Vector Field Design on Surfaces

EUGENE ZHANG
Oregon State University
KONSTANTIN MISCHAIKOW
Rutgers University
and
GREG TURK
Georgia Institute of Technology

Vector field design on surfaces is necessary for many graphics applications: example-based texture synthesis, nonphotorealistic rendering, and fluid simulation. For these applications, singularities contained in the input vector field often cause visual artifacts. In this article, we present a vector field design system that allows the user to create a wide variety of vector fields with control over vector field topology, such as the number and location of singularities. Our system combines basis vector fields to make an initial vector field that meets user specifications.

The initial vector field often contains unwanted singularities. Such singularities cannot always be eliminated due to the Poincaré-Hopf index theorem. To reduce the visual artifacts caused by these singularities, our system allows the user to move a singularity to a more favorable location or to cancel a pair of singularities. These operations offer topological guarantees for the vector field in that they only affect user-specified singularities. We develop efficient implementations of these operations based on *Conley index theory*. Our system also provides other editing operations so that the user may change the topological and geometric characteristics of the vector field.

To create continuous vector fields on curved surfaces represented as meshes, we make use of the ideas of *geodesic polar maps* and *parallel transport* to interpolate vector values defined at the vertices of the mesh. We also use geodesic polar maps and parallel transport to create basis vector fields on surfaces that meet the user specifications. These techniques enable our vector field design system to work for both planar domains and curved surfaces.

We demonstrate our vector field design system for several applications: example-based texture synthesis, painterly rendering of images, and pencil sketch illustrations of smooth surfaces.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Geometric algorithms, languages, and systems*

General Terms: Algorithms

Additional Key Words and Phrases: Vector field design, topology, surfaces, computational geometry, nonphotorealistic rendering, texture synthesis

This research is funded by NSF grants DMS-0138420 and DMS-0107396.

Authors' addresses: E. Zhang, (at time of submission) College of Computing and GVU Center, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30332-0280; (current) School of Electrical Engineering and Computer Science, Oregon State University, 2111 Kelley Engineering Center, Corvallis, OR 97331; email: zhange@eecs.oregonstate.edu; K. Mischaikow, (at time of submission) Center for Dynamical Systems and Nonlinear Studies, School of Mathematics, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30332-0280; (current) Department of Mathematics, Hill Center—Busch Campus, Rutgers, The State University of New Jersey, Piscataway, NJ 08854-8019; email: mischaik@math.rutgers,edu; G. Turk, College of Computing and GVU Center, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA 30332-0280; email: turk@cc.gatech.edu.

1. INTRODUCTION

Many graphics applications require an input vector field. For instance, example-based texture synthesis makes use of a vector field to define local texture orientation and scale [Praun et al. 2000; Turk 2001; Wei and Levoy 2001]. In nonphotorealistic rendering, vector fields are used to guide the orientation of brush strokes [Hertzmann 1998] and hatches [Hertzmann and Zorin 2000]. In fluid simulation, the external force is a vector field which need not correspond to any physical phenomenon and can exist on synthetic 3D surfaces [Stam 2003]. A vector field design system enables these applications to achieve many different visual effects by using different input fields. It can also be used to create datasets for testing the efficiency and correctness of a vector field visualization technique [van Wijk 2002, 2003].

Vector field design refers to creating a continuous vector field on a 3D surface based on user specifications or application-dependent requirements. It is different from vector field simplification, which is used to reduce the complexity of large and noisy datasets while maintaining their major features. In vector field design, both adding and removing features may be required.

There are several challenges when it comes to designing vector fields. First, a design system should enable the user to create a wide variety of vector fields with relatively little effort. Most existing design systems generate some subclasses, such as *curl-free* and *divergence-free* vector fields. This limits their potential applications. Second, the user often needs control over vector field topology, such as the number and location of the singularities. As pointed out in Praun et al. [2000] and Hertzmann and Zorin [2000], this is necessary for applications such as example-based texture synthesis and nonphotorealistic rendering, in which unwanted singularities often cause visual artifacts. Figure 1 illustrates this with an example from texture synthesis, in which a sink (dot) in the middle of the bunny's tail (left) causes the synthesis pattern to break up (right).

In addition, many applications of vector field design are based on 3D mesh surfaces, such as texture synthesis, fluid simulation, and artistic illustration of surfaces. A typical mesh consists of vertices, edges, and triangles. In order to control vector field topology on a mesh, we need continuous vector fields for which we can perform particle tracing that follows the flow in a consistent manner. However, surface normal and tangent planes are discontinuous at the vertices and across the edges, and the definition of vector field continuity from smooth surfaces does not apply. Furthermore, the popular piecewise linear representation [Tricoche et al. 2001] produces continuous vector fields *only* when the mesh represents a planar domain. Stam [2003] uses subdivision surfaces to ensure vector field continuity. However, it is difficult to extract and control vector field topology with this representation because of its complexity. Also, subdivision surfaces often incur higher computational costs than polygonal meshes.

In this article, we present a vector field design system for surfaces. This system employs a three-stage pipeline: initialization, analysis, and editing. In the initialization stage, the user can quickly create a vector field by using basis flow fields. Next, the system performs geometric and topological analysis on the initial field and provides visual feedback to the user. In the third stage, the user makes controlled editing operations to the current vector field, such as moving a singularity (singularity *movement*) or cancelling a pair of singularities (singularity *pair cancellation*). This process of iterative analysis and editing is repeated until the user is satisfied with the result.

Our system enables the user to create a wide variety of vector fields (curl-free, divergence-free, and generic) by using basis fields of different kinds. It also provides control over vector field topology, such as the number and location of singularities. We provide efficient algorithms for both singularity pair cancellation and singularity movement based on ideas from Conley index theory, which is more general and powerful than the well-known Poincaré index theory. To enable these operations to work for generic vector fields, as opposed to only curl-free vector fields, we use *flow rotations* and *reflections* to handle the numerical instabilities associated with regions of a high curl and regions near a saddle.

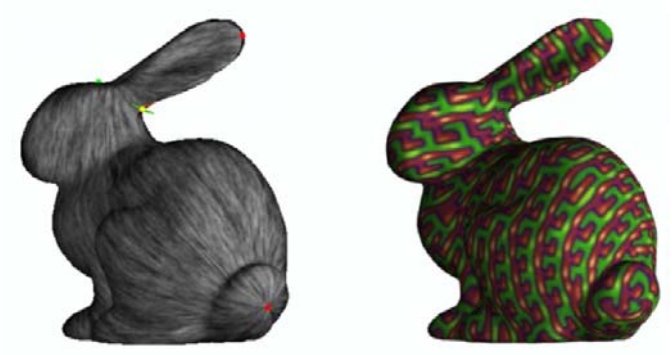

Fig. 1. This figure highlights the need for control over vector field topology in texture synthesis. The input vector field contains a singularity at the center of the bunny's tail (left), and it causes the synthesis patterns to break up (right).

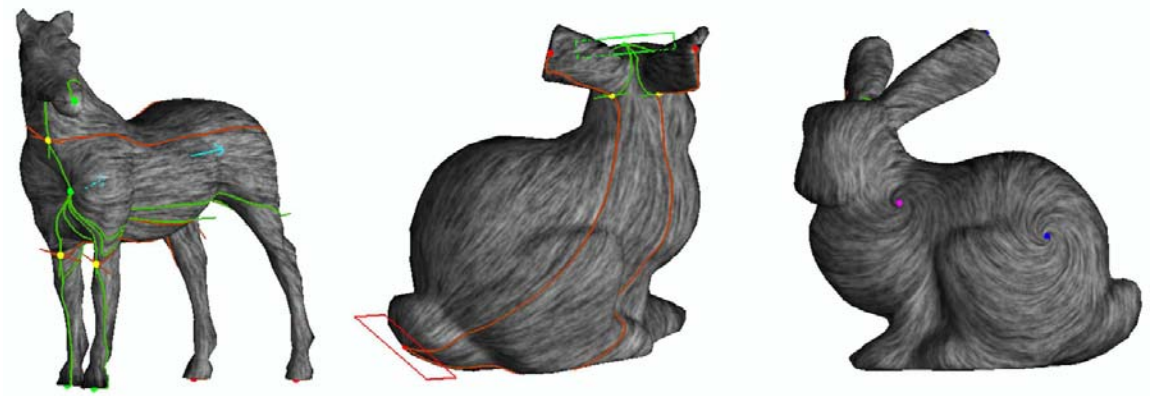

Fig. 2. Example vector fields created by our design system. The field shown in the right was used to guide texture synthesis (Figure 22, upper-right).

We have also developed a novel piecewise interpolation scheme for meshes that guarantees the creation of continuous vector fields based on values defined at the vertices. This scheme supports efficient analysis and editing. In addition, we will describe a new way to build basis vector fields on surfaces from user specifications. The ideas behind both the interpolation scheme and construction of surface basis fields are based on the concepts of *geodesic polar maps* and *parallel transport* from classical differential geometry. Figure 2 shows some vector fields created using our system. The dots correspond to singularities, and the curves indicate their connectivity.

The remainder of the article is organized as follows. We first review the relevant background on vector fields in Section 2. Then, in Section 3 we review existing vector field design systems and simplification techniques. We present our design system for planar domains in Sections 4 and 5, and its adaptation to 3D mesh surfaces in Section 6. Section 7 provides some results of applying our system to various graphics applications, such as painterly rendering, pencil sketches of surfaces, and texture synthesis. Finally, we summarize our contributions and discuss some possible future work in Section 8.

2. BACKGROUND ON VECTOR FIELDS

In this section, we review some basic facts about vector fields. A *vector field* V for a manifold surface $\mathbf{S}$ is a smooth vector-valued function that associates to every point $\mathbf{p} \in \mathbf{S}$ a tangent vector $V(\mathbf{p})$. A vector

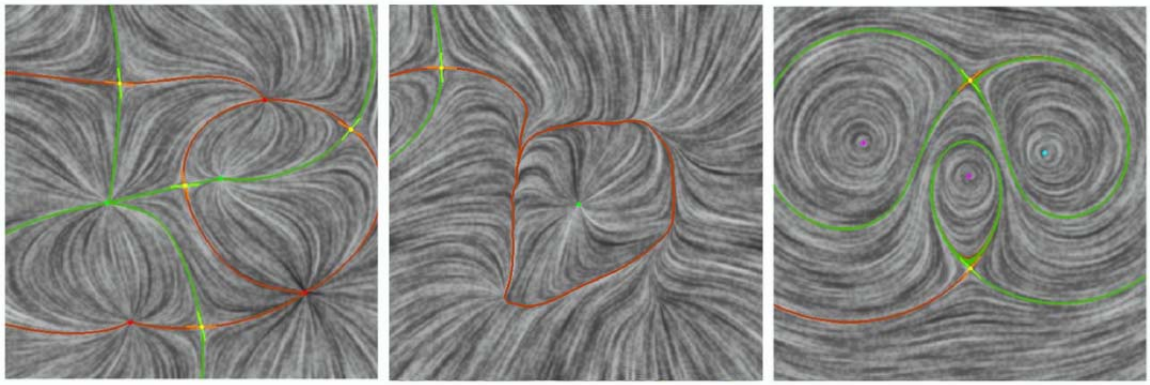

Fig. 3. This figure illustrates three vector fields that are curl-free (left), divergence-free (right), and neither (middle). Singularities are depicted as colored dots, and principle directions for saddles are drawn as crosses. Furthermore, incoming separatrices for saddles are shown in green while outgoing separatrices are shown in red. The vector field in the middle contains a periodic orbit that separates the flow inside from the flow outside. The visualization technique is based on that of van Wijk [2002].

field defines a system of differential equations: $\frac{d\mathbf{p}}{dt} = V(\mathbf{p})$. With appropriate restrictions on V, for each point $\mathbf{p}_0 \in \mathbf{S}$, there exists a solution $\mathbf{p} : \mathbb{R} \to \mathbf{S}$ with the property that $\mathbf{p}(0) = \mathbf{p}_0$ [Hale and Kocak 1991; Hirsch and Smale 1974]. As we will be interested in studying multiple solutions simultaneously, it is useful to introduce the notion of the flow induced by V that is a continuous function $\varphi : \mathbb{R} \times \mathbf{S} \to \mathbf{S}$ with the property that $\varphi(t, \mathbf{p}_0) = \mathbf{p}(t)$. The set $\{\mathbf{p}(t) \mid t \in \mathbb{R}\} = \varphi(\mathbb{R}, \mathbf{p}_0)$ is called the *trajectory* through $\mathbf{p}_0$. Uniqueness of solutions to ordinary differential equations guarantees that the set of trajectories forms an equivalence relationship on $\mathbf{S}$. In particular, if $\mathbf{q}_0$ belongs to the trajectory of $\mathbf{p}_0$, then $\mathbf{p}_0$ belongs to the trajectory of $\mathbf{q}_0$. This implies that $\mathbf{S}$ can be decomposed into the set of all trajectories. Some trajectories are of particular significance, such as singularities.

A singularity $\mathbf{p}_0 \in S$ is a point such that $V(\mathbf{p}_0) = 0$. Observe that the trajectory through a singularity consists of a single point. For many of our calculations, we will want to use a singularity classification based on the *local linearization* of the vector field. For simplicity, let V be a vector field defined for some planar domain $D \subset \mathbb{R}^2 = \{(x, y) \mid x, y \in \mathbb{R}\}$ such that $V(x, y) = (F(x, y)\ G(x, y))$. The local linearization at a point $\mathbf{p}_0$ is: $V^*(\mathbf{p}) = V(\mathbf{p_0}) + DV(\mathbf{p}_0)(\mathbf{p} - \mathbf{p}_0)$, where $DV = \begin{pmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} \\ \frac{\partial G}{\partial x} & \frac{\partial G}{\partial y} \end{pmatrix}$ is the *Jacobian* of V. A singularity $\mathbf{p}_0$ is linear if $DV(\mathbf{p}_0)$ has a full rank. For the remainder of this article, we will assume that $\mathbf{p}_0$ is a linear singularity. Results from linear algebra state that the two eigenvalues of $DV(\mathbf{p}_0)$ are either both real numbers or a pair of conjugate complex numbers. In the first case, $\mathbf{p}_0$ is a *source* if both eigenvalues are positive, a *sink* if both are negative, or a *saddle* if one is positive and the other is negative. In the second case, $\mathbf{p}_0$ is either a *center* if the real part of both eigenvalues is zero, or a *focus* otherwise.

Other trajectories of particular importance are *separatrices* and *periodic orbits*. A separatrix is a trajectory for which the limit as $t \to \infty$ or $t \to -\infty$ of the solution function $\mathbf{p}(t)$ is a saddle. For planar vector fields, the vector field *topology* is determined by the set of singularities, separatrices, and periodic orbits. Figure 3 illustrates these special trajectories with three vector fields. Singularities are highlighted by the colored dots: sources (green), sinks (red), centers (cyan or magenta, depending on the orientations), and saddles (yellow). Furthermore, incoming and outgoing separatrices are colored in green and red, respectively. The vector field in the middle contains a periodic orbit.

Two useful *analytic* characterizations of a vector field are its curl and divergence. Divergence measures the difference between the amount of flow leaving and approaching the measurement point. For instance, the divergence is positive for a source and negative for a sink. Curl measures the amount of flow that circles around the measurement point. The distributions of curl and divergence in the domain can help us understand the geometric structure of the trajectories. The two extreme cases are curl-free vector fields, in which the curl is zero everywhere, and divergence-free vector fields, in which the divergence is zero everywhere. It should be noted that a typical vector field is neither curl-free nor divergence-free. Figure 3 shows three vector fields of different analytical behaviors. The vector field shown in the left is curl-free. In this case the typical singularities are sources, sinks, and saddles. Furthermore, the separatrices divide the domain into a number of combinatorial quadrilaterals called *basins*. The boundary of each basin consists of a source, a sink, and one or two saddles in between them. Inside each basin, all the trajectories leave the same source and approach the same sink. The vector field in the right is divergence-free, in which the typical singularities are centers and saddles. The separatrices divide the domain into a number of bounded regions. Inside each region is a family of periodic orbits that circle around the same center. A generic vector field is shown in the middle, which is neither curl-free nor divergence-free and may contain periodic orbits.

