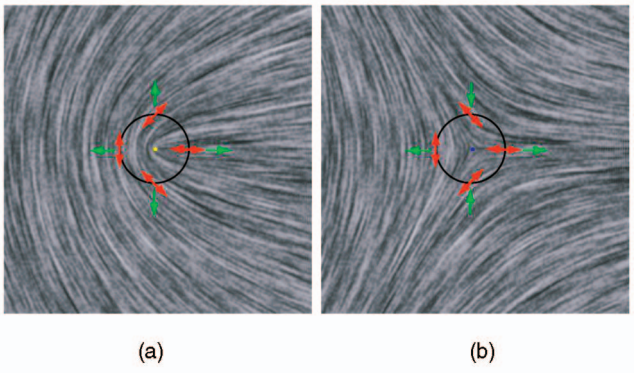

Fig. 9. Tensor-to-vector conversion α (Section 6.2.3) was applied to two example tensor fields. (a) After doubling the angle between the major eigenvector field (indicated by red double arrows) and the X-axis and then converting it into a vector field (green arrows), a wedge is turned in a source. (b) A trisector is transformed into a saddle. Notice this conversion does not change the number and location of the degenerate points.

movement. We will refer to them as *pair cancellation* and *movement* from now on. The pair cancellation operation allows a user to eliminate a pair of unwanted degenerate points with opposite tensor indices. Due to the Poincaré theorem for tensor fields, degenerate points can only be eliminated in pairs so that the total index sum does not change. The movement operation provides control over the location of degenerate points. In our system, both operations are designed to provide topological guarantees in that only the intended degenerate points are affected. There have been several algorithms for pair cancellation, such as the one developed by Tricoche and Scheuermann [29]. To the best of our knowledge, the movement operation is new.

Tricoche and Scheuermann [29] perform degenerate point pair cancellation by first finding a small neighborhood surrounding the degenerate point pair, and then iteratively updating tensor values at the interior vertices so that the tensor index for each cell inside the region is zero. This method requires planar tensor fields, and it is intended for degenerate point pairs that are closer to each other than to other degenerate points. We have set our goals on performing pair cancellation on tensor fields that are defined on either planar domains or curved surfaces, and for degenerate point pairs even when they are not closest neighbors. Zhang et al. [35] provide robust algorithms for pair cancellation and movement of singularities in a surface vector field based on Conley index theory [18]. We wish to adapt their algorithms to surface tensor fields. However, Conley index theory is defined in terms of vector fields, and it is not obvious how it might be extended to tensor fields.

To address the problem, we consider ways of converting a tensor field to a vector field such that any degenerate point in the tensor field becomes a singularity in the vector field. One possibility is to remove the sign ambiguity from the eigenvector field. However, as we have seen in Section 5, places near odd-order degenerate points (wedges, trisectors) will cause discontinuity in the resulting vector field. Therefore, we must look for other ways of converting a tensor field into a vector field. Consider the following mapping α from a deviate tensor field to a vector field:

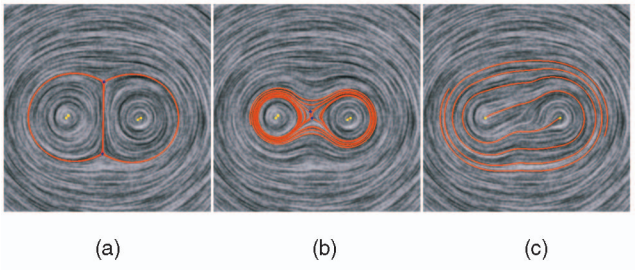

Fig. 10. This figure illustrates the topological editing operations in our system. (a) For this tensor field, a user first moves the two trisectors (blue dots) to be near each other, (b) thereby forming a saddle type of pattern in the region. (c) Next, the user cancels the trisectors with a wedge from each side, resulting in an elongated center pattern. The conversions between tensor and vector fields enable us to reuse algorithms from vector fields, such as those developed by Zhang et al. [35].

$$\alpha : \begin{pmatrix} F & G \\ G & -F \end{pmatrix} \rightarrow \begin{pmatrix} F \\ G \end{pmatrix}. \tag{9}$$

