HAL
open science

Representing Higher-Order Singularities in Vector Fields on Piecewise Linear Surfaces

Wan Chiu Li, Bruno Vallet, Nicolas Ray, Bruno Lévy

To cite this version:

Wan Chiu Li, Bruno Vallet, Nicolas Ray, Bruno Lévy. Representing Higher-Order Singularities in Vector Fields on Piecewise Linear Surfaces. IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization '06), Oct 2006, USA. inria-00105598

HAL Id: inria-00105598

https://inria.hal.science/inria-00105598v1

Submitted on 11 Oct 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Representing Higher-Order Singularities in Vector Fields on Piecewise Linear Surfaces

Wan-Chiu Li, Bruno Vallet, Nicolas Ray, and Bruno Lévy

Abstract—Accurately representing higher-order singularities of vector fields defined on piecewise linear surfaces is a non-trivial problem. In this work, we introduce a concise yet complete interpolation scheme of vector fields on arbitrary triangulated surfaces. The scheme enables arbitrary singularities to be represented at *vertices*. The representation can be considered as a *facet-based* "encoding" of vector fields on piecewise linear surfaces. The vector field is described in *polar coordinates* over each facet, with a facet edge being chosen as the reference to define the angle. An integer called the *period jump* is associated to each edge of the triangulation to remove the ambiguity when interpolating the direction of the vector field between two facets that share an edge. To interpolate the vector field, we first *linearly* interpolate the angle of rotation of the vectors along the edges of the facet graph. Then, we use a variant of Nielson's *side-vertex* scheme to interpolate the vector field over the entire surface. With our representation, we remove the bound imposed on the complexity of singularities that a vertex can represent by its connectivity. This bound is a limitation generally exists in vertex-based linear schemes. Furthermore, using our data structure, the index of a vertex of a vector field can be *combinatorily* determined.

We show the simplicity of the interpolation scheme with a GPU-accelerated algorithm for a LIC-based visualization of the so-defined vector fields, operating in image space. We demonstrate the algorithm applied to various vector fields on curved surfaces.

Index Terms—vector field visualization, higher-order singularities, line integral convolution, GPU.

1 Introduction

Vector fields on surfaces are important objects which appear frequently in scientific simulation in CFD (Computational Fluid Dynamics) or modeling by FEM (Finite Element Method) [21]. To be visualized, such vector fields are usually linearly interpolated for the sake of simplicity and performance considerations. Namely, the vector field is sampled at each vertex of the underlying piecewise linear surface and interpolated linearly, similarly to what is done for the geometry. However, the price that one pays for the simplicity that comes with the linearity is that only linear (or first-order) singularities can be represented in the triangles [24]. This leads to the mis-representation of some vector fields where higher-order singularities are present (e.g. a dipole of a magnetic field). This mis-representation of the singularities changes the perception of the topology of the vector fields and hence may lead to mis-interpretation of the simulation results. Therefore, finding a simple representation that enables the meaningful information of the higher-order singularities to be preserved is a research direction worths exploring.

Among the pioneers in this direction, Tricoche *et al.* [25] explored the possibility of representing higher-order singularities using only piecewise linear 2D vector field. Their approach creates higher-order singularities at vertices by using a sequence of the three basic sector types (see Figure 3). The technique had been applied to simplify the topology of 2D vector fields [30]. In [23], Theisel proposed an approach that enables one to design piecewise linear vector field and hence vector field compression. The main limitation of the purely piecewise linear schemes is that the complexity of singularities that a vertex can represent is limited by its connectivity. Moreover, the elliptic sectors need to be split in two due to their non-linearity.

In [19], Scheuermann *et al.* proposed a mixed higher-order/linear vector field scheme to visualize non-linear vector field topology. Some non-linear schemes, for instances, [20, 32] were also proposed to represent higher-order singularities. Unfortunately, mixed and non-linear schemes impede the acceleration by graphics hardwares. It is therefore difficult to achieve an interactive frame rate with those approaches.