2.1 Topological Descriptions of Vector Fields

The vector field design problem requires that the user be able to control the trajectories of a vector field both locally and globally. Doing so requires the introduction of a topological characterization of vector fields. In this section, we review a well-known topological descriptor, the *Poincaré index*, and a more general characteristic, the *Conley index*.

A singularity $\mathbf{p}_0$ is *isolated* if there exists an open neighborhood U of $\mathbf{p}_0$ with the property that $\mathbf{p}_0$ is the unique singularity in the interior of U. An isolated singularity $\mathbf{p}_0$ can be characterized by its Poincaré index, which is defined in terms of the *winding number* for the *Gauss map*.

Definition 2.1. Let V be a vector field defined on some planar domain D, and let $D_0 \subset D$ be the zero set for V. The Gauss map $\alpha : D \setminus D_0 \to S^1$ is defined as $\alpha(x) = \frac{V(x)}{|V(x)|}$.

For a simple closed curve $\Gamma \subset D \setminus D_0$, the Gauss map α induces a continuous map $\alpha|_\Gamma$. If we travel along Γ in the positive direction once, the image under $\alpha|_\Gamma$ necessarily covers the unit circle S^1 an integer number of times counting orientation. This integer is the winding number of V along Γ. The Poincaré index of an isolated singularity $\mathbf{p}_0$ is the winding number of any simply connected curve that encloses $\mathbf{p}_0$ and contains no other singularities, either in its interior or on the boundary. Denote this number as $\kappa(V; \mathbf{p}_0)$. The Poincaré index is $+1$ for sources, sinks, centers, and foci. It is -1 for saddles, and 0 for regular points. The Poincaré-Hopf theorem links the topology of a vector field to that of the underlying domain in the following way. Let S be a closed orientable manifold with a Euler characteristic E. Furthermore, let V be a continuous vector field defined on S with only isolated singularities $\mathbf{p}_1, \ldots, \mathbf{p}_n$. Then $\sum_{i=1}^{n} \kappa(V; \mathbf{p}_i) = E$. An immediate corollary of the Poincaré-Hopf theorem is that given a particular vector field V, if we want to remove a singularity of a positive or negative Poincaré index, then we must simultaneously remove a singularity of the opposite sign. In fact, for a two-manifold, a zero total Poincaré index for a region R guarantees that it is possible to replace the vector field inside R with a singularity-free vector field.

The Poincaré index is a powerful tool for describing singularities. However, it does not distinguish between sources and sinks, nor does it provide information about periodic orbits and separatrices. Figure 4 illustrates this with a number of examples. First, the Poincaré indices for the disk in cases (b) and (d) are both one. When simplifying the vector field inside the disk such that only one singularity remains, the Poincaré index alone cannot predict whether the singularity is a source or sink. Second,

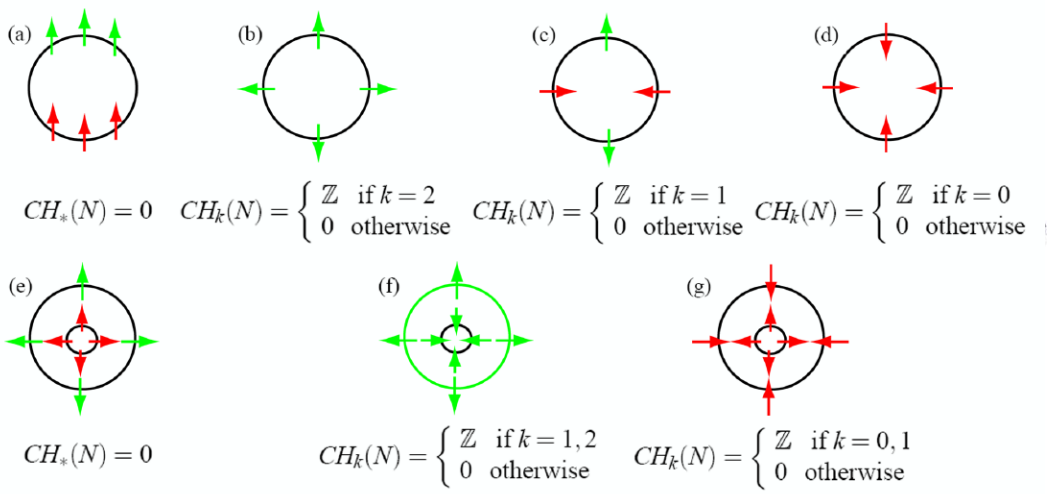

Fig. 4. Seven basic scenarios of isolating blocks and their associated Conley indices. The Conley index can be used to distinguish between sources and sinks (b) and (d), and it provides information on periodic orbits (e), (f), (g). By comparison, the Poincaré indices are the same for cases (b) and (d), and for (e)–(g). Cases (a), (b) and (e) are of particular interest, since they are used in topological editing operations (Section 5.3).

the Poincaré indices for the ring-shaped region in (e)–(g) are zero. Yet, the three vector fields have very different characteristics. For example, when vector fields inside the region are singularity-free, the vector fields in (f) and (g) necessarily contain a periodic orbit, while the vector field in (e) does not. The design of vector fields requires the imposition of additional quantitative information, including the location of the singularities, periodic orbits, and separatrices, and control of the smoothness and/or curvature of the vector field. In this work, we have chosen to control singularities for vector fields defined on two-manifolds for which the Poincaré index is insufficient, since it does not distinguish between sources and sinks. In addition, we wish to set up a framework that has the potential of being extended to control separatrices and periodic orbits. For these reasons, we borrow basic ideas from Conley index theory and provide implementations for our topological editing operations (Section 5.3) according to this theory.

The Conley index is defined in the context of arbitrary vector fields that produce continuous flows. It possesses the continuation properties of the Poincaré index, while being able to distinguish between sinks and sources. It provides sufficient conditions on whether two singularities can cancel. More importantly, it can be used to identify periodic orbits and separatrices, and to indicate whether two periodic orbits can be cancelled.

The following concept is the starting point for Conley index theory. Given a region $N \subset \mathbf{S}$, let ∂N denote the boundary of N. A compact set N is an *isolating neighborhood* if for every $\mathbf{p} \in \partial N, \varphi(\mathbb{R}, \mathbf{p}) \not\subset N$, that is, the trajectory of any point on ∂N leaves eventually either in forward or backward time. The set of boundary points which leave or enter N immediately can be characterized, respectively, by

$$N^- := \{\mathbf{p} \in \partial N \mid \varphi([0,t),\mathbf{p}) \not\subset N,\ \forall t > 0\}, \quad N^+ := \{\mathbf{p} \in \partial N \mid \varphi((t,0],\mathbf{p}) \not\subset N,\ \forall t < 0\}. \tag{1}$$

A compact set N is an *isolating block* if for each boundary point, there is either a forward or backward trajectory that immediately leaves the region, that is, $\partial N = N^- \cup N^+$. Observe that an isolating block is a special case of an isolating neighborhood.

Given an isolating block N for a vector field V, its Conley index is defined to be the relative homology [Kaczynski et al. 2004] of N with respect to N^-, that is, $CH_*(N) := H_*(N, N^-)$. Here, $CH_*(N) = \{CH_k(N) \mid k = 0, 1, 2, \dots\}$ is a collection of *groups*. When the domain of the vector field is a surface, $CH_k(N) = 0$ for $k \geq 3$ (see Conley [1978], Mischaikow [2002], Mischaikow and Mrozek [2002] for further details and references). For the purposes of this article, the computation of this index is fairly simple, since our isolating block N will always take the form of a polygonal region and N^- will be a finite number of disjoint sets consisting of the boundary edges of N. Idealized isolating blocks and their associated Conley indices are indicated in Figure 4. Cases (a) and (e) have the trivial Conley index, and (b), (c), and (d) have the Conley index of a source, saddle, and sink, respectively. Of particular interest are cases (a), (b), and (e). We construct regions of these types for topological editing operations (Section 5.3).

3. PREVIOUS WORK

Vector field *analysis* and *visualization* have been well-studied, and a good survey is available in Hauser et al. [2002]. On the other hand, vector field *design* is far less explored. We will review existing design systems for planar domains and 3D surfaces. In addition, because our system allows the user to perform vector field simplification both geometrically and topologically, we also review existing simplification techniques.

3.1 Existing Vector Field Design Systems

There has been some prior work in creating vector fields on surfaces. In all the instances that we know, such systems have been created in a quick manner to generate vector fields for a particular application, such as texture synthesis [Praun et al. 2000; Turk 2001; Wei and Levoy 2001] and fluid simulation [Stam 2003], or for testing a vector field visualization technique [van Wijk 2003]. Furthermore, the details of these design systems have not been published.

There are several approaches for creating a surface vector field. In the first approach, a 3D vector field is specified and projected onto the surface [van Wijk 2003]. This is similar to performing texture synthesis on surfaces through solid textures. While it is simple and fast, achieving control in this method is hard. In the second approach, the user specifies desired vector values at a few locations on the surface, and the system performs relaxation to obtain a global surface vector field [Turk 2001; Wei and Levoy 2001]. This can be seen as a diffusion process in which the desired values are smoothly propagated from seed points to the rest of the surface. In the third approach, the user again specifies the vector values at a few places on the surface. Then, a global vector field is constructed by interpolating these locations using Gaussian radial basis functions over the surface [Praun et al. 2000]. Another way of creating surface vector fields is to parameterize the surface and define vector fields in the parametric domain [Stam 2003]. These design systems do not provide control over vector field topology, such as the number and location of the singularities.

For planar domains, vector field design systems based on topological information have been developed. Van Wijk develops such a design system to demonstrate his image-based flow visualization technique [2002]. In this system, the user specifies the desired singularity locations and types. The system converts each specification into a simple vector field and combines them into a global field using radial basis functions. The idea of using basis vector fields is inspired by the work of Wejchert and Haumann [1991]. However, the vector fields created in this manner often have more singularities than the user intended. The system does not provide a way of removing undesired singularities, and therefore lacks control over vector field topology. Rockwood and Bunderwala [2001] propose a technique that uses geometric algebra to create a vector field based on user-specified singularity locations and types (source, saddle, etc). The user can interactively create a vector field by adding desired singularities or

removing and editing the user-specified singularities. This system also lacks control over vector field topology, since unspecified singularities may appear. Theisel [2002] proposes a 2D vector field design system in which the user has complete control over vector field topology. To do so, the user needs to specify the *topological skeleton* of the desired vector field, and the system creates a field to match the skeleton. However, specifying the topological skeleton for a complicated vector field can be cumbersome. Both of the aforementioned topology-based design systems [Rockwood and Bunderwala 2001; Theisel 2002] require a planar parameterization, and it is not obvious how they should be generalized to 3D surfaces.

All of the preceding systems have certain traits that we wish to incorporate into our design system. In fact, we will borrow techniques from existing systems to serve our purposes at various stages. This will become clear in Sections 4, 5, and 6.

3.2 Vector Field Topology

In their pioneering work, Helman and Hesselink [1991] visualize a vector field by extracting and displaying its topological skeleton, which consists of the singularities and their connectivity. They propose an efficient method for extracting the topological skeleton for continuous vector fields that are defined in either a plane or volume. This work has inspired a great deal of interest in understanding and visualizing vector fields through topological analysis. There has been considerable work in the visualization community on vector field topology, and we only mention a few relevant publications here. Scheuermann et al. [1998] use Clifford algebra to study the nonlinear singularities in a vector field and propose an efficient algorithm for merging nearby linear singularities into a higher-order singularity. More recently, Polthier and Preuß have used Hodge decomposition to locate singularities of different types in a vector field [2003]. Wischgoll and Scheuermann [2001] propose an efficient algorithm for extracting the periodic orbits in a planar flow.

3.3 Vector Field Simplification

Vector field simplification has been well-researched by the scientific visualization community. Most of the datasets that come from scientific simulation are difficult to analyze due to noise in the data. Vector field simplification refers to reducing the complexity of a vector field while maintaining its major features. A simplification technique can be either topology-based (TB) or nontopology-based (NTB).

NTB methods are performed either globally or locally. Existing NTB techniques, such as those of Polthier and Preuß [2003], Westermann et al. [2000], and Tong et al. [2003], are often based on performing Laplacian smoothing on the potential of a vector field, which is a scalar field. For example, Tong et al. [2003] decompose a vector field into three components: curl-free, divergence-free, and harmonic. Each component is individually smoothed and the results are summed. Vector-based smoothing is performed only on the harmonic part, while potential-based smoothing applies to the divergence-free and curl-free components. Smoothing operations reduce the vector field complexity and are most likely to remove a large percentage of the singularities.

TB methods simplify vector field topology explicitly. According to the Poincaré-Hopf theorem, it is possible to eliminate a pair of singularities with opposite Poincaré indices at the same time. This idea has been formulated into an operation called singularity pair cancellation, which forms the foundation of many existing TB methods. A class of TB methods perform pair cancellation on scalar fields defined on surfaces [Edelsbrunner et al. 2002, 2003] by changing the values of the scalar function near the singularity pair. This is equivalent to simplifying the gradient vector field of the scalar function. Ni et al. [2004] allow the user to design fair Morse functions over a mesh surface, which is equivalent to designing gradient vector fields. In their work, the user specifies the desired number and configuration of the critical points of the function, and the system performs multigrid relaxation to determine a Morse

function that meets the requirements. Another class of TB methods perform cancellation on a vector field directly. For instance, Tricoche et al. [2001] first locate a region surrounding the singularity pair, and then perform a nonlinear optimization on the vector values at interior vertices so that the Poincaré indices are zero for every triangle inside the region. All of the aforementioned TB methods are based on Morse theory, for example, gradient vector fields.

Our system provides both an NTB method (Section 5.2) and a TB method (Section 5.3.1), and the implementations of our methods are rather different from existing techniques. For instance, our singularity pair cancellation algorithm is based on Conley index theory, which allows us to work with arbitrary vector fields. Furthermore, existing topological analysis and simplification techniques are limited to planar and volume domains because it is not clear how to represent a continuous vector field on a mesh surface. We describe a piecewise interpolation scheme in Section 6 that overcomes this problem, which allows vector field analysis and editing to be adapted to meshes.

4. DESIGN FOR PLANAR DOMAINS

Our planar vector field design system consists of three stages: *initialization*, *analysis*, and *editing*. During the initialization stage, the user quickly creates a vector field with a set of specifications. Vector field topology is not a concern at this point. Next, the system performs both geometric and topological analysis of the current vector field and provides visual feedback to the user. In the editing stage, the user modifies the vector field through a set of predefined editing operations. The user may perform many editing operations before accepting the result. The initialization and analysis stages are relatively straightforward, and we describe them in Sections 4.1 and 4.2, respectively. The editing stage is at the core of our design system, and we will describe this in Section 5.

4.1 Initialization

The first stage allows the user to easily create an initial vector field, without being concerned about its topology. There have been two ways of creating such a field: relaxation [Turk 2001; Wei and Levoy 2001], and using basis vector fields [Praun et al. 2000; van Wijk 2002]. We adopt van Wijk's basis vector approach [2002] because we are impressed by its intuitive nature and simplicity. In this approach, every user-specified constraint is employed to create a basis vector field that is defined in the plane. An initial vector field is then constructed as a weighted sum of these basis vector fields. We will refer to each user-specified constraint as a *design element*, which can be either *singular* or *regular*. A design element has a center location and a set of control parameters which will be described next.

A singular element corresponds to a vector field that has a singularity of certain type at a desired location. For instance, if the user desires an isotropic source at location $\mathbf{p}_0 = (x_0, y_0)$ with strength $k > 0$, the system will create the following vector field for any point $\mathbf{p} = (x, y)$ in the plane:

$$V(\mathbf{p}) = e^{-d\|\mathbf{p}-\mathbf{p}_0\|^2} \begin{pmatrix} k & 0 \\ 0 & k \end{pmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix}, \tag{2}$$

where d is a decay constant that is used to control the amount of influence of the basis vector field. Other isotropic singular elements include a sink, saddle, counter-clockwise center, and clockwise center, whose matrices are the following: $\begin{pmatrix} -k & 0 \\ 0 & -k \end{pmatrix}$, $\begin{pmatrix} k & 0 \\ 0 & -k \end{pmatrix}$, $\begin{pmatrix} 0 & -k \\ k & 0 \end{pmatrix}$, $\begin{pmatrix} 0 & k \\ -k & 0 \end{pmatrix}$, respectively. The system allows the user to modify the scale, orientation, and center location of an existing singular element, as well as to remove it altogether. Modifications to singular elements will result in more complicated matrices (details can be found in van Wijk [2002]).