Essentially, α doubles the angle between the major eigenvector field and the X-axis and makes it a vector field. Fig. 9 illustrate this with two examples. In Fig. 9a, a wedge (red double arrows) is turned into a source (green arrows). In Fig. 9b, a trisector is transformed into a saddle. α has the following desirable properties. First, α maps a continuous tensor field T to a continuous vector field $V = \alpha(T)$. The key is the fact that an eigenvector v and its reverse $-v$ correspond to the same vector under α. Notice that this is different from the sign ambiguity removal method that we used for tensor field visualization (Section 5). Second, a point $\mathbf{p}_0$ is a degenerate point of T if and only if $\mathbf{p}_0$ is a singularity of $\alpha(T)$. Third, the tensor index of $\mathbf{p}_0$ with respect to T is half of the vector (Poincaré) index with respect to $\alpha(T)$. The inverse of α is well-defined, which we denote by α^{-1}. While the concepts of α and α^{-1} are simple, they enable ideas and algorithms from vector fields to be applied to tensor fields, especially those that address degenerate points. Tricoche [28] describe yet another relationship between a tensor field and a vector field based on the concept of *covering spaces*. We did not use this relationship because it maps a wedge in the tensor field to a regular point in the vector field.

To perform pair cancellation and movement on a tensor field T, we first convert it to $V = \alpha(T)$. We then perform the corresponding topological editing operation on V to obtain V', which we convert back to a tensor field $T' = \alpha^{-1}(V')$. Fig. 10 illustrates the topological editing operations on a tensor field with two centers and two trisectors (Fig. 10a). First, the trisectors (blue dots) were moved into nearby positions to form a saddle pattern. Next, the trisectors were canceled with a wedge from each side. This results in an elongated center pattern (Fig. 10c).

6.3 Tensor Field Design on Surfaces

Designing tensor fields on surfaces is considerably more difficult than on the plane. First, building basis tensor fields requires a global parameterization, which is often lacking for a surface. Second, tensor field analysis and editing require continuous tensor fields. However, the surface normal for a mesh surface is discontinuous at the vertices and across the edges. As illustrated by Zhang et al. [35], the piecewise linear representation for planar vector fields does

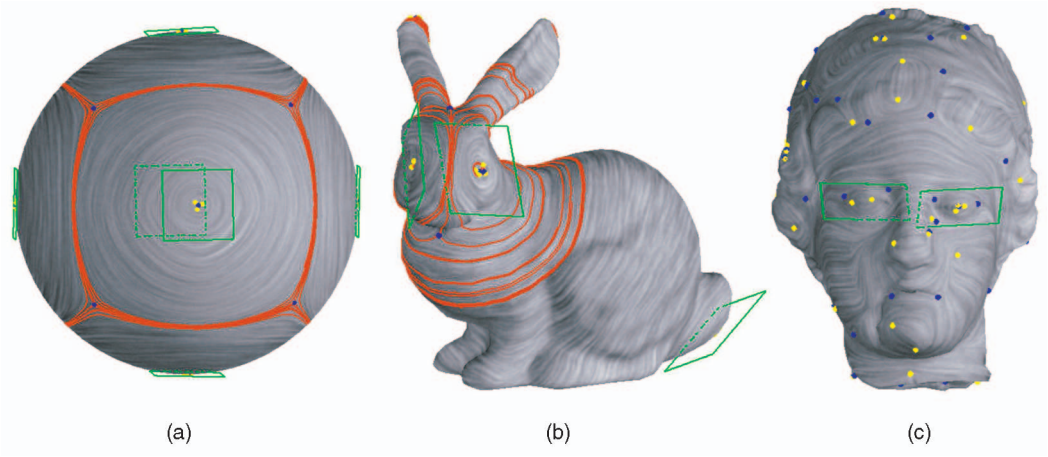

Fig. 11. Example tensor fields designed on various test models. (a) The tensor field was created by placing a center element at each of the six evenly-spaced points on the sphere. The topological skeleton of the major field is very similar to the edges of a cube. (b) The field was created by putting node elements on both sides of the face and on the tail. (c) The tensor field was obtained by combining the curvature tensor field with center elements on Venus' eyes to emphasize them.

not produce continuous vector fields on surfaces. This is also true for tensor fields.

To remedy the problems, we adapt the surface vector field interpolation scheme and design algorithms of Zhang et al. [35] to surface tensor fields, which is based on the concepts of *geodesic polar maps* and *parallel transport*. The adaptation is straightforward due to the similarities between vector fields and tensor fields. We refer our readers to their work for details on the representation and editing of vector fields on surfaces. Example tensor fields on various 3D surfaces are shown in Fig. 11. The colored boxes indicate singular elements, and colored arrows correspond to regular elements. Also shown are the separatrices that correspond to the major field (the red curves). Notice that the tensor field interpolation scheme allows us to *consistently* trace hyperstreamlines based on a surface tensor field without the need for a surface parameterization. In fact, we used the scheme to compute separatrices in a tensor field (Fig. 11), to create hatches in pen-and-ink sketching (Figs. 13 and 14), and to trace lines of curvature in anisotropic remeshing (Fig. 15).