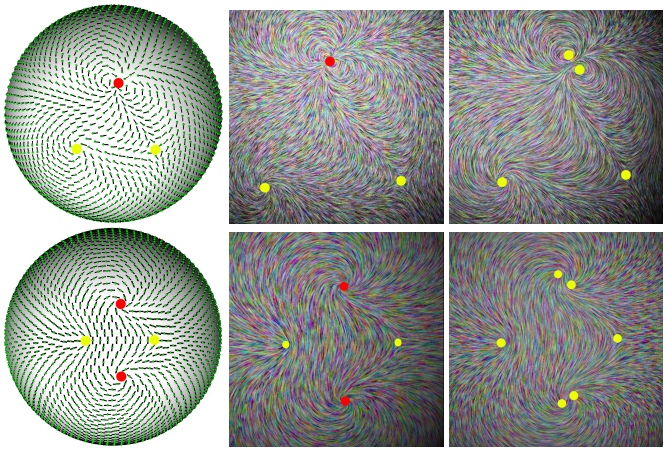

Fig. 1. Two examples of non-linear vector fields on the sphere represented by our facet-based data structure (left). First-order (yellow) and higher-order (red) singularities are defined at vertices. The top one is a vector field (with singularities of indices 2,1,-1). The bottom one is a vector field with sign ambiguity (with singularities of indices 1,1/2,-1/2). The vector fields are visualized using our LIC-based method (middle). On the right, we show the same two vector fields represented by the classic vertex-based piecewise linear representation. Note that the higher-order singularities are split into combinations of first-order singularities.

1.1 Contributions

In the context of interactive vector field visualization, linear interpolation schemes are often preferred due to both their simplicity and performance issues. However, they are limited by the bound imposed on the complexity of the singularities that a vertex can represent by its connectivity. In this work, we present a novel scheme to "encode" vector fields on piecewise linear surfaces such that any higher-order singularity can be represented without the mentioned upper bound.

In our setting, the vector field is piecewise defined over the surface

- *Wan-Chiu Li is with INRIA-Alice, France, E-mail: wan-chiu.li@loria.fr.*
- *Bruno Vallet is with INRIA-Alice, France, E-mail: vallet@loria.fr.*
- *Nicolas Ray is with University Nancy 2, France, E-mail: ray@loria.fr.*
- *Bruno Lévy is with INRIA-Alice, France, E-mail: levy@loria.fr.*

Manuscript received 31 March 2006; accepted 1 August 2006; posted online 6 November 2006.

For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

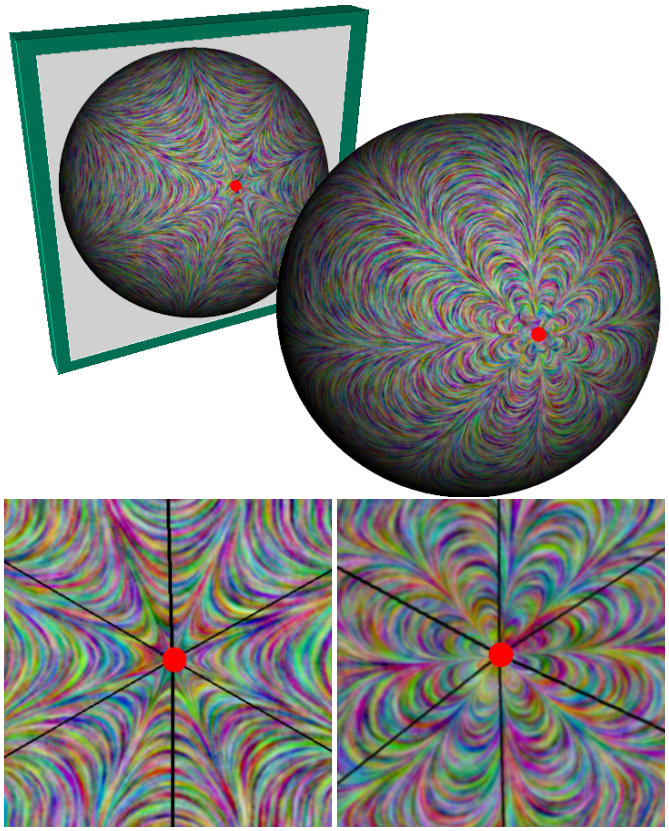

Fig. 2. Top: a sphere with only two higher-order singularities of index +5 (front) and -3 (back), the sum of which equals the Euler characteristic (2 for a sphere). Bottom: zooming in the vicinity of the singularities. One can observe that the hyperbolic and elliptic structure of the curvilinear sectors, which have been described in Tricoche *et al.*'s work (see Figure 3). Note that using our approach, the separatrices of the curvilinear sectors do not need to appear on the incident edges (black lines) of the vertex.

by a simple interpolation scheme, over the cells of the subdivision complex (or barycentric triangulation, as shown Figure 5 further in the paper). We give a detailed description in Section 2. In short, the vector field is described in polar coordinates on each facet. In addition, an integer that we call the *period jump* specifies the number of turns that the vector field undergoes when crossing the edge. Hence, we remove the ambiguities when interpolating the field across an edge shared by two facets. To interpolate the vector field over the whole surface, we proceed as follows:

- ◇ first, we *linearly* interpolate the direction and norm of the vector field along each dual edge;
- ◇ then, we interpolate the vector field over the whole surface using a variant of *side-vertex* interpolation [15].