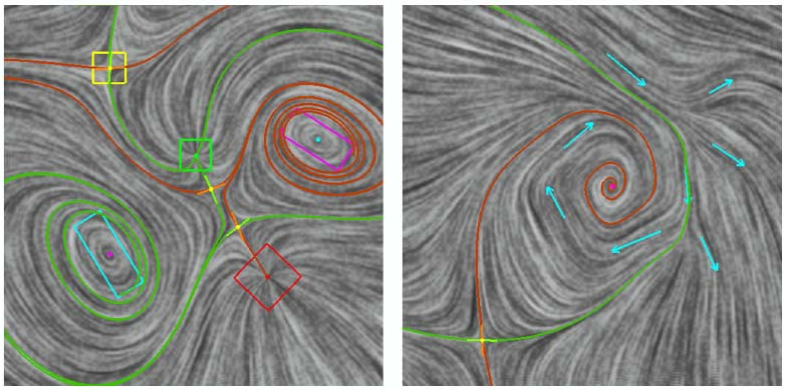

Fig. 5. An initial vector field can be created using singular elements (left, colored boxes), and regular elements (right, colored arrows). The centers of the colored boxes are the locations of the desired singularities. Notice there are unspecified singularities in both examples.

A regular element assigns a particular nonzero vector value V_0 at a desired location $\mathbf{p}_0$. Again, the system creates a basis vector field as follows:

$$V(\mathbf{p}) = e^{-d\|\mathbf{p}-\mathbf{p}_0\|^2} V_0 \tag{3}$$

The resulting vector field is interactively updated and displayed as the user continues to make adjustments to the sets of regular and singular elements. Figure 5 shows two vector fields that were generated using singular elements (left) and regular elements (right). In practice, both types of specifications can be combined to create an initial vector field. Notice that summing the basis vector fields may cause additional (perhaps unwanted) singularities to appear, that is, the ones that are not at the centers of any box in this figure. They can be handled through the topological editing operations that we will describe in Section 5.

4.2 Analysis

Our system performs the following analysis on a given vector field: computing curl and divergence, locating singularities and determining their types, and tracing separatrices.

The initial vector field created in the first stage is difficult to analyze because of its complicated formulae (Eqs. (2) and (3)). Furthermore, analytical formulae are not available for 3D surfaces that lack a global parameterization. To perform analysis in a fast and efficient manner and to be able to generalize the method to surfaces, we follow the approach by Helman and Hesselink [1991] and use a piecewise approximation in which the underlying domain is tessellated by a triangular mesh. The vector values are sampled at vertices according to the analytic formula and linearly interpolated on the edges and inside the triangles. To be more specific, for a given planar triangular mesh, our system represents a vector field V by assigning vector values $\{W_1, W_2, \ldots, W_n\}$ at the mesh vertices $\{v_1, v_2, \ldots, v_n\}$. For a point $\mathbf{p} = (x, y)$ inside a triangle $T = \{v_{T_1}, v_{T_2}, v_{T_3}\}$ whose barycentric coordinates are $(\alpha_1, \alpha_2, \alpha_3)$, we have

$$V(\mathbf{p}) = \sum_{j=1}^{3} \alpha_j W_{T_j}, \tag{4}$$

or under some local coordinate system of T, $V(\mathbf{p}) = \mathbf{M_T}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$, where $\mathbf{M_T} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is the Jacobian. This representation does not require an analytical formula and is compatible with many

graphics applications that use vector fields. Furthermore, Eq. (4) can be adapted to represent a continuous surface vector field (Section 6.2.1).

For each triangle, our system computes the following information: the divergence and curl, Poincaré index, the location of the singularity inside (if any), and the incoming and outgoing directions if the triangle contains a saddle. The details of computing these quantities using piecewise linear representation can be found in Tricoche [2002]. We also compute the topological skeleton of the vector field, which is done by following the approach of Helman and Hesselink [1991]. Starting from every saddle point, we follow the flow forward in its outgoing directions until the flow is stopped at a singularity or hits the boundary. To trace the trajectories away from a saddle, we use a Runge-Kutta algorithm with adaptive stepsize control [Cash and Karp 1990]. This gives us the two outgoing separatrices. Similarly, we obtain the two incoming separatrices by following the flow backward along the incoming directions of the saddle. Figures 3 and 5 show the topological skeletons of the corresponding vector fields.

5. EDITING

Vector field editing is at the heart of our design system. The set of useful editing operations is application-dependent. For instance, in texture synthesis and nonphotorealistic rendering, the user often needs to remove unwanted singularities or to move them to less visible regions. Fluid simulation may require adjusting the amount of curl and divergence of an external force. Furthermore, noisy datasets often contain a large number of singularities and rather complex behaviors. Simplifying the flow while maintaining its major features is a necessary task for any vector field design system. We provide the following operations.

(1) Matrix actions on flows: flow *rotations* and *reflections*;

(2) *flow smoothing* within a user-defined region; and

(3) topological editing operations of *singularity pair cancellation* and *singularity movement*.

Matrix actions can be used to adjust flow characteristics, such as curl and divergence. Flow smoothing is an efficient vector field simplification operation that can also simplify vector field topology.

Topological editing operations are used to provide explicit control over the number and location of singularities in the vector field. Most existing singularity pair cancellation algorithms assume that there is a connecting orbit between the singularity pair, as in the case of a source/saddle or sink/saddle cancellation. When the pair involves a center or a focus of a high curl, however, the connecting orbit either does not exist or cannot be computed in a numerically stable fashion. Consequently, these techniques do not address such cases. A similar issue comes up in singularity movement, where it is necessary to compute the trajectory that connects the singularity to its new desired location under the current flow. Such a trajectory does not always exist when the singularity is a saddle or center. As we will describe later in Sections 5.3.1 and 5.3.2, matrix actions can also be used to modify the types of singularities such that the aforementioned connecting orbits exist and can be computed easily in the modified field. This is essential to overcome the numerical instabilities associated with regions of high curls and regions near saddles.

Because we use a piecewise linear approximation, all editing operations affect the vector values at vertices only. These values are then extended to a continuous vector field defined on the whole mesh surface through a piecewise interpolation scheme.

5.1 Matrix Actions on Flows

Any 2×2 matrix $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ induces a vector field operator as follows: $(M(V))(\mathbf{p}) = \begin{pmatrix} a & b \\ c & d \end{pmatrix} V(\mathbf{p})$. When M has a full rank, it does not change the number or location of singularities in the vector field. Furthermore,

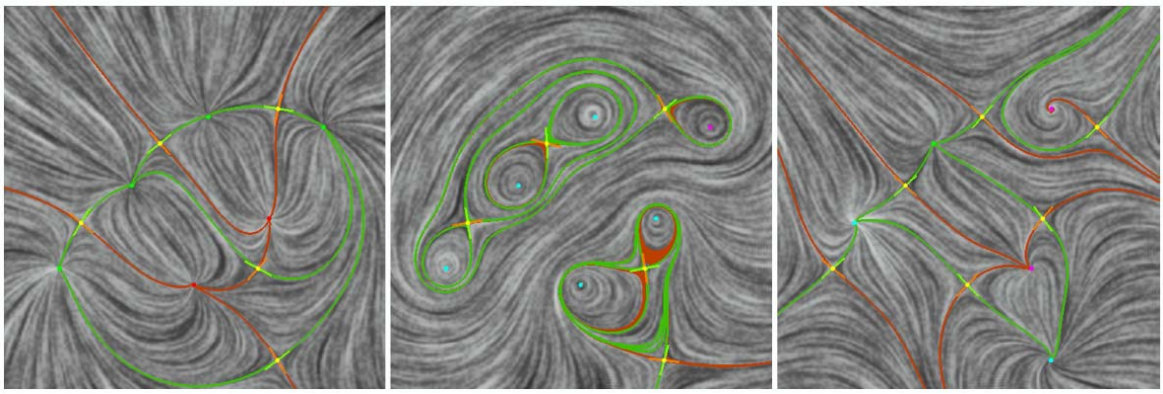

Fig. 6. In this figure, a vector field (left) is first rotated by $\frac{\pi}{2}$ (middle), then reflected with respect to the X-axis (right).

M maintains the Poincaré index if $\det(M) > 0$, and negates it if $det(M) < 0$. For any $\theta \in \mathbb{R}$, $R_\theta = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$ is a *flow rotation operator* and $F_\theta = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ -\sin(\theta) & -\cos(\theta) \end{pmatrix}$ is a *flow reflection operator*. The actions of R_θs and F_θs are of particular interest to us.

For any θ and vector field V, it is straightforward to verify that

$$(curl(R_\theta(V)))^2 + (div(R_\theta(V)))^2 = (curl(V))^2 + (div(V))^2. \tag{5}$$

This implies that for any point $\mathbf{p}$ in the domain, there are appropriate rotations of V such that $curl(R_\theta(V))(\mathbf{p}) = 0$ or $div(R_\theta(V))(\mathbf{p}) = 0$. Furthermore, a curl-free vector field can be rotated into a divergence-free vector field, and vice versa, with a $\frac{\pi}{2}$ rotation. Topologically speaking, flow rotations do not alter the number, location, or Poincaré index of the singularities ($\det(R_\theta) = 1 > 0$). Any singularity with a Poincaré index of $+1$ can be converted into a source with an appropriate rotation. A saddle remains a saddle under flow rotations. However, its incoming and outgoing directions are rotated, possibly by different amounts. These topological properties make flow rotations essential for singularity pair cancellation (Section 5.3.1) and singularity movement operations (Section 5.3.2), especially in regions of high curls.

F_θ induces a reflection on the values of V with respect to the axis $\sin(\frac{\theta}{2})X + \cos(\frac{\theta}{2})Y = 0$. It is straightforward to verify that $F_\theta^2 = Id$. For planar domains, flow reflections do not alter the number or location of the singularities in V. Since $\det(F_\theta) = -1 < 0$, they negate the sign of the Poincaré indices. Just as flow rotations can convert a singularity with a Poincaré index of $+1$ into a source, they can turn any saddle into a source with an appropriate choice of reflection axis. This makes flow reflections crucial for our singularity movement operations on saddles (Section 5.3.2).

Figure 6 shows the effect of applying flow rotations and reflections to a planar vector field. The actions are $R_0 = Id$ (left), $R_{\frac{\pi}{2}}$ (middle), and $F_{\frac{\pi}{2}}$ (right). Notice that in all instances, the number and location of singularities do not change. Flow rotations maintain the Poincaré indices, while flow reflections negate their signs.

The concepts of flow rotations and reflections are not new. Theisel and Weinkauf [2002] define four types of operations for feature matching between vector fields. These operations include rotations and negative scalings (including reflections). However, we believe that it is a novel idea to use flow rotations and reflections to overcome the numerical difficulties associated with regions of high curl and regions near saddles.

5.2 Flow Smoothing

We now describe our implementation of flow smoothing, which is a nontopology-based vector field simplification operation. This operation is carried out in two stages. First, the user specifies a simply connected region by drawing a loop in the domain. Next, the flow inside the region is replaced by a "simpler" flow. The key is to let the user decide the region for smoothing. Once the region is determined, a number of known smoothing techniques, such as those of Westermann et al. [2000], Polthier and Preuß [2003], and Tong et al. [2003], can be used to replace the flow inside. In particular, Tong et al. [2003] compute a Hodge-Helmholtz decomposition of the original vector field. Smoothing is performed on the potentials of curl-free and divergence-free parts. However, smoothing the harmonic component still requires vector-valued smoothing. We choose to perform smoothing on the vector values directly, which is faster because it avoids the costs of performing decomposition and the two additional potential-based smoothings. Furthermore, vector-valued smoothing tends to remove high-frequency noise from the data, as well as reducing the number of singularities in the vector field. This is supported by our numerical tests. For remeshing purposes, Alliez et al. [2003] employ a similar component-based approach to smooth curvature tensor fields.

Given a vector field V and user-specified region R, we replace V with another vector field $\overline{V}$ inside R. This is achieved by solving the vector-valued Laplace equation inside R, with V being fixed on ∂R. Let $\overline{V}(\mathbf{p}) = \begin{pmatrix} \overline{F}(\mathbf{p}) \\ \overline{G}(\mathbf{p}) \end{pmatrix}$. The values of $\overline{V}$ inside R are given by

$$\begin{pmatrix} \nabla^2 \overline{F} = 0 \\ \nabla^2 \overline{G} = 0 \end{pmatrix}. \tag{6}$$

In practice, the user-specified region R is part of the underlying mesh that is used to represent the planar domain. To solve Eq. (6) on this discrete mesh, we fix the values at the boundary vertices of R, that is, $\overline{F} = F, \quad \overline{G} = G$. The values of $\overline{F}$ and $\overline{G}$ for an interior vertex v_i are determined by

$$\begin{pmatrix} \overline{F}(v_i) \\ \overline{G}(v_i) \end{pmatrix} = \sum_{j \in J} \omega_{ij} \begin{pmatrix} \overline{F}(v_j) \\ \overline{G}(v_j) \end{pmatrix}. \tag{7}$$

Here, 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 [2003], since this method guarantees ω_{ij} to be nonnegative. This leads to a pair of sparse linear systems which we solve through an implicit biconjugate solver.

In Figure 7, a complicated vector field with many singularities (left) is converted into a vector field with only one singularity (right) through smoothing. The boundary of the user-specified region is highlighted with a white loop. Notice that the vector field is not altered outside the user-specified region. Later, we make use of flow smoothing to perform singularity pair cancellation (Section 5.3.1) and singularity movement (Section 5.3.2).

5.3 Topological Editing Operations

Our system provides two topological editing operations: singularity pair cancellation, and singularity movement. The former refers to removing a pair of (unwanted) singularities with opposite Poincaré indices, while the latter is used to move a singularity to a more desirable location. Both operations provide topological guarantees in that they only affect the intended singularities, and our implementation is based on Conley index theory.

5.3.1 *Singularity Pair Cancellation.* As discussed in Section 2.1, singularity elimination must be performed for a pair of singularities with opposite Poincaré indices. This operation is therefore called

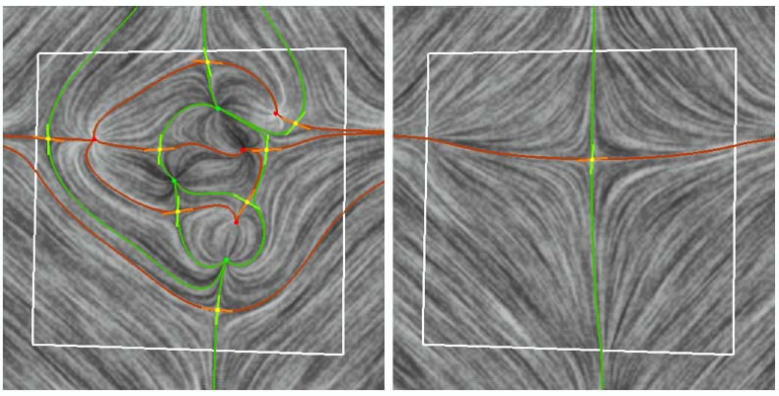

Fig. 7. Flow smoothing inside a user-specified region (bounded by the white loop). Notice the vector field outside the region is unchanged.

singularity pair cancellation. There have been several pair cancellation methods for simplifying scalar fields on surfaces [Edelsbrunner et al. 2002, 2003]. These techniques achieve singularity pair cancellation for the gradient field by modifying the scalar values in a region near the singularity pair. It is not clear how these techniques can be used for generic vector fields, which need not correspond to any scalar functions.