7 Results and Applications

All tensor fields shown in this article were created using our system. In addition, we demonstrate the capability of our system with three graphics applications: painterly rendering, pen-and-ink illustration of surfaces, and anisotropic remeshing.

7.1 Painterly Rendering

Painterly rendering is a well-researched area, and to review all existing algorithms is beyond our scope. In this work, we use the approach of Hertzmann [12] and Hays and Essa [10] with the following modification: Instead of using the image edge field to guide brushstroke orientations, the user creates a tensor field either from scratch or by modifying the image edge field with our design system. Fig. 1 illustrates this with three example tensor fields on the same image of a human's eye. The field shown in the left column is the tensor-based image edge field (TIEF). While it captures the major features in the image, such as the eye and the eyebrow, it is not smooth near the corners of the eye and around the pupil. By adding a center element in the middle of the eye (middle column), the noise around the pupil becomes less noticeable. Finally, the images shown in the right correspond to a tensor field that was created from scratch. Fig. 12 provides additional examples: Mona Lisa (TIEF) (Fig. 12a), Mona Lisa (modified TIEF) (Fig. 12b), and a cat's face (a field designed from scratch) (Fig. 12c). For Mona Lisa, the image edge field contains a wedge on the left side of her forehead that is visually distracting in the painting. Also, part of her left eye was "washed out." By performing degenerate point movement, the wedge was moved from her forehead to the corner of her left eye, removing the artifacts in both areas. In the cat example, the tip of ears can be easily modeled by wedges. In contrast, it would have been difficult to model the ears smoothly using features in vector fields.

7.2 Pen-and-Ink Sketch

Pen-and-ink sketching is an efficient tool in illustrating the shape of an object. Salisbury et al. [22] provide a direction design tool that allows an artist to match the hatch orientations to the features in the input image through a set of functions such as "comb," blending tool, and region fill. These functionalities are vector-based, and topological control is lacking in the system. There have been numerous algorithms on hatch-based illustration of 3D surfaces, and we will only mention those that are most relevant to our work. Girshick et al. [9] demonstrate that principle curvature directions are best in illustrating the shape of a surface. Note that principle curvature directions are the eigenvector fields of the curvature tensor field. Hertzmann and Zorin [11] use principle curvature directions to guide the hatch fields. In their algorithm, two families of evenly spaced streamlines are computed from the principle curvature directions, and hatches are generated based on these streamlines. We adopt a similar approach with one modification: Instead of using the curvature tensor field to guide hatching directions over the surface, we allow the user to design a tensor field from which streamlines are created. There are two advantages to this approach.