Tricoche et al. [2001] propose a pair cancellation technique for vector fields by allowing a saddle to be cancelled with either a source or a sink. To achieve this, they first find a narrow neighborhood that encloses the singularity pair and their connecting orbit. Then, an iterative nonlinear optimization is performed on the vector values at interior vertices of this region so that the Poincaré index for every triangle is zero. There are a number of issues with this approach. From a theoretical viewpoint, any simplification technique based on the Poincaré index cannot be applied to the cancellation of a repeller/attractor pair in which one of the entities is a periodic orbit (Section 2.1). From a numerical point of view, this technique is not robust in handling pair cancellations that involve a center or focus with high curl. Furthermore, the nonlinear optimization technique is computationally expensive and does not guarantee that a solution can be found.

In this work, we propose a new pair cancellation technique based on Conley index theory which provides theoretical guarantees for any attractor/repeller pair, including objects other than singularities. We will only consider the case of a source/saddle pair cancellation. If the singularity with a positive Poincaré index is not a source, we can always find an appropriate rotation to turn it into a source, while the saddle does not change its type. Our algorithm consists of two stages. First, the system determines an isolating block R with trivial Conley index such that R encloses the singularity pair in its interior. Second, the flow inside R is replaced with a new, singularity-free vector field. Figure 8 provides an illustration of the idea. Later, we use a similar two-stage approach for moving a singularity (Section 5.3.2).

Let $\mathbf{s}_+$ and $\mathbf{s}_-$ be the source and saddle, respectively. When there is a unique connecting separatrix between them, we can construct an isolating block R containing $\mathbf{s}_+$ and $\mathbf{s}_-$, on which the cancellation can be performed [Mischaikow and Mrozek 2002]. We need the following definition:

Definition 5.1. For a given vector field V, let φ denote the flow induced by V. For a region Q in the domain, we define its images under the forward and reverse flow as

$$\Omega(Q) = \varphi(Q, [0, \infty)) \quad \Omega^{-1}(Q) = \varphi(Q, (-\infty, 0]). \tag{8}$$

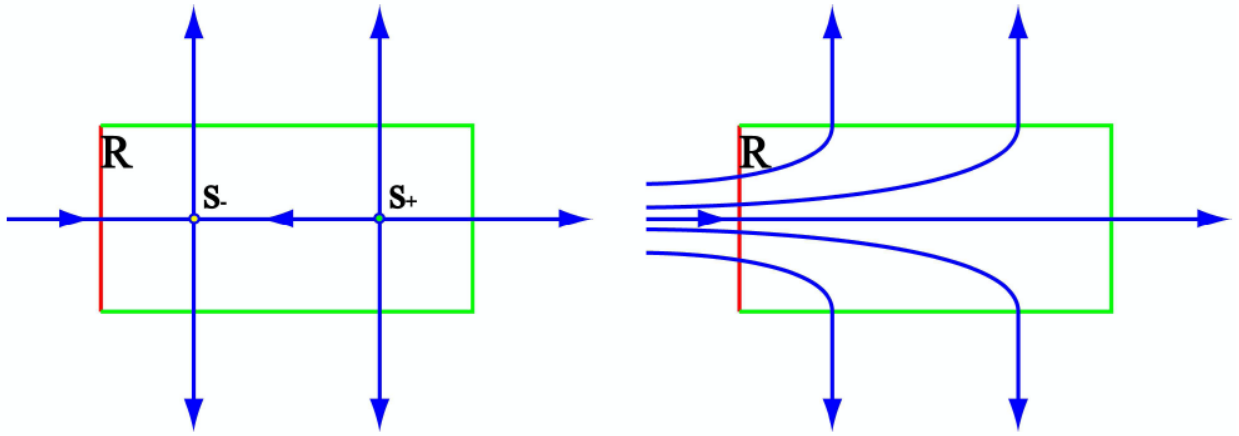

Fig. 8. An illustration of our two-step algorithm for singularity pair cancellation between a source $\mathbf{s}_+$ and saddle $\mathbf{s}_-$. In the left, an isolating block R is found to enclose both singularities, and its boundary consists of two segments: inflow (red) and outflow (green). The vector field inside R is replaced with a flow that has no singularities (right).

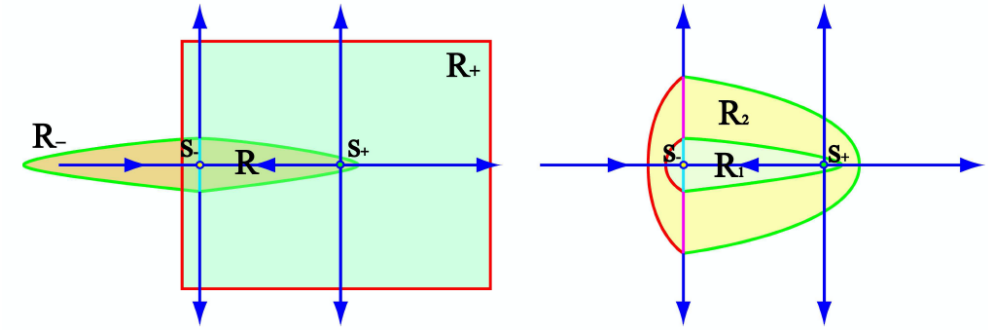

Fig. 9. Our construction of an isolating block R for singularity pair cancellation. In the left, a region R_+ is generated by following the flow forward from a neighborhood of $\mathbf{s}_+$. Similarly, a region R_- is obtained by following the reverse flow from a neighborhood of $\mathbf{s}_-$. When there is a unique connecting orbit between $\mathbf{s}_+$ and $\mathbf{s}_-$, $R = R_+ \cap R_-$ is an isolating block with trivial Conley index. In the right, two valid regions R_1 and R_2 are obtained by using different sizes of neighborhoods of $\mathbf{s}_-$. R_2 is preferred, since it is larger and tends to result in smoother flows after the cancellation.

To find the isolating block R, we begin with the isolating neighborhoods M and N of $\mathbf{s}_+$ and $\mathbf{s}_-$, respectively. In general, $R = \Omega(M) \bigcap \Omega^{-1}(N)$ is an isolating neighborhood. If there exists a unique separatrix going from $\mathbf{s}_+$ to $\mathbf{s}_-$, then the Conley index of R is trivial and it is possible to replace the vector field inside R with one that is singularity free [Mischaikow and Mrozek 2002] (Figure 9, left).

Next, we describe a practical algorithm for computing R over a domain represented by a triangular mesh. Let M and N be sets of triangles that enclose $\mathbf{s}_+$ and $\mathbf{s}_-$, respectively. Here, $\Omega(M)$ is obtained by performing region growing from M and following the flow forward. Similarly, $\Omega^{-1}(N)$ is obtained by performing region growing from N and following the flow backward. We need the following definition:

Definition 5.2. Given a vector field V and polygonal region R, a boundary edge e is an *exit* for the forward flow with respect to V if $\max_{\mathbf{p}\in e}(N_\mathbf{p} \cdot V_\mathbf{p}) > 0$. Here, $N_\mathbf{p}$ is the outward normal to the region at point $\mathbf{p}$. Similarly, e is an exit for the backward flow with respect to V if $\min_{\mathbf{p}\in e}(N_\mathbf{p} \cdot V_\mathbf{p}) < 0$.

If V is a piecewise linear vector field on a boundary edge $e = (v_1, v_2)$ with a constant outward normal N_e, the tests for exit edges reduce to determining the signs at the vertices. Here, e is an exit edge for the forward flow with respect to V if $\max(V(v_1) \cdot N_e, V(v_2) \cdot N_e) > 0$. Similarly, e is an exit edge for the backward flow with respect to V if $\min(V(v_1) \cdot N_e, V(v_2) \cdot N_e) < 0$.

Let M be the triangle that contains $\mathbf{s}_+$. Starting from M, we construct $\Omega(M)$ by adding one triangle at a time and keeping track of the behavior of the flow on boundary edges of $\Omega(M)$. A new triangle can be added only by crossing an exit edge. The region growing process continues until there are no more exit edges, that is, the flow enters $\Omega(M)$ everywhere on its boundary.

$\Omega^{-1}(N)$ is constructed in a similar fashion by starting from N and following the flow backward. However, the choice of N is a delicate issue, and affects the shape of $\Omega^{-1}(N)$ and subsequently, R. Due to the limited resolution of the underlying mesh, R needs to be as large as possible, so long as its Conley index remains trivial. We perform a linear search on the length of the outgoing separatrices of $\mathbf{s}_-$ such that the covering triangles form N. Figure 9 shows the effect of following these separatrices to varying lengths. Here, $R^+ = \Omega(M)$ and $R^- = \Omega^{-1}(N)$.

To replace the flow inside R, we use flow smoothing. As described earlier, this operation tends to simplify the vector field topology, and our numerical results indicate that flow smoothing is efficient for singularity pair cancellation, as long as the region R has a reasonable shape. The following pseudocode illustrates our algorithm for cancelling a source/saddle pair, where the source $\mathbf{s}_+$ has a zero curl.

(1) **PairCancellation**(V, $\mathbf{s}_+$, $\mathbf{s}_-$)
(2) Let M be the triangle containing $\mathbf{s}_+$, and we use region growing to find $\Omega(M)$.
(3) Let $\gamma = 1$, $d\gamma = 0.5$, and $V_{working} = V$.
(4) Let S_1 and S_2 be the two outgoing separatrices at $\mathbf{s}_-$.
(5) **while** $\gamma > 0$ and $d\gamma > \delta$ (δ is a user-specified constant)
(6) Let $S_{1,\gamma} \subset S_1$ be the portion starting from $\mathbf{s}_-$ such that $length(S_{1,\gamma}) = \gamma length(S_1)$. Define $S_{2,\gamma}$ similarly.
(7) Let N be the minimal set of triangles that contain $S_{1,\gamma}$ and $S_{2,\gamma}$, and we compute $\Omega^{-1}(N)$ using region growing.
(8) $R = \Omega(M) \cap \Omega^{-1}(N)$.
(9) **if** R does not satisfy the necessary Conley conditions,
(10) $\gamma = \gamma - d\gamma$, $d\gamma = d\gamma/2$.
(11) **else**
(12) perform smoothing in R.
(13) **if** the resulting flow contains any singularity inside R,
(14) undo smoothing.
(15) $\gamma = \gamma - d\gamma$, $d\gamma = d\gamma/2$.
(16) **else**
(17) update $V_{working}$ with the smoothed flow.
(18) $\gamma = \gamma + d\gamma$, $d\gamma = d\gamma/2$.
(19) **end if**
(20) **end if**
(21) **end while**

In Line (9), in order to meet Conley conditions, R must be simply connected, contain no singularities other than $\mathbf{s}_+$ and $\mathbf{s}_-$, and have a trivial Conley index. The purpose of the binary search on γ is to determine an optimal length along the outgoing separatrices such that the region R has a reasonable shape. When $\gamma = 0$, R is a narrow region that covers both the singularity pair and their connecting orbit, and has a trivial Conley index. When $\gamma \to 1$, R tends to have a nice shape. However, it may cover other singularities, such as the sinks that are linked to $\mathbf{s}_-$ through the outgoing separatrices, and the

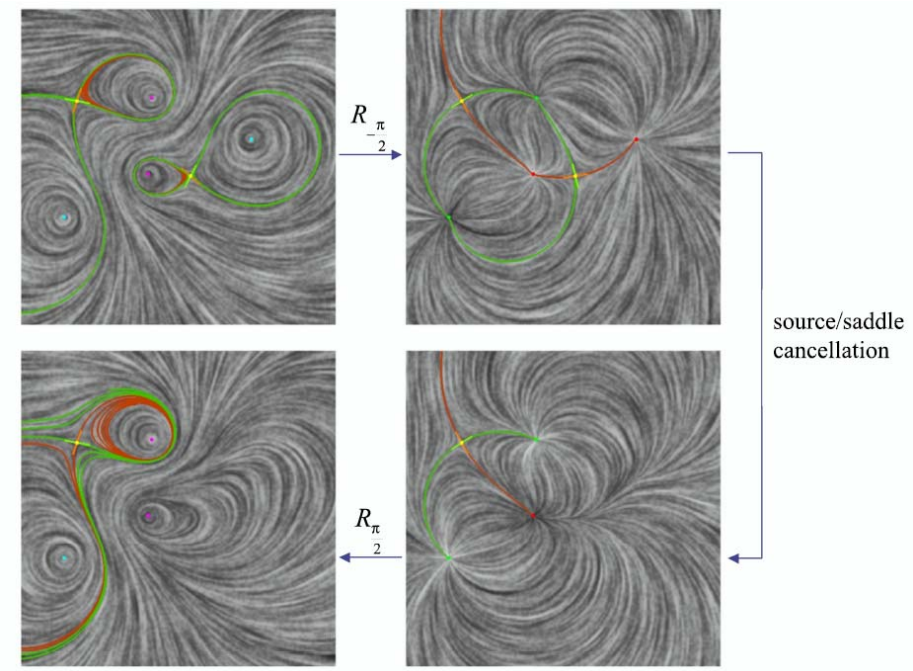

Fig. 10. We use flow rotations to help overcome the numerical difficulties associated with high curls in a vector field. Here, a center/saddle cancellation is performed on a vector field (upper-left) to obtain a new one (lower-left). The vector field is first rotated by $\frac{\pi}{2}$ (upper-right), followed by a pair cancellation (lower-right), before a compensating rotation is performed (lower-left).

Conley index of R is no longer trivial. The binary search process balances between the two factors, and tends to converge very quickly, since region growing and smoothing are very fast. In addition, when $d\gamma$ is small enough, γ and $\gamma + d\gamma$ often correspond to the same set of triangles N, and the computation can simply be avoided.

Flow rotations are crucial for the success of pair cancellation operations. If the original vector field has a high curl around $\mathbf{s}_+$, as in the case of a divergence-free flow, the connecting orbit between singularities may not even exist. Figure 10 demonstrates how our system cancels a center and saddle pair (upper-left). The flow is first rotated by $\frac{\pi}{2}$ to become a curl-free vector field (upper-right) in which the center has changed to a source, and there is now a connecting orbit between the source and saddle. Next, the source and saddle are cancelled. Finally, a compensating rotation of $-\frac{\pi}{2}$ is performed (lower-left).

5.3.2 *Singularity Movement.* Moving a singularity to a new location provides the user with control over the position of the singularities in a vector field. To our knowledge, this is the first time such an operation has been proposed and an algorithm presented. Through flow reflections and rotations, the problem of moving a singularity is reduced to moving a source. Similar to singularity pair cancellation, our algorithm for moving a source is based on Conley index theory and is carried out in two stages. First, we compute an isolating block R such that it encloses the connecting orbit for the current location $\mathbf{s_{old}}$ and desired new location $\mathbf{s_{new}}$ under the current vector field V. By construction, R has the Conley index of a source and does not contain any other singularities, either in its interior or on its boundary (Figure 4(b)). Next, the vector field inside R is modified to contain only one singularity at $\mathbf{s_{new}}$ (Figure 11).

Let $R = \Omega(M) \bigcap \Omega^{-1}(N)$, where M is a small neighborhood of $\mathbf{s_{old}}$ and N is a neighborhood of $\mathbf{s_{new}}$. To ensure $\mathbf{s_{new}}$ is in the interior of R, another point $\mathbf{s}'$ is located such that it is on the forward trajectory

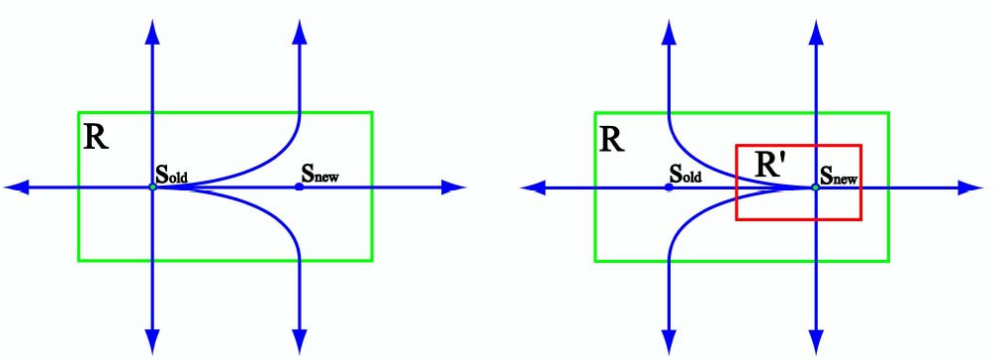

Fig. 11. Moving a source from $\mathbf{s_{old}}$ to $\mathbf{s_{new}}$. An isolating block R is found to enclose both $\mathbf{s_{old}}$ and $\mathbf{s_{new}}$ such that R has the Conley index of a source. Then a small region R' is found to enclose $\mathbf{s_{new}}$, and appropriate vector values are assigned to $\partial R'$ such that it forces a source at $\mathbf{s_{new}}$. For region $R \setminus R'$, flow smoothing produces a new vector field without any singularity.

from $\mathbf{s_{new}}$ under V. Let us consider the trajectory J of $\mathbf{s}'$ under the flow $R_{\frac{\pi}{2}}(V)$. Here, J serves the same purpose as the outgoing separatrices of the saddle in pair cancellation. Let N be the largest segment on J that makes R an isolating block with the Conley index of a source. This ensures that R is a wide region.

Let T be the triangle that contains $\mathbf{s_{new}}$. Our system assigns vector values at the three vertices of T to enforce a source at $\mathbf{s_{new}}$. Let $R' = \{T\}$. Then region $L = R \setminus R'$ has two boundaries. The flow enters L from the inner boundary and leaves at the outer boundary. Therefore, L has the trivial Conley index (see Figure 4(e)), and the flow smoothing inside L usually produces a vector field without singularities.

Moving a center or saddle is considerably more challenging. First, finding a connecting orbit between the saddle and a regular point is numerically unstable. Moreover, finding a connecting orbit between a center and a regular point is almost impossible. To solve these numerical issues, we make use of flow rotations and reflections to make singularity movement applicable to any linear singularity. If $\mathbf{s_{old}}$ is a saddle, we use flow reflection to turn it into a singularity with positive index. Then, we perform an appropriate rotation such that the resulting vector field has a zero curl at $\mathbf{s_{old}}$, and the singularity is now a source. This simplifies the process of locating the connecting orbit between $\mathbf{s_{old}}$ and $\mathbf{s_{new}}$. Figure 12 provides an example of moving a saddle in a vector field (upper-left). First, flow reflection is applied to turn the saddle into a source (upper-right). Next, the source is moved (lower-right) before a compensating reflection is applied (lower-left).

6. DESIGN FOR 3D MESH SURFACES

In this section, we describe how we adapt our three-stage vector field design system for planar domains to mesh surfaces. There are several challenges that we must overcome. First, a 3D surface often lacks a global parameterization, which is needed to correlate the tangent vectors defined at different locations in order to build surface-based basis fields from design elements. Second, topological analysis of vector fields requires a scheme that interpolates the vectors that are defined at the vertices and produces a continuous vector field everywhere in the mesh. However, the tangent planes of a mesh surface are discontinuous at vertices and edges, and the definition of vector field continuity from smooth manifolds does not apply. In addition, as we will demonstrate in Section 6.2.1, the piecewise linear interpolation scheme that works well for planar vector fields will cause inconsistent vector values across the edges. To address these issues, we borrow the ideas of geodesic polar maps and parallel transport from classical differential geometry to set up correlations between tangent vectors defined at different parts of the surface. The correlations are used for two purposes. First, we extend the construction of basis vector

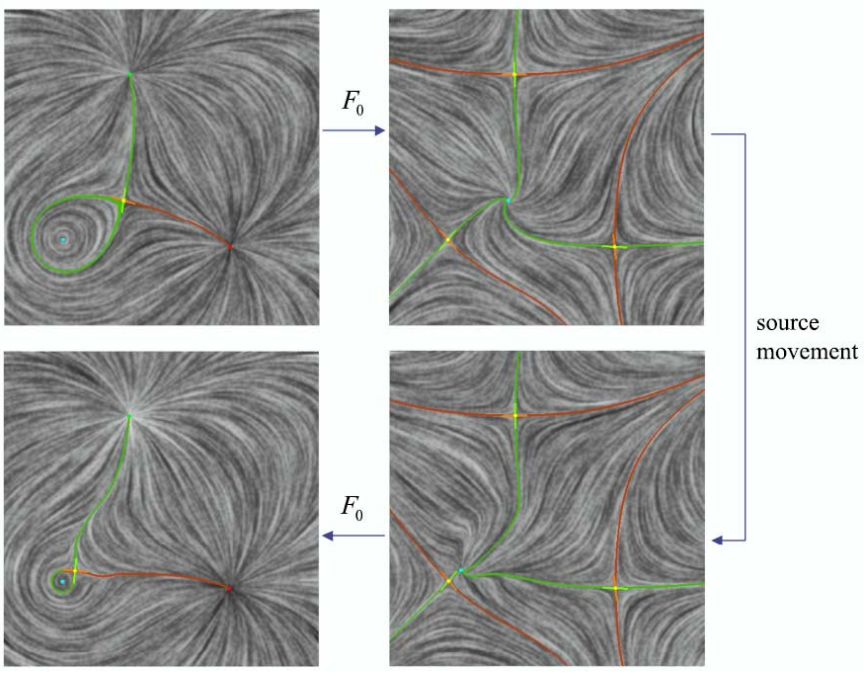

Fig. 12. Flow reflections are useful to overcome the numerical difficulties associated with saddles. In this example, a singularity movement operation is applied to a vector field (upper-left) to obtain a new vector field (lower-left). The vector field is first reflected so that the saddle becomes a source (upper-right), followed by a source movement (lower-right), before a compensating reflection is performed (lower-left).

fields (Section 4.1) to surfaces by parallel transporting tangent vectors from the location of the design element to anywhere on the surface. Second, we adapt the piecewise linear approximation from planar domains (Section 4.2) to mesh surfaces by parallel transporting vectors from a vertex to anywhere inside its one-ring neighborhood. The piecewise interpolation scheme results in a continuous surface vector field, and supports efficient analysis and editing operations.

6.1 Initialization: Construction of Basis Vector Fields on Surfaces

In this section, we describe how we construct a basis vector field on a mesh from a design element. Recall that in the planar case, a design element O is converted into a global basis field, according to Eqs. (2) and (3). To extend this method to surfaces, we perform the following three-step process, as illustrated in Figure 13. First, we compute a geodesic polar map with respect to the location of O (left), which assigns every point A on the surface with a pair of coordinates (x_A, y_A). This can seen as building a global parameterization for the surface using the tangent plane at O. Next, (x_A, y_A) are substituted into Eqs. (2) or (3) to obtain a tangent vector value W_A defined at O (middle). Finally, W_A is parallel transported from O to A along the shortest geodesic that connects them (right). The process is based on several ideas from classical differential geometry, namely, geodesics, geodesic polar maps, and parallel transport. We will review each of these in turn.

A geodesic on a curved surface is a locally shortest and straightest curve. It is a generalization of a straight line in the plane. Starting from a point $\mathbf{p}$ on the surface, there is a geodesic in every tangent direction $\vec{v}$. Denote this geodesic by $\gamma_{\mathbf{p},\vec{v}}$. A point $\mathbf{q}$ on $\gamma_{\mathbf{p},\vec{v}}$ with a distance ρ from $\mathbf{p}$ can be identified by the coordinates (ρ, θ), where θ is the angular coordinate of $\vec{v}$ with respect to some local frame at $\mathbf{p}$.

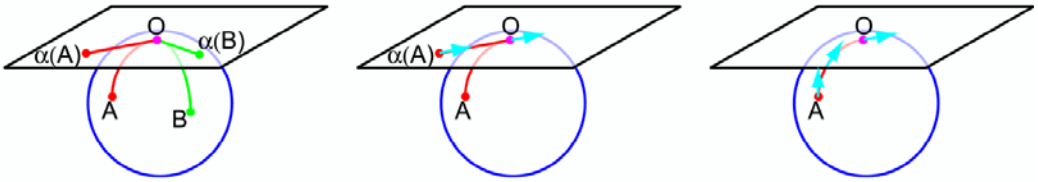

Fig. 13. Our three-step algorithm (from left-to-right) for creating a surface basis vector field from a user-specified constraint O. First, the surface is parameterized using a geodesic polar map with respect to O. This parameterization is denoted as α. Second, the basis vector field is computed inside the tangent plane at O with the polar coordinates from α. Finally, the vectors are parallel transported along shortest geodesics on the surface.

In the plane, the coordinates reduce to the familiar polar coordinates. This map is the geodesic polar map. On a curved surface, a geodesic polar map is neither bijective nor continuous. For example, on the Earth, a geodesic polar map with respect to the North Pole will have discontinuities at the South Pole. However, the focus of a design element is in a nearby region, and the geodesic polar map with respect to the design element meets our needs.

In differential geometry, parallel transport is used to correlate tangent vectors that are defined at different locations with a geodesic that connects them. Formally, let $\mathbf{p}$ and $\mathbf{q}$ be two points on a smooth manifold S, and let $\gamma : [0, 1] \rightarrow S$ be a geodesic such that $\gamma(0) = \mathbf{p}$ and $\gamma(1) = \mathbf{q}$. Furthermore, let $V_\mathbf{p}$ and $V_\mathbf{q}$ be tangent vectors defined at $\mathbf{p}$ and $\mathbf{q}$, respectively. If the oriented angle between $\gamma'(0)$ and $V_\mathbf{p}$ equals that between $\gamma'(1)$ and $V_\mathbf{q}$, then $V_\mathbf{p}$ and $V_\mathbf{q}$ are said to be parallel with respect to γ , and $V_\mathbf{q}$ is said to be the parallel transport of $V_\mathbf{p}$ along γ. Notice that γ gives rise to an orthonormal and bijective linear map between $TM_\mathbf{p}$ and $TM_\mathbf{q}$, the tangent planes at $\mathbf{p}$ and $\mathbf{q}$. This map is a transport function and denoted by $f_\mathbf{pq}$.

For a design element d, let $\alpha_d : S \rightarrow \mathbb{R}^2$ be a geodesic polar map with respect to d, and let $f_{d\mathbf{p}} : TM_d \rightarrow TM_\mathbf{p}$ be the transport function along a geodesic $\gamma_{d\mathbf{p}}$. Then, the surface basis vector field $W_d(\mathbf{p})$ that corresponds to a design element d is constructed as $W_d(\mathbf{p}) = f_{d\mathbf{p}} V(\alpha_d(\mathbf{p}))$. In this equation, V is evaluated according to formulae such as Eqs. (2) or (3). For the purposes of building the geodesic polar map α_d and computing the transport function $f_{d\mathbf{p}}$, we need to compute a geodesic from any vertex of the surface to the design element d.

Assume that the design element d is situated inside a triangle T. We first compute the geodesic distance function g_d with respect to d for every vertex using the fast marching method [Kimmel and Sethian 1998]. The values of g_d at a vertex is the radial component of the geodesic polar map. To construct the angular component in the ideal situation, we need to perform particle tracing from a vertex in the opposite direction of $\bigtriangledown g_d$, the gradient vector field of g_d. However, performing particle tracing for every vertex is expensive. In addition, $-\bigtriangledown g_d$ often has local minima other than d. To overcome these problems, we propose a two-region approach in which the angular component θ is computed directly *only* within a surface disk surrounding d such that the disk contains a user-specified percentage of the total vertices in the mesh. We use 25% in practice. For a point inside the disk, we project it onto the tangent plane at d to obtain θ. For a vertex $\mathbf{p}$ outside the disk, we perform particle tracing from $\mathbf{p}$ in the direction of $-\bigtriangledown g_d$ until it hits an edge $e = \mathbf{rs}$ that is on the boundary of the disk. If $\theta(\mathbf{r})$ and $\theta(\mathbf{s})$ are both known, then $\theta(\mathbf{p})$ is obtained by linearly interpolating between $\theta(\mathbf{r})$ and $\theta(\mathbf{s})$. If particle tracing from $\mathbf{p}$ fails to reach any boundary edge, then there is no shortest geodesic between $\mathbf{p}$ and d. In this case, a random θ value is assigned to $\mathbf{p}$. Although this may seem to have created discontinuities in the vector field, let us recall that vector values are only computed at the vertices at this stage. In the next section, we will describe a piecewise interpolation scheme in which a continuous vector field is created based on the values defined at the vertices.

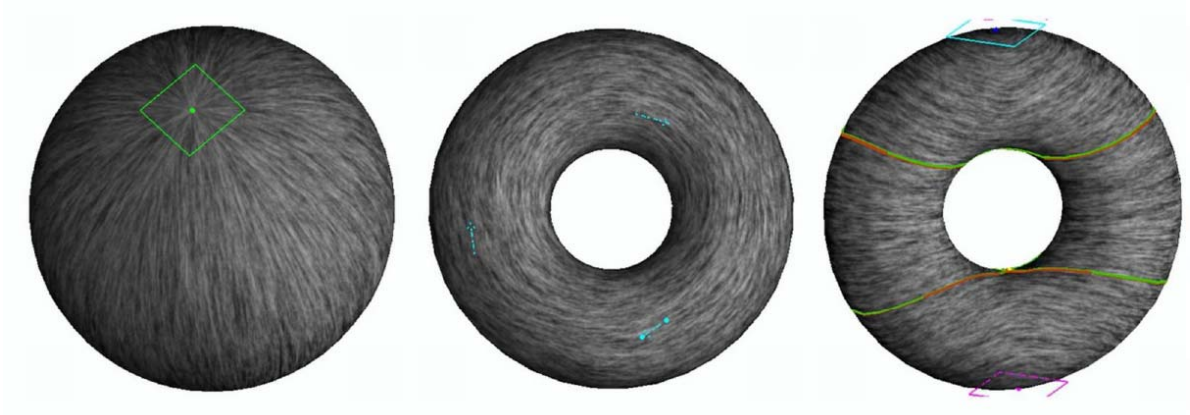

Fig. 14. Design elements for creating an initial vector field. From left to right: a dipole vector field on a sphere using a source element, a singularity-free vector field on a torus with three regular elements, and another vector field on the torus with a clockwise center element and counter-clockwise element. Notice that the surface basis vector fields are very efficient for creating initial vector fields.

The aforementioned method works well for nearly flat or spherical regions. However, around nearly cylindrical features, the projection is likely to result in undesired values on the side of the cylinder that is opposite to the center of the design element. We are investigating other possible surface parameterizations, such as cylindrical coordinates, to address this.

Given a geodesic polar map, a tangent vector can be parallel transported to a vertex along a geodesic. This completes the construction of a basis vector field. Figure 14 provides three examples: a dipole vector field on a sphere with a single source element (left), a singularity-free vector field on a torus with three regular elements (middle), and another vector field on a torus with a clockwise center element and a counter-clockwise center element (right). Additional examples are shown in Figure 2.

Let us stress that this is not the only way to create basis vector fields. In van Wijk's visualization tool [2003], an element is translated into a 3D vector field before being projected onto the surface. While our approach appears to be more intuitive in this case, given that a surface is locally homeomorphic to a plane, van Wijk's 3D projection method is faster, since it does not require the construction of geodesic polar maps. Constrained optimization [Turk 2001; Wei and Levoy 2001] is another way to produce an initial vector field with desired behaviors. Praun et al. [2000] propose a vector field propagation approach in which a vector value is defined inside one face of the mesh surface. Through region growing, the vector value for a new triangle is obtained by computing the average tangent vector of neighboring triangles that are already part of the region. This vector is then projected onto the face.