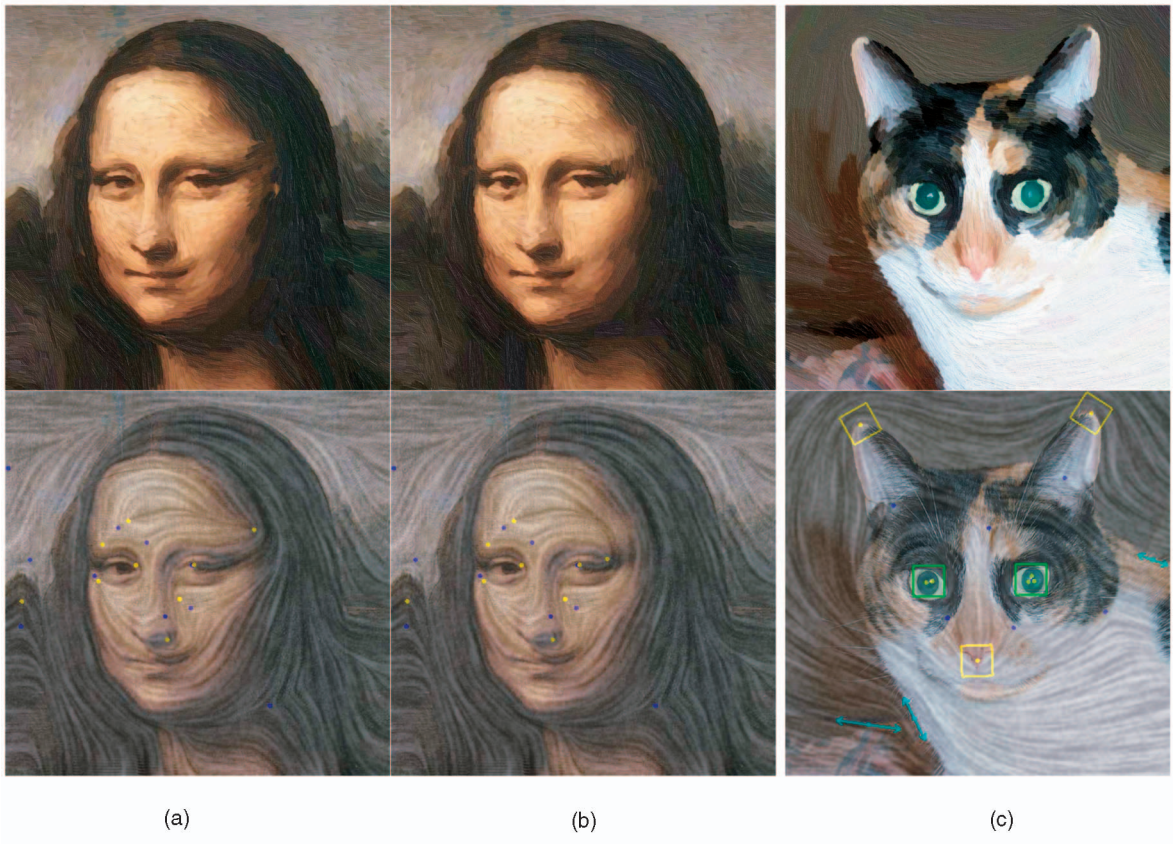

Fig. 12. Additional examples of applying tensor field design to painterly rendering. (a) The tensor-based image edge field contains artifacts on Mona Lisa's left eye and forehead. (b) Through a degenerate point movement operation, a wedge was moved from her forehead to the corner of her eye, and artifacts in both regions were removed. (c) The user created a tensor field from scratch to match the main features. The painterly results were obtained based on the offline high-quality painterly rendering program of Hays and Essa [10].

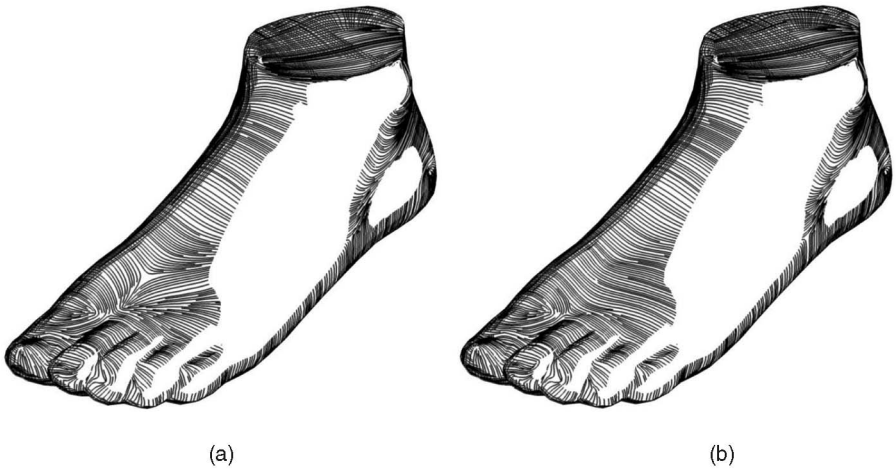

Fig. 13. Pen-and-ink sketching of a foot model using the curvature tensor and a user-modified field. (a) The curvature tensor contains a trisector on the back of the foot near the toes, and it causes an unnatural look in the sketch. (b) Through design, the trisector was canceled with a nearby wedge to obtain this image. Notice the visual artifact has been reduced.

First, while there have been many algorithms for estimating the curvature tensor field on a 3D surface [11], [17], [3], it remains a challenge due to the numerical difficulties associated with polygonal surfaces and the complexity of the underlying shapes. In particular, degenerate points often appear in unnatural places, and they cause visual artifacts in the sketching [11]. While tensor field smoothing can be used to reduce a large percentage of degenerate points, it lacks explicit control over the number and location of degenerate points in the field as smoothing is done to the surface as a whole. Consequently, natural degenerate points are sometimes removed and new degenerate points may emerge in undesirable locations. The user often needs to tune certain smoothing parameters, such as the number of smoothing steps and the speed for smoothing. The tuning process can be considered as *design*. Our approach also involves a design process. However, it provides explicit control over the number, location, and