6.2 Analysis: Vector Field Continuity and Piecewise Interpolation Scheme for Meshes

Once vector values are determined at the vertices, we use a piecewise interpolation scheme to construct a continuous vector field on the mesh surface. This is needed in order to provide control over vector field topology. Unfortunately, the piecewise linear approximation for planar domains does not produce consistent vector fields on mesh surfaces. Figure 15 illustrates this with an example. Given a vertex O and its one-ring neighborhood (a), the vectors are zero at A, B, C, and D. In addition, $V(O)$ is in the direction of $\overrightarrow{OC}$. With piecewise linear representation, the vectors at the midpoints of edges OD and OB are fixed (c). Due to continuity constraints across the edges, the vectors at the middle points OA in triangles $\triangle ODA$ and $\triangle OAB$ lead to inconsistencies (d). The problem is due to the angle deficit at O and discontinuity of tangent planes at the vertices and across the edges. In this section, we describe an interpolation scheme that is guaranteed to produce a continuous vector field

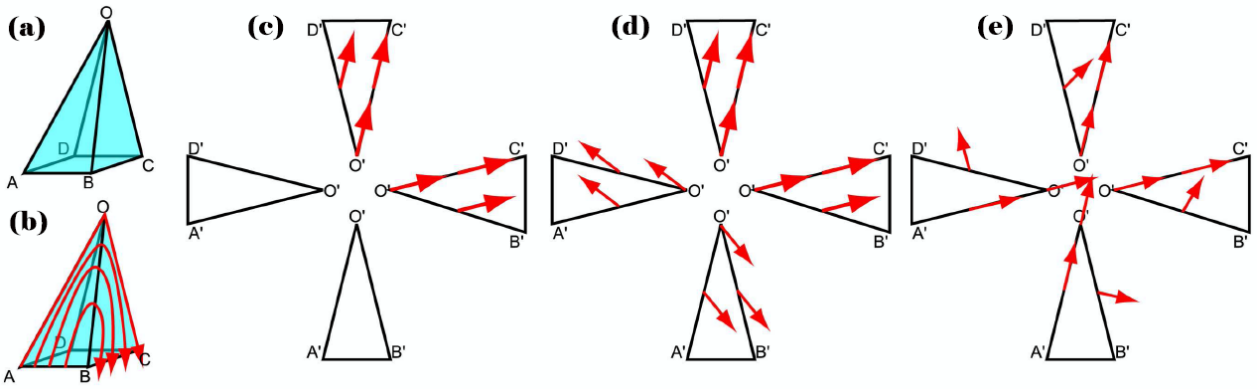

Fig. 15. The piecewise linear representation for planar domains does not produce continuous vector fields on mesh surfaces. The vectors are zero at A, B, C, and D. The vector at O is in the direction of $\overrightarrow{OC}$ (a). The piecewise linear representation and vector field consistency along edges OD and OB eventually lead to vector field discontinuity along edges OA (c) and (d). In contrast, our piecewise interpolation scheme (Section 6.2) produces a continuous vector field (e), which corresponds to a family of nonintersecting and spacing-filling trajectories in the one-ring neighborhood of O (b).

directly on mesh surfaces. This scheme is a generalization of the piecewise linear representation from the planar case, and it allows fast and efficient analysis and editing of the vector fields on meshes. Before describing the scheme, however, we first need a definition of the vector field continuity for mesh surfaces.

For planar domains, the concept of vector field continuity is well-defined because any two vectors can be compared, regardless of their locations. This is no longer true for a general surface, since the tangent planes at different locations are distinct and there is not an obvious and consistent way to correlate them without a global parameterization. Furthermore, the tangent planes of mesh surfaces are often discontinuous across the vertices and edges. Stam [2003] addresses the problem by using a subdivision surface whose tangent planes are continuous everywhere. However, for most geometric processing operations, subdivision surfaces incur higher computational costs than polygonal meshes.

Recall that for a smooth vector field, a point is either a singularity or regular point. We can always define singularities for surface vector fields because zero vectors can be identified, regardless of location. In addition, the *flow-box theorem* for ordinary differential equations gives us a picture of what happens near a regular point [Hale and Kocak 1991].

THEOREM 6.1. *Let V be a smooth vector field defined in $D \subset \mathbb{R}^n$. If $\mathbf{p}_0 \in D$ is a regular point of V, then there exists a neighborhood U of $\mathbf{p}_0$ and a homeomorphism $h : U \to \mathbb{R}^n$ which carries each piece of a trajectory lying on U onto a straight line of $\mathbb{R}^n$ parallel to the X-axis.*

In other words, near a regular point, it is possible to warp the space such that nearby trajectories are parallel and space-filling. We propose to use this property as the definition for *vector field consistency* (continuity) for a regular point on mesh surfaces. Notice that the flow-box theorem is true, even when the vector field V is only *Lipschitz-continuous* [Calcaterra and Boldt 2003], which is a stronger condition than continuity, but weaker than smoothness. Basically, a vector field V over a domain is Lipschitz-continuous if there exists a constant K such that for any $\mathbf{x}, \mathbf{y}$ in the domain, $|V(\mathbf{x}) - V(\mathbf{y})| < K|\mathbf{x} - \mathbf{y}|$. We propose the following definition:

Definition 6.2. Let V be a vector field defined on a mesh surface $\mathbf{M}$. Here, V is *consistent* at a point $\mathbf{p}_0 \in \mathbf{M}$ if one of the following situations is true:

(a) For any *path* γ $[0, 1) \rightarrow \mathbf{M}$ such that $V(\gamma(t))$ is well-defined for any $t \in [0, 1)$ and for $\lim_{t\rightarrow 1} \gamma(t) = \mathbf{p}_0$, we have $\lim_{t\rightarrow 1} V(\gamma(t)) = 0$. In this case, $\mathbf{p}_0$ is a singularity.

(b) There exists a neighborhood U of $\mathbf{p}_0$ and a homeomorphism $h : U \rightarrow \mathbb{R}^2$ which carries each piece of a trajectory lying in U onto a straight line in $\mathbb{R}^2$ parallel to the X-axis. In this case, $\mathbf{p}_0$ is *regular*.

In other words, a consistent vector field on a mesh surface should exhibit the same local behaviors as those defined in a plane. Notice that in this definition, we require continuity at singularities, and unique solvability at regular points. Unique solvability refers to the fact that for any point $\mathbf{p}_0$, there exists a unique solution to the differential equation induced by the vector field. Observe that as curves on a continuous surface, it makes sense to discuss the continuity of the trajectories of a vector field. Figure 15(e) illustrates the result of our interpolation scheme, to be described next (compare this to Figure 15(d)). Notice that this scheme leads to a family of nonintersecting and space-filling trajectories in the one-ring neighborhood of O (Figure 15(b)).

6.2.1 *Piecewise Approximation.* For every vertex in the mesh, we record its surface normal and the local frame for the tangent plane. This allows a tangent vector to be transformed from its local coordinates to global coordinates.

A vector field V on a mesh surface is represented by assigning tangent vectors $\{W_1, W_2, \ldots, W_n\}$ at the mesh vertices $\{v_1, v_2, \ldots, v_n\}$. For each W_i, we maintain its local coordinates for vector field design and 3D global coordinates for display. We cannot simply perform interpolation of W_is, since they are in general not coplanar. Furthermore, without a surface parameterization, tangent vectors that are defined at different vertices are not correlated. To overcome these problems, we first define a local parameterization for the one-ring neighborhood of a vertex v_i. This parameterization allows the parallel transport of W_i to any point $\mathbf{p}$ inside v_i's one-ring neighborhood. Let μ_i be such a transport function (which we will soon describe). Then, for a point $\mathbf{p}$ inside a triangle $T = \{v_{T_1}, v_{T_2}, v_{T_3}\}$ whose barycentric coordinates are $(\alpha_1, \alpha_2, \alpha_3)$, Eq. (4) can now be rewritten as the weighted sum of the tangent vectors that are parallel transported from the three vertices:

$$V(\mathbf{p}) = \sum_{j=1}^{3} \alpha_j \mu_{T_j}(W_{T_j}, \mathbf{p}) \tag{9}$$

Let us consider V_i, the vector field that is constructed according to Eq. (9) under the assumption that $W_j = 0$ for every $j \neq i$. We have $V = \sum_{i=1}^{n} V_i$. Notice V_i is zero outside the one-ring neighborhood of vertex v_i. As we will soon see, V_i is a consistent vector field over the mesh surface for every i, and so is V. Before we describe the parameterization and transport function in detail, we need the following definitions [Polthier and Schmies 1998]:

Definition 6.3. Let M be a polyhedral mesh representing a closed curved surface. Let v be a vertex with incident triangles T_j $(j = 1, .. n)$, and θ_j be the interior angle of T_j at v. Then,

(1) The total vertex angle at $\theta(v)$ is given by $\theta(v) = \sum_{j=1}^{n} \theta_j$; and
(2) $2\pi - \theta(v)$ is the Gaussian curvature at v. A vertex v is *Euclidean* if it has a zero Gaussian curvature; otherwise, it is *non-Euclidean*.

Define $r = \frac{\theta(v)}{2\pi}$. Notice that $r = 1$ for vertices with a zero Gaussian curvature. There are two ways of measuring the angles between two rays emanating from a vertex v. The first is the Euclidean angle, which is the angle measured on the mesh. The second angle is the normalized angle, as measured in the tangent space (each ray corresponds to a tangent vector). The normalized angle α is related to the Euclidean angle β by $\beta = r\alpha$ [Polthier and Schmies 1998]. Figure 16 provides an illustration. In the left

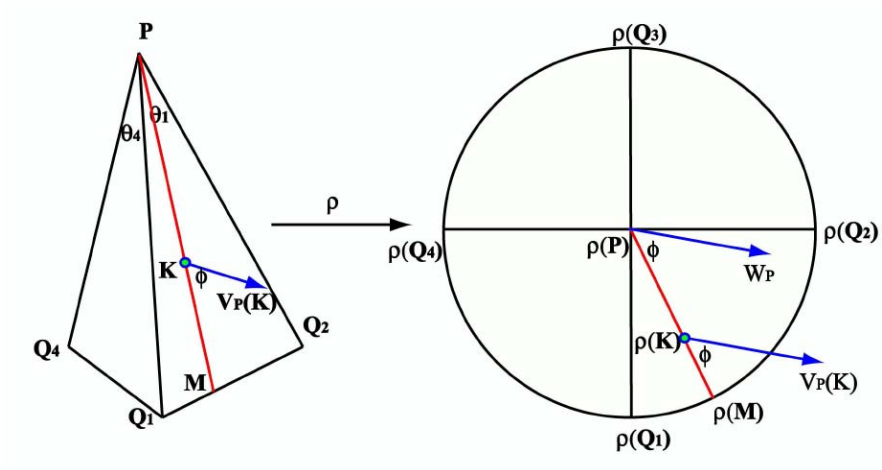

Fig. 16. This figure illustrates the idea of parallel transporting a tangent vector $W_i = W_{\mathbf{P}}$ from a vertex $v_i = \mathbf{P}$ to a point $\mathbf{K}$ inside $\mathbf{P}$'s one-ring neighborhood, R. First, we build a local parameterization ρ for R. Then, $W_{\mathbf{P}}$ is parallel transported to $\mathbf{K}$ along the ray $\overrightarrow{\rho(\mathbf{P})\rho(\mathbf{K})}$. This construction guarantees vector field consistency on mesh surfaces.

portion, $\mathbf{P} = v_i$ is a vertex with the tangent plane $TM_{\mathbf{P}}$ (right). Its one-ring neighborhood R consists of the triangles $\triangle\mathbf{P}\mathbf{Q}_1\mathbf{Q}_2$, $\triangle\mathbf{P}\mathbf{Q}_2\mathbf{Q}_3, \ldots,$ and $\triangle\mathbf{P}\mathbf{Q}_n\mathbf{Q}_1$ $(n = 4)$. Let $\theta_j = \angle\mathbf{Q}_j\mathbf{P}\mathbf{Q}_{j+1}$. In the right portion, let D be the unit disk in $TM_{\mathbf{P}}$ and let ρ be the following homeomorphism from R to D:

(1) ρ induces a bijective mapping between the boundary of R and the boundary of the unit circle. For any point $\mathbf{M} \in \partial R$, ρ is a linear map from $\overrightarrow{\mathbf{PM}}$ to $\overrightarrow{\rho(\mathbf{P})\rho(\mathbf{M})}$.

(2) ρ is linear scaling on the angles between rays. If two rays emanating from $\mathbf{P}$ have a Euclidean angle of θ, then the angle between their images (normalized angle) is $\frac{\theta}{r}$.

Note that this construction is similar to the geodesic polar maps used by Welch and Witkin [1994] for free-form shape design, with a minor difference: In their setting, the parameterization domain is a polygon, not the unit disk, as in our case. Polthier and Schmies [1998] have used similar maps to perform parallel translation on vectors over a polygonal surface.

To transfer W_i to a point $\mathbf{K}$ inside triangle $\mathbf{Q}_j\mathbf{P}\mathbf{Q}_{j+1}$, we first locate the ray $\overrightarrow{\mathbf{PM}}$ that contains $\mathbf{K}$. Let ϕ be the counter-clockwise angle between W_i and the ray $\overrightarrow{\rho(\mathbf{P})\rho(\mathbf{M})}$. We define $\mu_i(W_i, \mathbf{p})$ as the vector at $\mathbf{K}$ such that the angle between $\mu_i(W_i, \mathbf{p})$ and $\overrightarrow{\mathbf{PM}}$ equals ϕ. Furthermore, $|\mu_i(W_i, \mathbf{p})| = |W_i|$.

Basically, we have created a constant vector field $(= W_i)$ inside the unit disk (Figure 16, right), which is then scaled such that the magnitude is one at the origin and linearly decreases to zero along each line segment connecting the origin and a point on the boundary of the disk. Notice the resulting vector field is Lipschitz-continuous. Finally, the scaled vector field is mapped to the one-ring neighborhood to obtain V_i through parallel transport, as described earlier. Note that the distortion contained in the parameterization ρ is caused by the Gaussian curvature of v_i, and therefore has an upper bound. This implies that V_i is also Lipschitz-continuous. As a result, vector field consistency (continuity and unique solvability) is ensured. Intuitively, as a parameterization, ρ does not distinguish between the points inside a triangle or on an edge. Consequently, vector field continuity is automatically guaranteed here. Furthermore, the continuity of ρ ensures the continuity of the resulting vector field at the vertices. For planar domains, $r = 1$ everywhere and this approximation reduces to the piecewise linear representation (Section 4.2).

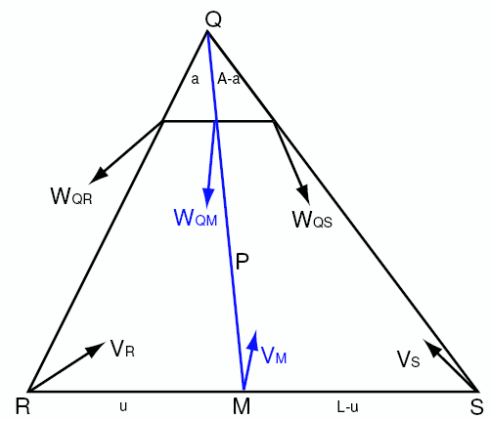

Fig. 17. Singularity determination inside a triangle under the piecewise interpolation scheme. Here, Q is the only non-Euclidean vertex.

Our piecewise interpolation scheme induces a vector field W that is continuous and nonlinear inside each triangle T minus three vertices. We can think of such a region as a triangle minus three arbitrarily small corners, each around a vertex. Therefore, W can be seen as being defined over a hexagon that is arbitrarily close to T. Along each edge of the triangle, W is linear in terms of length. Along the sides where the corners are cut, W is linear in terms of vertex angle. Locating singularities of W is rather difficult in this setting. Furthermore, the Poincaré index for a hexagon can be ± 2, which implies possibly two linear singularities or one second-order singularity inside T. This makes topological control more difficult. To overcome these difficulties, we perform a four-fold triangle subdivision for the input mesh. Basically, the midpoint of every edge in the original mesh becomes a new Euclidean vertex, since its total vertex angle is 2π. This means that every triangle in the subdivided mesh can have, at most, one non-Euclidean vertex, and the analysis becomes more tractable. From now on, we will assume the input mesh already satisfies this requirement.