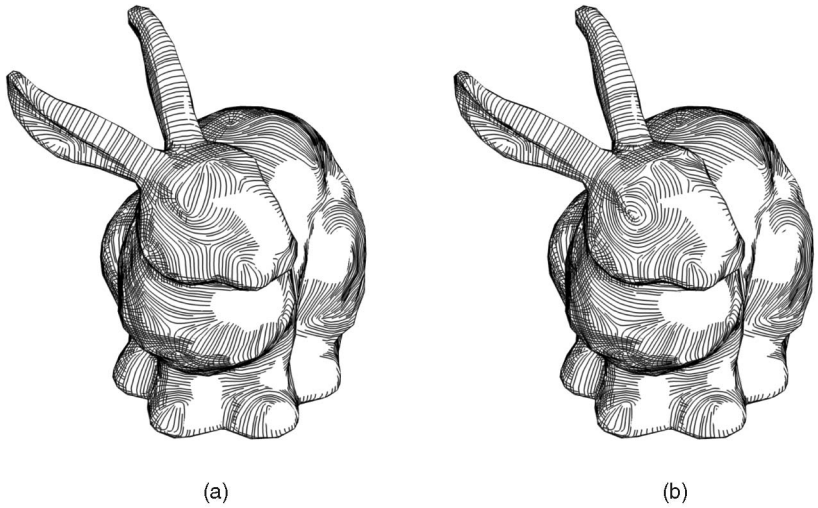

Fig. 14. A pair of center elements were used to create artificial eyes on the bunny for pen-and-ink sketch (b). Notice that the original curvature tensor field (a) does not contain such features.

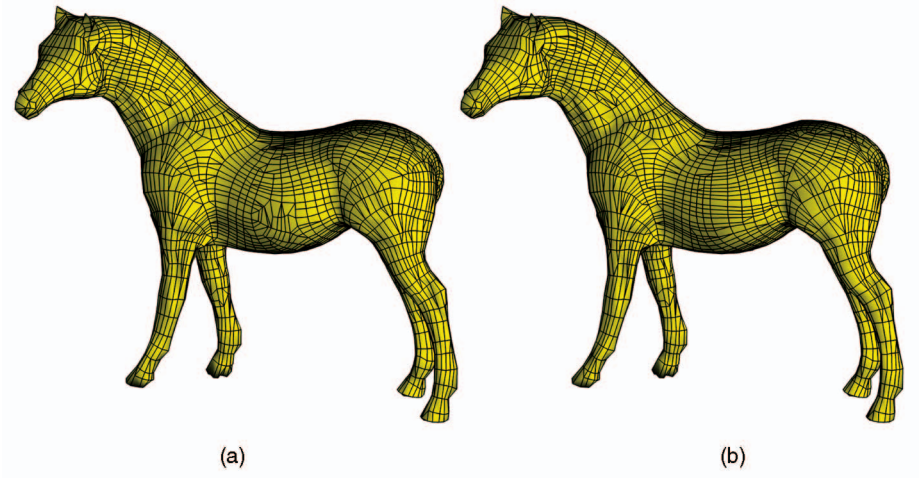

Fig. 15. This figure demonstrates the usefulness of topological editing operations in anisotropic remeshing. (a) The curvature tensor contains a wedge and trisector pair in the middle of the horse's torso, which requires special care during remeshing. (b) Through pair cancellation, this region becomes free of degenerate points.

type of degenerate points in the field. In particular, with degenerate point pair cancellation and movement operations, the user has the guarantee that only intended degenerate points are affected.

Second, tensor field design allows the user to create new features that are either not part of the surface geometry or otherwise difficult to extract. For example, the eyes in the bunny surface are rather difficult to detect automatically due to the relative flatness around the regions. While numerical instabilities can be reduced through better detection algorithms, it is much harder to automate the real-world semantics, such as the bunny's eyes. With design, the user was able to create features without causing problems elsewhere on the model.

In Figs. 13 and 14, we compare pen-and-ink sketch using curvature tensor fields (Figs. 13a and 14a) and with user-designed fields (Figs. 13b and 14b). The designed field for the foot model was produced by removing a trisector on the back of the foot through degenerate point pair cancellation, and the one for the bunny was created by adding center elements to create the illustration of eyes.

7.3 Quad-Dominant Remeshing

Anisotropic remeshing has received much attention recently, thanks to the work of Alliez et al. [1]. Anisotropic remeshing converts an input mesh that is often noisy and over-tessellated into a quad-dominant mesh to achieve an optimal sampling rate. A typical algorithm works as follows: First, a tensor field is computed by either estimating the curvature tensor [1], [16] or through the design of fair Morse functions [7]. Second, a family of evenly spaced streamlines are traced for both the major and minor eigenvector fields. Third, every intersection between any line from each family is found, and dangling edges are removed. Finally, the intersection points are used to produce quad-dominant meshes. Optimal remeshing near degenerate points (or *umbilics*, when the tensor field is the curvature tensor) is more difficult than for regions that are free of degenerate points. Therefore, it is important to control the number and location of the degenerate points in the field. Fig. 15 illustrates the need to use topological editing for anisotropic remeshing. The curvature tensor on the horse surface contains a wedge and trisector pair near

the belly that requires special care during anisotropic remeshing (Fig. 15a). By performing pair cancellation, the same region is degenerate point free, and anisotropic remeshing becomes straightforward (Fig. 15b).

8 Conclusions and Future Work

In this paper, we have identified tensor field design as an important problem in computer graphics, and we advocate that edges in an image be treated as a tensor field rather than a vector field. We present an interactive tensor field design system that allows a user to create a wide variety of tensor fields on planar domains and curved surfaces with relatively little effort. In particular, a degenerate point of any order can be created with our system. We also provide control over the number and location of degenerate points in the field. Our system supports efficient degenerate point pair cancellation and movement operations by converting a tensor field into a vector field with the same set of singularities, which allows us to reuse similar algorithms for vector fields. While the conversions are simple, they can be useful in other types of tensor field operations that involve degenerate points. We also provide an interactive tensor field visualization algorithm for both planar domains and surfaces. To illustrate the benefits of our approach, we have applied tensor field design to three graphics applications: painterly rendering, pen-and-ink sketching of surfaces, and anisotropic remeshing.

There are a number of issues that we wish to investigate further in our system. First, we have so far concentrated on controlling degenerate points in a tensor field. It is a natural next step to consider creating and controlling separatrices and closed orbits. While the conversions between vector fields and tensor fields that we described in Section 6.2.3 relate degenerate points to singularities, they do not provide any information on how a trajectory, such as a separatrix and a periodic orbit in the tensor field, is related to a trajectory in the corresponding vector field. The mapping based on *covering spaces* [28] seems promising in addressing this issue even though it does not always map a degenerate point to a singularity or vice versa. While it seems unlikely that the problem of tensor field design can be turned into vector field design through a "magic" map, vector fields and tensor fields are intrinsically connected in many aspects. Understanding these connections will likely benefit the study of both vector fields and tensor fields. Second, we are investigating techniques for automatic pairing of degenerate points for cancellation. The algorithm of Tricoche and Scheuermann [29] is a good starting point. Third, we wish to extend our system to other domains, such as volumes. Finally, understanding and visualizing high-order tensor data is of great interest to us.

Appendix

Higher-Order Degenerate Points

In this Appendix, we provide a proof for the following statement. Given the following tensor field

$$T = D^N \begin{pmatrix} a\cos(N\theta) + b\sin(N\theta) & c\cos(N\theta) + d\sin(N\theta) \\ c\cos(N\theta) + d\sin(N\theta) & -(a\cos(N\theta) + b\sin(N\theta)) \end{pmatrix},$$

where $D = \sqrt{x^2 + y^2}$ and $ad - bc \neq 0$, T has a unique degenerate point at the origin with a tensor index $sgn(ad - bc)(\frac{N}{2})$. By using the conversions between tensor fields and vector fields that are described in Section 6.2.3, it suffices to show that the vector field

$$V = D^N \begin{pmatrix} a\cos(N\theta) + b\sin(N\theta) \\ c\cos(N\theta) + d\sin(N\theta) \end{pmatrix} \tag{10}$$

has a unique singularity at (0, 0) with a Poincaré index of $sgn(ad - bc)N$.

First, we consider the special case where $a = d = 1$ and $b = c = 0$. Call this vector field V_0. It is straightforward to verify that the vector field $V_0 = D^N \begin{pmatrix} \cos(N\theta) \\ \sin(N\theta) \end{pmatrix}$ has a unique singularity at the origin with a Poincaré index of N.

Next, we consider the general cases in which a, b, c, and d are arbitrary real numbers that satisfy $ad - bc \neq 0$. Then $V = MV_0$ where $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Since the determinant of M is not zero, V also has a unique singularity at the origin. To show that V and V_0 have the same Poincaré index at the origin when ignoring the signs, consider the following map: $\mu : S^1 \rightarrow S^1$, by $\mu(w) = \frac{Mw}{|Mw|}$. Here, S^1 is the unit circle centered at the origin, and w is a unit vector. Notice that $M(S^1)$ is an ellipse centered at the origin, which implies that μ is an automorphism on S^1, i.e., a self-homeomorphism. When traveling along S^1 in counterclockwise fashion once, the image under μ will also cover S^1 once either counterclockwise when $ad - bc > 0$ or clockwise when $ad - bc < 0$. Therefore, the Poincaré index for the singularity of V is $sgn(ad - bc)N$.