6.2.2 *Analysis.* Our interpolation scheme results in a nonlinear vector field inside a triangle, which requires new ways of computing the Jacobian, curl, divergence, as well as the singularities and separatrices. In this section, we provide solutions to these issues. Let $T = \triangle QRS$ be a triangle with exactly one non-Euclidean vertex Q. Then the vector field V, as constructed in Eq. (9), is linear on RS and along any ray emanating from Q. Furthermore, W is a continuous vector field defined on T minus an arbitrarily small corner near Q, that is, a quadrilateral, as illustrated in Figure 17. Let $V_R = V(R)$ and $V_S = V(S)$ be the values at R and S, respectively. At Q, we need a direction to determine the value. Let W denote the vector field of V along an arbitrarily small line segment near Q such that $W_{QR} = W(QR)$ and $W_{QS} = W(QS)$ are the vectors in the direction $\overrightarrow{QR}$ and $\overrightarrow{QS}$.

Since the Jacobian is no longer constant inside T, we compute the pointwise Jacobian through a local approximation. First, two points M_1 and M_2 are selected inside T such that they are sufficiently close to M_0, and $\overrightarrow{M_0M_1}$ and $\overrightarrow{M_0M_2}$ are not colinear. Next, we build a linear vector field $\overline{V}$ such that $\overline{V}(M_i) = V(M_i)$ for $i = 0, 1, 2$. Finally, the Jacobian of V at M_0 is approximated by the Jacobian of $\overline{V}$.

Pointwise curl and divergence are computed from the Jacobian. On the other hand, the curl and divergence for a triangle can be obtained accurately by calculating the divergence and curl along the three edges of the triangle. Notice that the piecewise interpolation scheme that we describe in this section is linear along these edges. Let N_e and D_e be the outward normal and directional vector on an

edge e, then we have the following results:

$$\begin{aligned} div(T) = & \frac{(W_{QR} + V_R) \cdot N_{\overrightarrow{QR}}}{2} |\overrightarrow{QR}| \\ & + \frac{(V_R + V_S) \cdot N_{\overrightarrow{RS}}}{2} |\overrightarrow{RS}| \\ & + \frac{(V_S + W_{SQ}) \cdot N_{\overrightarrow{SQ}}}{2} |\overrightarrow{SQ}| \end{aligned} \tag{10}$$

$$\begin{aligned} curl(T) = & \frac{(W_{QR} + V_R) \cdot D_{\overrightarrow{QR}}}{2} |\overrightarrow{QR}| \\ & + \frac{(V_R + V_S) \cdot D_{\overrightarrow{RS}}}{2} |\overrightarrow{RS}| \\ & + \frac{(V_S + W_{SQ}) \cdot D_{\overrightarrow{SQ}}}{2} |\overrightarrow{SQ}| \end{aligned} \tag{11}$$

Note that the total curl and divergence is zero for any closed two-manifold. By construction, our piecewise interpolation scheme maintains this property for mesh surfaces.

The Poincaré index of a triangle is computed for a quadrilateral, as illustrated in Figure 17. Along each side, the vector field continuously and monotonically interpolates the vector values at the end points. We treat W_{QR} and W_{QS} as vector values defined at the ends of an arbitrarily small edge. The total index angle is in $(-4\pi, 4\pi)$, which implies that T can have, at most, one linear singularity. The piecewise interpolation scheme maintains the Poincaré-Hopf theorem, and our numerical results support this. If the Poincaré index of T is not zero, there must be a singularity inside. To locate the singularity P, we perform a binary search for a point $M \in \overrightarrow{RS}$ such that $V_M = V(M)$ and $W_{QM} = W(QM)$ are colinear and point in opposite directions. Let $W_{QM} = -\alpha V_M$, then $P = (1 - m)Q + mM$ $(m = \alpha/(\alpha + 1))$ is the location of the singularity. The Jacobian at P is used to determine its type, and in the case of a saddle, the incoming and outgoing directions.

The Runge-Kutta method that we used for computing the separatrices for planar vector fields (Section 4.2) can be adapted to mesh surfaces. Polthier and Schmies have also suggested a fourth-order Runge-Kutta method for computing trajectories for a continuous vector field on mesh surfaces [1998]. Figures 18 and 2 show some example vector fields on various 3D models along with their topological skeletons.

6.3 Editing

While the main concepts for editing operations on a surface remain the same as those for planar domains, some changes need to be made to reflect the differences in vector field representations and the complexity of the surface geometry, such as curvature and higher genus.

To perform flow rotations on a mesh surface, we simply rotate the vector value W_is in the tangent planes at each vertex. Since the transport functions are orthonormal transformations between the tangent planes (Section 6.2.1), rotating W_is by an angle of θ results in a rotation of the same angle inside every triangle and edge. Therefore, flow rotations maintain the number, location, and Poincaré index of the singularities. Furthermore, for any point inside a triangle, Eq. (5) remains valid. In contrast, a flow reflection requires that the local frames and reflection axes at every vertex be correlated. We make use of a global polar map to parallel transport this information. A flow reflection negates the sign of the Poincaré indices. However, it may create some additional singularities due to singularities in the

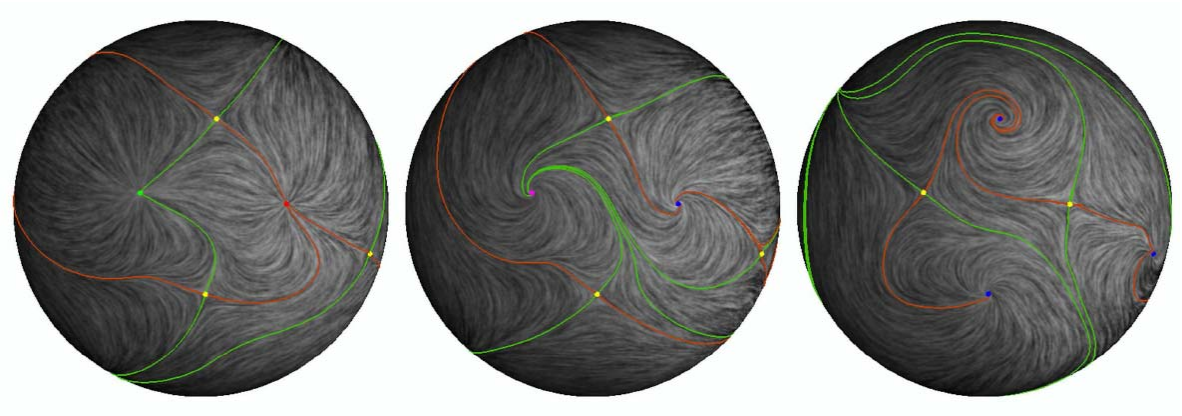

Fig. 18. A vector field on the sphere (left) is first rotated by $\frac{\pi}{4}$ (middle), then reflected (right).

fields of local frames and the field of reflection axes. Regardless of these issues, the resulting vector field still satisfies the Poincaré-Hopf theorem. Figure 18 illustrates the effects of flow rotations and reflections on a vector field defined over a sphere. The original field (left) is first rotated by $\frac{\pi}{4}$ (middle), then reflected (right). Compare this figure with Figure 6.

Similar to the planar case, flow smoothing on a surface vector field is carried out by performing vector-valued smoothing inside a user-specified region. We have implemented two variations for this. In the first, we perform vector-valued smoothing to the original vector field as a 3D vector field and project the resultant vector field onto the surface. The second approach parameterizes R based on some planar domain and performs smoothing in this domain. Both smoothing techniques provide similar results. However, theoretically speaking, the second approach seems more natural for surfaces.

7. APPLICATIONS

All the vector fields shown in this article were created using our system. In addition, we have applied vector field design to several graphics applications: painterly rendering of images, pencil sketch illustration of smooth surfaces, and example-based texture synthesis.

7.1 Painterly Rendering

Painterly rendering refers to creating digital images that have the appearance of being painted. There are numerous published approaches to painterly rendering, and to review them all is beyond the scope of this article. These techniques have focused on providing the user with control over certain aspects of brush strokes, such as textures and styles, while automatically determining other aspects, such as base colors and orientations. In particular, image-based gradient fields have often been used to guide the orientation of brush strokes. While this may be appropriate for some parts of the image (near the feature lines), it often produces brush strokes with noisy orientations in areas with nearly uniform colors. Furthermore, it creates unnecessary constraints on the way that artists may express themselves. Our goal for this application is to let the user control brush stroke orientations through vector field design.

We use a level-of-detail approach by Hertzmann [1998]. In this approach, a painting is created in a series of layers, starting with a rough sketch drawn with brush strokes of a large size. Then the sketch is painted over with brush strokes of gradually decreasing size at places where signals of higher frequencies are present. This approach is very fast and high-quality. However, we make the following modification: Instead of using the image gradient field to guide the brush stroke orientations, let the

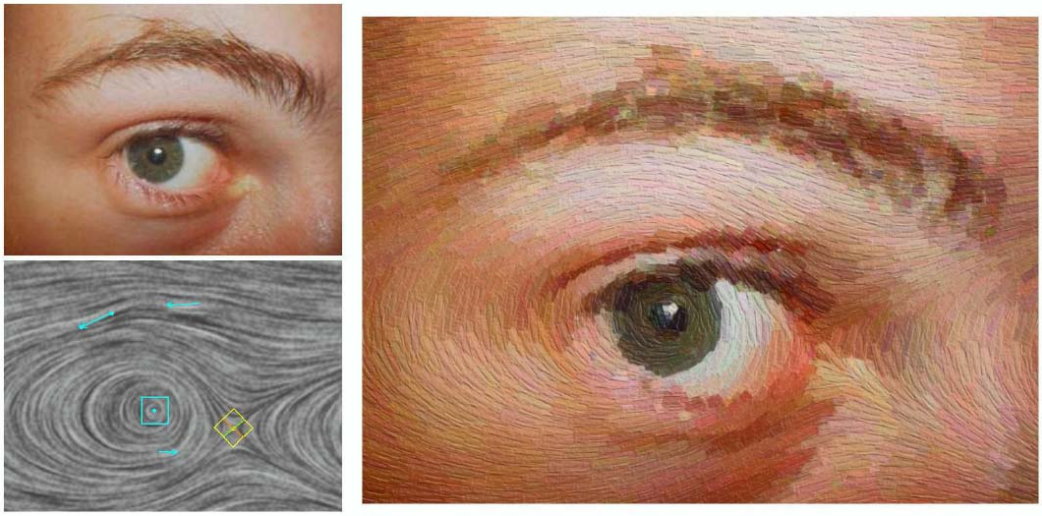

Fig. 19. Painterly rendering of a human eye image through vector field design. The input vector field was created using our system (lower-left). The high-quality van Gogh-style rendering (right) was produced offline with the approach of Hays and Essa [2004].

user create a synthetic vector field with our design system. To make this task fast and effective, we incorporate the painterly rendering algorithm into our system. In addition to viewing the vector field, the user can also switch to the painterly rendering that uses the current vector field. The results are interactively displayed as the user makes changes to the vector field. Figures 19 and 20 show the results for two source images: a human's eye (van Gogh-style) and a cat's face (impressionism). For the human eye, a center element was placed at the middle of the pupil and a saddle element was placed at the corner of the eye. Two regular elements were placed along the eyebrow to ensure that the brush strokes did not cross the feature. With five elements, a vector field (lower-left) was produced that matched the main features of the image (the eye and eyebrow). For the cat's face, two center elements of opposite orientations were placed at the middle of each eye. A saddle element was placed underneath the nose and six regular elements were placed along the ears and chin. The final high-quality painterly images in both figures were created offline using the algorithm of Hays and Essa [2004].

7.2 Nonphotorealistic Illustration of Surfaces

There has been much work in creating hatch-based illustrations of surfaces, and to review all of them is beyond the scope of this article. Girshick et al. [2000] have shown that principle curvature fields are good at conveying shapes. Traditional techniques often make use of principle curvature directions to guide the hatch field. Hertzmann and Zorin [2000] present an efficient algorithm for approximating the principle curvature fields over the mesh by local surface fitting, which leads to a high-quality pen-and-ink style of rendering of 3D shapes. Praun et al. [2001] treat the problem of hatch-based illustration as performing texture synthesis on surfaces, which leads to a real-time hatching system in which the user has the option to guide the orientation of hatches with a vector field on a 3D model. The researcher van Wijk [2003] applies his image-based flow visualization technique to curvature fields to produce nonphotorealistic illustrations of 3D surfaces. Similar to the image gradient field, principle curvature fields are rather noisy for regions where the principle directions are not prominent.

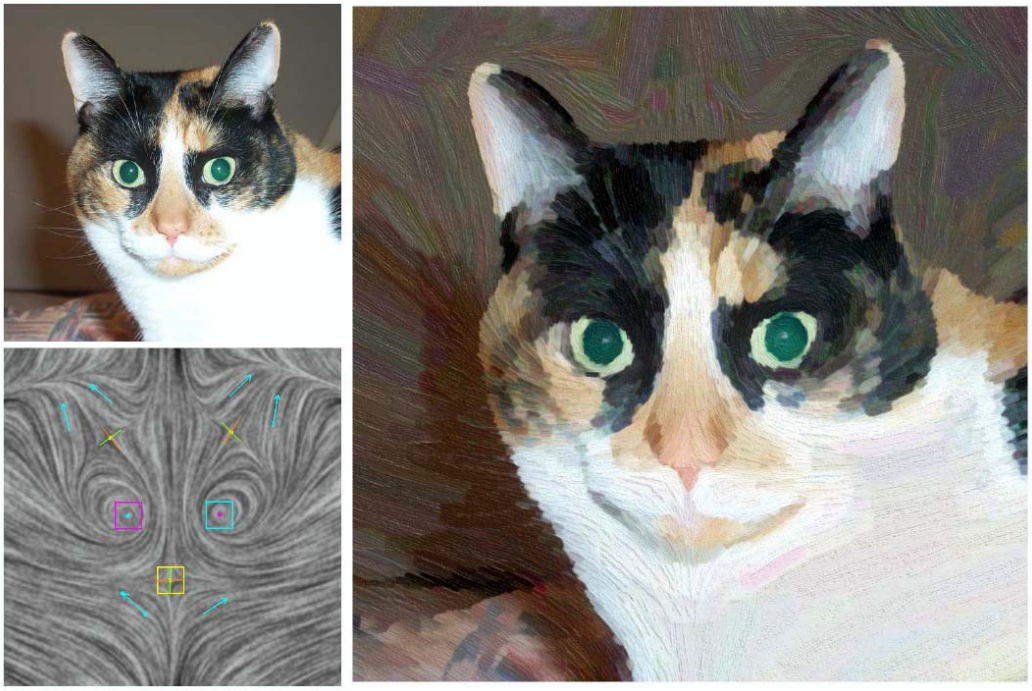

Fig. 20. Another example of painterly rendering based on vector field design: a cat's face. The high-quality impressionistic image shown in the right was also produced offline with the approach of Hays and Essa [2004].

In this work, we allow the user to guide the hatch field through vector field design. Figure 21 shows the results of applying this technique to various 3D models. The vector field used for the dragon model in the lower-right image was obtained by rotating the vector field from the lower-left image by $\frac{\pi}{3}$.

7.3 Example-Based Texture Synthesis

Example-based texture synthesis refers to creating patterns on surfaces based on a given input image of a texture. Praun et al. [2000] propose "lapped textures" in which the surface is partitioned into overlapping regions and each region receives a portion of the input image. This method is fast, but causes seams due to surface partition. For textures that contain only high frequencies, the seams are relatively unnoticeable. Another class of methods [Turk 2001; Wei and Levoy 2001] performs synthesis on surfaces directly, without creating seams. For any point on the surface, its color is copied from the pixel in the same image that provides the best neighborhood match, based on a distance criterion. For both types of synthesis methods, a vector field is used to provide local orientation and scale, as well as to determine the synthesis order.

Figure 22 shows the results of applying our vector field design system to texture synthesis on the bunny and the feline. The texture synthesis method is based on those of Turk [2001] and Wei and Levoy [2001]. The two vector fields used for the bunny are a sink element at the tail and a source element on the forehead (upper-left), and a $\frac{\pi}{3}$ rotation of a dipole (a source element and sink element) on the visible side of bunny (upper-right). Notice that the spiraling in the second vector field near the singularities on the side of the bunny is evident in the texture in the upper-right image. The bottom row of Figure 22 shows the feline with a tiger stripe pattern that is guided by two different vector fields, both of which lead to reasonable results.

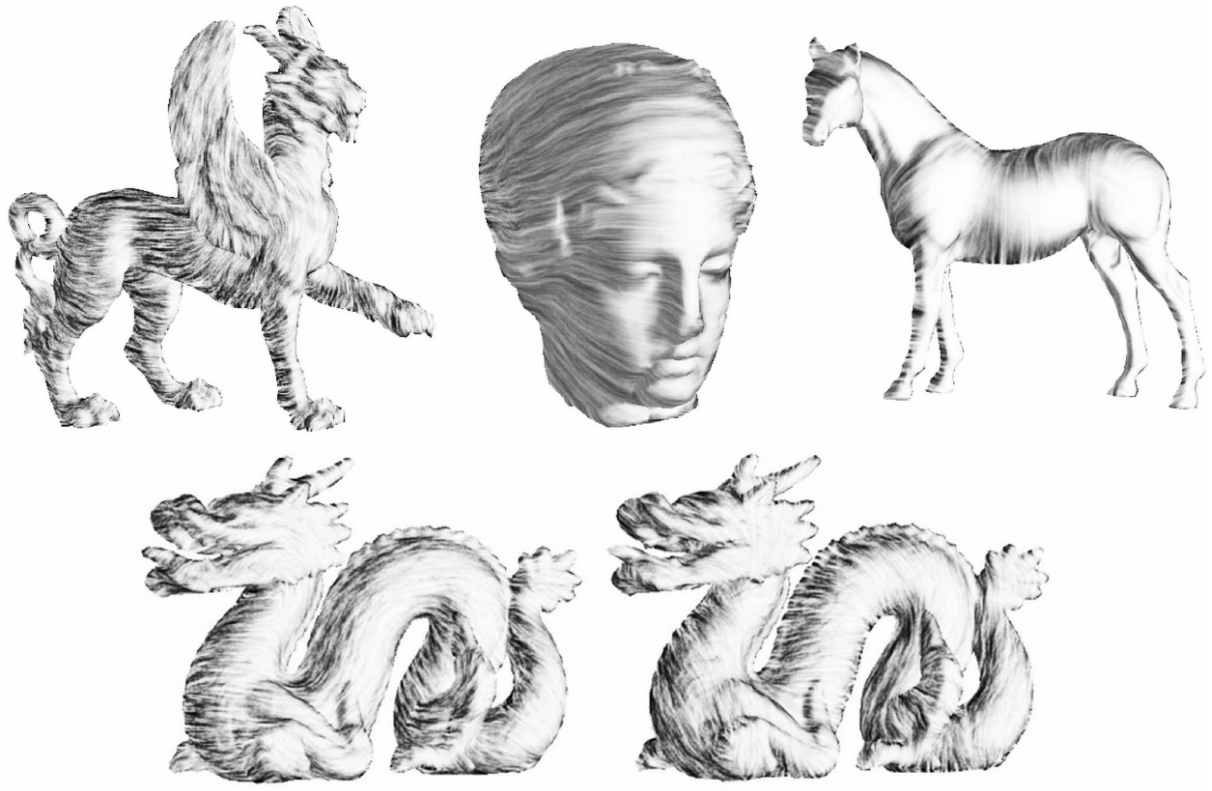

Fig. 21. We have applied vector field design to nonphotorealistic illustration of 3D surfaces. The pencil style illustration is based on van Wijk's image-based flow visualization technique [2003]. The vector field used for the dragon image in the lower-right was obtained by rotating the vector field from the lower-left by $\frac{\pi}{3}$.

8. CONCLUSION AND FUTURE WORK

Vector field design on surfaces is an important problem that has received relatively little attention. We have identified a number of graphics applications, such as nonphotorealistic rendering and texture synthesis, for which a vector field design system is needed. We also propose a set of requirements for a vector field design system. Namely, the user can create a wide variety of vector fields (not some subclass) with relatively little effort. Also, the user has control over vector field topology.

We present a vector field design system for both planar domains and 3D surfaces. To our knowledge, this is the first system that produces continuous vector fields on mesh surfaces and provides control over the number and location of singularities. The system has a three-stage pipeline: initialization, analysis, and editing. The editing operations are at the core of our system. To make the system fully functional, we have introduced algorithms to resolve several problems. Many of these problems are challenging in and of themselves.

(1) The piecewise interpolation scheme for mesh surfaces is novel, and enables efficient vector field analysis and editing on meshes. To our knowledge, existing singularity pair cancellation techniques work only for planar domains.

(2) We describe a new technique to construct surface basis vector fields that is based on the concepts of geodesic polar maps and parallel transport.

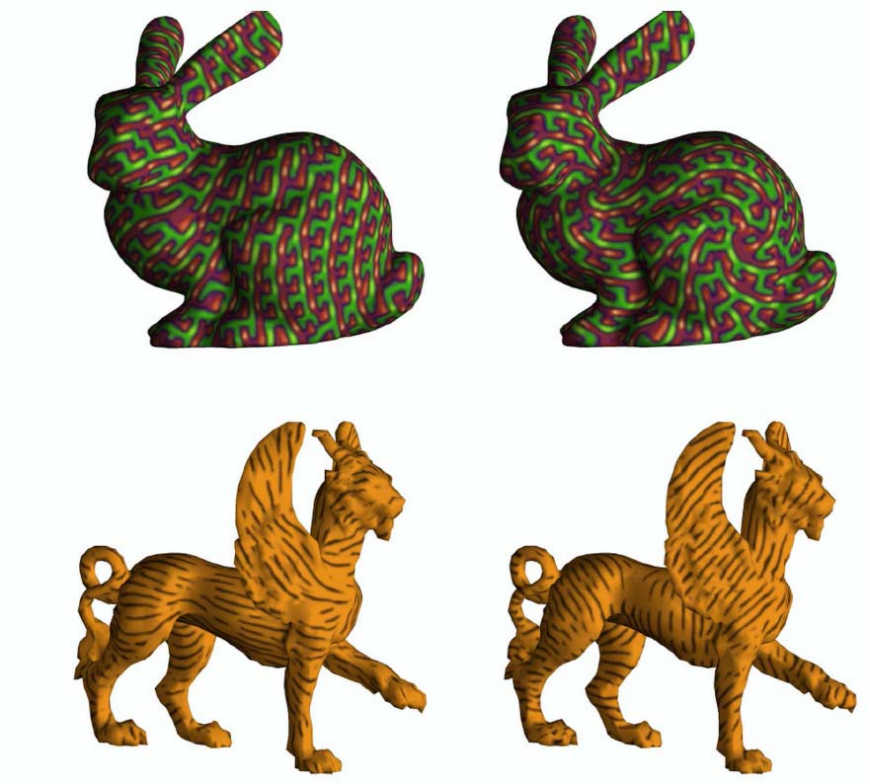

Fig. 22. This figure shows the results of applying our surface vector field design system to texture synthesis. Two vector fields are used for each model: the bunny, and the feline. Notice that singularities lead to the breakup of synthesis patterns (upper-right). Also, the spirals around these singularities are obvious in the synthesis result. For anisotropic textures, different vector fields lead to different visual appearances (compare the bunny images and feline images, respectively).

(3) We allow the user to control the location of singularities with a novel singularity movement operation. Furthermore, we provide a unified framework for implementing both singularity pair cancellation and singularity movement based on Conley index theory, which is more general and powerful than the well-known Poincaré index. To our knowledge, this is the first time Conley index theory has been applied to computer graphics. Furthermore, the region optimization technique that we describe helps to produce smooth vector fields after editing operations have been performed.

(4) We use flow rotations and reflections to overcome the numerical instabilities associated with regions of high curl and regions near saddles, which allows control over any linear singularity.

There are a number of issues that we wish to improve upon in our current system. First, we have used the same decay constant d for all design elements when creating basis vector fields (Eqs. (2) and (3)). It may be desirable to let the user control this constant. Second, our topological editing algorithms sometimes produce regions that are larger than necessary. This means that the behavior of the flow may be changed at places that are far away from the user-specified singularities. We plan to investigate ways of restricting the size of such regions. Third, we rely on the user to specify the singularities pair for cancellation. It may be useful to provide the function "automatic singularity pairing for elimination," in which the user specifies one singularity to be removed and the system determines another for cancellation. Finally, our surface vector field design system currently only handles closed surfaces.

Surfaces with holes may cause incorrect results when computing geodesic polar maps, and we use hole filling techniques to remove these holes. It is desirable to consider other approaches in which surfaces with holes can be handled directly, without filling.

There are several possible areas of future research. So far, we have focused on controlling the singularities in a vector field. It is natural to ask for controls over separatrices and periodic orbits, which are also part of vector field topology. Can we extend the concept of singular elements and allow the creation of canonical separatrices and periodic orbits? What editing operations are necessary to edit them? Finally, and maybe more fundamentally, what applications will benefit from these operations?

Singularity pair cancellation can be seen as performing a particular type of *bifurcation* if we track the continuous change that is involved. Many other types of bifurcations exist. They are interesting mathematically, and also have applications for scientific visualization.

Vector field design for surfaces might be extended to handle vector fields that are defined for volumes or other higher-dimensional datasets. Also, we are interested in identifying other applications for vector field design, such as fluid simulation and hairstyle design. Fluid simulation will require the ability to create wavy fields, as well as time-dependent flows.

Another important area of application is for educational purposes, in which students learn important concepts of vector fields through creating and manipulating vector fields and observing the changes.

ACKNOWLEDGMENTS

We would like to thank the following people and groups for the 3D models they provided: Mark Levoy and the Stanford Graphics Group, Andrzej Szymczak, and Cyberware. We also appreciate the discussions with Jarek Rossignac, Andrzej Szymczak, John C. Hart, and Todd Moeller. Thanks to James Hays and Irfan Essa for their high-quality painterly rendering program that produces the painterly images shown in this article. Furthermore, we are grateful to the help of Spencer Reynolds in preparing the video. The image of the human eye is from pics4learning.com. Finally, we wish to thank our anonymous reviewers for their valuable comments and suggestions.

REFERENCES

ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LÉVY, B., AND DESBRUN, M. 2003. Anisotropic polygonal remeshing. *ACM Trans. Graph. 22*, 3, 485–493.

CALCATERRA, C. AND BOLDT, A. 2003. Flow-Box theorem for Lipschitz continuous vector fields. http://www.citebase.org/cgi-bin/citations?id=oai:arXiv.org:math/0305207.

CASH, J. R. AND KARP, A. H. 1990. A variable order Runge-Kutta method for initial value problems with rapidly varying right-hand sides. *ACM Trans. Math. Softw. 16*, 201–222.

CONLEY, C. 1978. *Isolated Invariant Sets and the Morse Index*. American Mathematics Society, Providence, RI.

EDELSBRUNNER, H., HARER, J., AND ZOMORODIAN, A. 2003. Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds. *Discrete Comput. Geom. 30*, 87–107.

EDELSBRUNNER, H., LETSCHER, D., AND ZOMORODIAN, A. 2002. Topological persistence and simplification. *Discrete Comput. Geom. 28*, 511–533.

FLOATER, M. S. 2003. Mean value coordinates. *Comput. Aided Geom. Des. 20*, 1, 19–27.

GIRSHICK, A., INTERRANTE, V., HAKER, S., AND LEMOINE, T. 2000. Line direction matters: An argument for the use of principal directions in 3d line drawings. In *Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering*. 43–52.

HALE, J. AND KOCAK, H. 1991. *Dynamics and Bifurcations*. Springer-Verlag, New York.

HAUSER, H., LARAMEE, R. S., AND DOLEISCH, H. 2002. State-of-the-Art report 2002 in flow visualization. Tech. Rep. TR-VRVis-2002–003, VRVis Research Center, Vienna, Austria. Jan.

HAYS, J. H. AND ESSA, I. 2004. Image and video-based painterly animation. In *Proceedings of the NPAR: 3rd International Symposium on Non-Photorealistic Animation and Rendering*. 113–120.

HELMAN, J. L. AND HESSELINK, L. 1991. Visualizing vector field topology in fluid flows. *IEEE Comput. Graph. Appl. 11*, 3, 36–46.

HERTZMANN, A. 1998. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. 453–460.

HERTZMANN, A. AND ZORIN, D. 2000. Illustrating smooth surfaces. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. 517–526.

HIRSCH, M. AND SMALE, S. 1974. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, London.

KACZYNSKI, T., MISCHAIKOW, K., AND MROZEK, M. 2004. *Computational Homology*. Springer, New York.

KIMMEL, R. AND SETHIAN, J. A. 1998. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences of the United States of America 95*, 15, 8431–8435.

MISCHAIKOW, K. 2002. Topological techniques for efficient rigorous computation in dynamics. *Acta Numerica*, 435–478.

MISCHAIKOW, K. AND MROZEK, M. 2002. Conley index. *Handbook of Dynamic Systems, North-Holland 2*, 393–460.

NI, X., GARLAND, M., AND HART, J. C. 2004. Fair Morse functions for extracting the topological structure of a surface mesh. *ACM Trans. Graph. 23*, 3, 613–622.

POLTHIER, K. AND PREUß, E. 2003. Identifying vector fields singularities using a discrete hodge decomposition. In *Mathematical Visualization*. Springer-Verlag, New York. 112–134.

POLTHIER, K. AND SCHMIES, M. 1998. Straightest geodesics on polyhedral surfaces. In *Mathematical Visualization*. Springer-Verlag, New York, 135–150.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. 465–470.

PRAUN, E., HOPPE, H., WEBB, M., AND FINKELSTEIN, A. 2001. Real-Time Hatching. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. 581–586.

ROCKWOOD, A. AND BUNDERWALA, S. 2001. A toy vector field based on geometric algebra. In *Proceedings of the Application of Geometric Algebra in Computer Science and Engineering Conference*. 179–185.

SCHEUERMANN, G., KRGER, H., MENZEL, M., AND ROCKWOOD, A. P. 1998. Visualizing nonlinear vector field topology. *IEEE Trans. Visualization Comput. Graph. 4*, 2, 109–116.

STAM, J. 2003. Flows on surfaces of arbitrary topology. *ACM Trans. Graph. 22*, 3, 724–731.

THEISEL, H. 2002. Designing 2d vector fields of arbitrary topology. *Comput. Graph. Forum 21*, 3, 595–604.

THEISEL, H. AND WEINKAUF, T. 2002. Vector field metrics based on distance measures of first order critical points. *WSCG 10*, 3, 121–128.

TONG, Y., LOMBEYDA, S., HIRANI, A., AND DESBRUN, M. 2003. Discrete multiscale vector field decomposition. *ACM Trans. Graph. 22*, 3 (July), 445–452.

TRICOCHE, X. 2002. Vector and tensor field topology simplification, tracking, and visualization. Ph.D. thesis, Universität Kaiserslautern.

TRICOCHE, X., SCHEUERMANN, G., AND HAGEN, H. 2001. Continous topology simplification of planar vector fields. In *Proceedings of the IEEE Visualization, Conference*, 159–166.

TURK, G. 2001. Texture synthesis on surfaces. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. 347–354.

VAN WIJK, J. J. 2002. Image based flow visualization. *ACM Trans. Graph. 21*, 3, 745–754.

VAN WIJK, J. J. 2003. Image based flow visualization for curved surfaces. In *Proceedings of the IEEE Visualization Conference*, 123–130.

WEI, L. Y. AND LEVOY, M. 2001. Texture synthesis over arbitrary manifold surfaces. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. 355–360.

WEJCHERT, J. AND HAUMANN, D. 1991. Animation aerodynamics. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*. 19–22.

WELCH, W. AND WITKIN, A. 1994. Free-Form shape design using triangulated surfaces. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. 247–256.

WESTERMANN, R., JOHNSON, C., AND ERTL, T. 2000. A level-set method for flow visualization. In *Proceedings of the IEEE Visualization Conference*. 147–154.

WISCHGOLL, T. AND SCHEUERMANN, G. 2001. Detection and visualization of planar closed streamline. *IEEE Trans. Visualization Comput. Graph. 7*, 2, 165–172.

Received October 2004; revised December 2005; accepted August 2006