Acknowledgments

The authors would like to thank the following people and groups for the 3D models they provided: Mark Levoy and the Stanford Graphics Group, Andrzej Szymczak, Cyberware, and the AIM@SHAPE Shape Repository. Images used in this article are courtesy of FreeFoto.com and Pics4Learning.com. The authors also appreciate the discussions they had with Konstantin Mischaikow and the help from Irfan Essa. Furthermore, the authors are grateful for the help by Spencer Reynolds in preparing the audio and video production. Finally, they wish to thank the anonymous reviewers for their valuable comments and suggestions. This work is funded by US National Science Foundation grants DMS-0138420 and CCF-0546881.

References

[1] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun, "Anisotropic Polygonal Remeshing," *ACM Trans. Graphics (Proc. SIGGRAPH '03)*, vol. 22, no. 3, pp. 485-493, July 2003.

[2] B. Cabral and C. Leedom, "Imaging Vector Fields Using Line Integral Convolution," *Computer Graphics Proc., Ann. Conf. Series (SIGGRAPH '93)*, pp. 263-270, 1993.

[3] D. Cohen-Steiner and J.M. Morvan, "Restricted Delaunay Triangulations and Normal Cycle," *Proc. 19th Ann. ACM Symp. Computational Geometry*, pp. 237-246, 2003.

[4] T. Delmarcelle and L. Hesselink, "Visualizing Second-Order Tensor Fields with Hyperstream Lines," *IEEE Computer Graphics and Applications*, pp. 25-33, 1993.

[5] T. Delmarcelle and L. Hesselink, "The Topology of Symmetric, Second-Order Tensor Fields," *Proc. IEEE Visualization Conf.*, pp. 140-147, 1994.

[6] T. Delmarcelle, "The Visualization of Second-Order Tensor Fields," PhD thesis, Stanford Applied Physics, 1994.
[7] S. Dong, S. Kircher, and M. Garland, "Harmonic Functions for Quadrilateral Remeshing of Arbitrary Manifolds," *Computer Aided Geometry Design,* 2005.
[8] M. Floater, "Mean Value Coordinates," *Computer Aided Geometric Design,* vol. 20, no. 1, pp. 19-27, 2003.
[9] A. Girshick, V. Interrante, S. Haker, and T. Lemoine, "Line Direction Matters: An Argument for the Use of Principal Directions in 3D Line Drawings," *Proc. First Int'l Symp. Non-Photorealistic Animation and Rendering (NPAR '00),* pp. 43-52, 2000.
[10] J.H. Hays and I. Essa, "Image and Video Based Painterly Animation," *Proc. Third Int'l Symp. Non-Photorealistic Animation and Rendering (NPAR '04),* pp. 113-120, 2004.
[11] A. Hertzmann and D. Zorin, "Illustrating Smooth Surfaces," *Computer Graphics Proc., Ann. Conf. Series (SIGGRAPH '00),* pp. 517-526, 2000.
[12] A. Hertzmann, "Painterly Rendering with Curved Brush Strokes of Multiple Sizes," *Computer Graphics Proc., Ann. Conf. Series (SIGGRAPH '98),* pp. 453-460, 1998.
[13] V. Interrante, "Illustrating Surface Shape in Volume Data via Principal Direction-Driven 3D Line Integral Convolution," *Computer Graphics Proc., Ann. Conf. Series (SIGGRAPH 1997),* pp. 109-116, 1997.
[14] R.S. Laramee, B. Jobard, and H. Hauser, "Image Space Based Visualization of Unsteady Flow on Surfaces," *Proc. IEEE Visualization Conf.,* pp. 131-138, 2003.
[15] P. Litwinowicz, "Processing Images and Video for an Impressionist Effect," *Computer Graphics Proc., Ann. Conf. Series (SIGGRAPH 1997),* pp. 407-414, 1997.
[16] M. Marinov and L. Kobbelt, "Direct Anisotropic Quad-Dominant Remeshing," *Proc. 12th Pacific Conf. Computer Graphics and Applications (PG '04),* pp. 207-216, 2004.
[17] M. Meyer, M. Desbrun, P. Schröder, and A.H. Barr, "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds," *Proc. Workshop Visualization and Math. (VisMath),* 2002.
[18] K. Mischaikow and M. Mrozek, "Conley Index," *Handbook of Dynamic Systems,* second ed., North-Holland, pp. 393-460, 2002.
[19] X. Ni, M. Garland, and J.C. Hart, "Fair Morse Functions for Extracting the Topological Structure of a Surface Mesh," *ACM Trans. Graphics (Proc. SIGGRAPH '04),* vol. 23, no. 3, pp. 613-622, 2004.
[20] E. Praun, A. Finkelstein, and H. Hoppe, "Lapped Textures," *Computer Graphics Proc., Ann. Conf. Series (SIGGRAPH '00),* pp. 465-470, 2000.
[21] A. Rockwood and S. Bunderwala, "A Toy Vector Field Based on Geometric Algebra," *Proc. Application of Geometric Algebra in Computer Science and Eng. (AGACSE '01),* pp. 179-185, 2001.
[22] M.P. Salisbury, M.T. Wong, J.F. Hughes, and D.H. Salesin, "Orientable Textures for Image-Based Pen-and-Ink Illustration," *Computer Graphics Proc., Ann. Conf. Series (SIGGRAPH '97),* pp. 401-406, 1997.
[23] D. Stalling, H.C. Hege, "Fast and Resolution Independent Line Integral Convolution," *Computer Graphics Proc., Ann. Conf. Series (SIGGRAPH '95),* pp. 249-256, 1995.
[24] J. Stam, "Flows on Surfaces of Arbitrary Topology," *ACM Trans. Graphics (Proc. SIGGRAPH '03),* vol. 22, no. 3, pp. 724-731, 2003.
[25] H. Theisel, "Designing 2D Vector Fields of Arbitrary Topology," *Computer Graphics Forum (Proc. Eurographics '02),* vol. 21, no. 3, pp. 595-604, 2002.
[26] Y. Tong, S. Lombeyda, A. Hirani, and M. Desbrun, "Discrete Multiscale Vector Field Decomposition," *ACM Trans. Graphics, (Proc. SIGGRAPH '03),* vol. 22, no. 3, pp. 445-452, 2003.
[27] X. Tricoche, G. Scheuermann, and H. Hagen, "Scaling the Topology of Symmetric Second Order Tensor Fields," *Proc. US Nat'l Science Foundation/Dept. of Energy Lake Tahoe Workshop Hierarchical Approximation and Geometrical Methods for Scientific Visualization,* 2001.
[28] X. Tricoche, "Vector and Tensor Field Topology Simplification, Tracking, and Visualization," PhD thesis, Universität Kaiserslautern, 2002.
[29] X. Tricoche and G. Scheuermann, "Topology Simplification of Symmetric, Second-Order 2D Tensor Fields," *Geometric Modeling Methods in Scientific Visualization,* B. Hamann, H. Müller, and H. Hagen, eds., Springer, 2003.
[30] G. Turk, "Texture Synthesis on Surfaces," *Computer Graphics Proc., Ann. Conf. Series (SIGGRAPH '01),* pp. 347-354, 2001.
[31] T. Urness, V. Interrante, I. Marusic, E. Longmire, and B. Ganapathisubramani, "Effectively Visualizing Multi-Valued Flow Data Using Color and Texture," *Proc. IEEE Visualization Conf.,* pp. 115-121, 2003.
[32] J.J. van Wijk, "Image Based Flow Visualization," *ACM Trans. Graphics (Proc. SIGGRAPH '02),* vol. 21, no. 3, pp. 745-754, 2002.
[33] J.J. van Wijk, "Image Based Flow Visualization for Curved Surfaces," *Proc. IEEE Visualization Conf.,* G. Turk, J. van Wijk, and R. Moorhead, eds., pp. 123-130, 2003.
[34] L.Y. Wei and M. Levoy, "Texture Synthesis over Arbitrary Manifold Surfaces," *Computer Graphics Proc., Ann. Conf. Series, (SIGGRAPH '01),* pp. 355-360, 2001.
[35] E. Zhang, K. Mischaikow, and G. Turk, "Vector Field Design on Surfaces," *ACM Trans. Graphics,* vol. 25, no. 4, 2006.
[36] X. Zheng and A. Pang, "Hyperlic," *Proc. IEEE Visualization Conf.,* pp. 249-256, 2003.

Eugene Zhang received the PhD degree in computer science in 2004 from Georgia Institute of Technology. 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 computational topology. He is a member of the IEEE.

James Hays received the BS degree in 2003 from Georgia Institute of Technology, where he worked with Irfan Essa on nonphotorealistic rendering. He is pursuing the PhD degree in computer science at Carnegie Mellon University on a US National Science Foundation Graduate Fellowship. He works with Yanxi Liu on regular texture analysis and synthesis as well as symmetry detection.

Greg Turk received the PhD degree in computer science in 1992 from the University of North Carolina (UNC) at Chapel Hill. He was a postdoctoral researcher at Stanford University for two years, followed by two years as a research scientist at UNC Chapel Hill. He is currently an associate professor at the Georgia Institute of Technology, where he is a member of the College of Computing and the Graphics, Visualization, and Usability Center. His research interests include computer graphics, computer vision, and scientific visualization. He is a member of the IEEE.