The interpolation scheme combined with our notion of period jump enable us to represent non-linear curvilinear sectors in a cell of the subdivision complex (which is not possible in Tricoche *et al.*'s scheme in which they have to split elliptic sectors into two triangles).

Hence, we remove the bound imposed on the complexity of singularities that a vertex can represent by its connectivity. Using our representation, one can represent arbitrary indices (see Section 3). Moreover, by construction, singularities can only exist at vertices of the surface and the index of each vertex can be found *combinatorily*.

The reason that we define the vector on the facets is that, on a surface in $\mathbb{R}^3$, due to the curvature, i.e. geometry of the embedding, two tangent vectors on the surface defined at two vertices need a parameterization, either local or global, to be compared. On the other hand, comparing two vectors defined on a pair of adjacent facets can be simply done by flattening the pair due to the zero Gaussian curvature along the common edge.

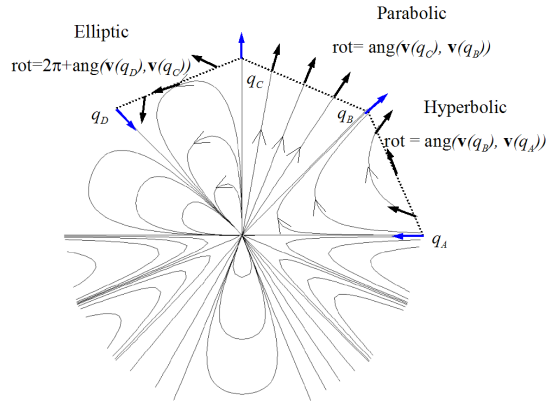

Fig. 3. Modeling a higher order singularity with the three types of curvilinear sectors by Tricoche *et al.*'s method

In order to demonstrate the simplicity of the representation, we show a GPU-accelerated LIC-based visualization of the so-defined vector fields, operating in image space.

The remainder of the paper is organized as follows. We will first review the previous work on the representation of higher-order singularities in vector fields as well as vector field visualization by line integral convolution. Then, in Section 2, we will introduce our representation of vector fields. Section 3 deals with the particular issue of representing arbitrary singularities on piecewise linear surfaces. In Section 4, we will explain how to convert existing vector fields into our representation. The LIC-based visualization algorithm accelerated by GPU is presented in Section 5. In Section 6, the results are discussed. Finally, conclusions are drawn.

1.2 Previous work

In this section, we review the previous work that concerns the representation of higher-order singularities, using linear and non-linear interpolation schemes. We also review the methods to visualize vector fields on surfaces.

1.2.1 Representing Higher-order Singularities in Vector Fields

Different methods have been proposed to encode vector fields on surfaces, using either linear schemes, or more sophisticated higher-order schemes. In all these methods, the vector fields are supposed to be sampled at the vertices of the piecewise linear domains.

Linear schemes:
Tricoche *et al.* [25] explored the possibility of representing non-linear singularities with a piecewise linear vector field. They proposed a way to model singularities with complex topologies at a vertex and conversely, given a vertex, to classify the possible types of singularities encountered at that vertex. They pointed out that a singularity in a 2D vector field located at a vertex is generally non-linear since there is no neighborhood of the singularity completely lying in the definition domain of a single linear field. By construction, the complexity of the singularity at a vertex is limited by its connectivity, namely, the number of incident triangles. They "synthesize" non-linear singularities at vertices by using sequences of parabolic, hyperbolic and elliptic sectors (see Figure 3). This is based on the observation that the vicinity of a critical point in a 2D vector field can be characterized as a sequence of these three basic sector types [1, 5]. In fact, they can be understood as three possible interpolations of two radial vectors originated from the vertex in the counter-clockwise direction. In the parabolic and hyperbolic sectors, the angle of rotation is defined as $ang(\mathbf{v}(q_2), \mathbf{v}(q_1))$, where $ang(.,.)$ means the natural rotation which is always in $(-\pi, \pi]$. This angle is always positive (resp. negative) in

parabolic (resp. hyperbolic) sector. In the elliptic sector, the angle of rotation defined as $2\pi + ang(\mathbf{v}(q_2), \mathbf{v}(q_1))$. Note that since the elliptic sector is non-linear, it has to be split into two triangles when being modeled by linear triangles. In [30], the above scheme was used to simplify 2D vector fields. First, singularities to be merged are clustered. Then, each cluster is represented by a piecewise linear vector field with only one higher-order singularity.

In [23], Theisel introduced a method to design piecewise linear vector fields with prescribed topology. With his approach, all types of critical points of a 2D vector field can be represented. As with Tricoche *et al.*'s method, higher-order singularities are "synthesized" at vertices with the three basic sector types. Thus, as far as higher-order singularities are concerned, this method also suffers from the fact that the maximum complexity of a singularity at a vertex is bounded by the number of triangles meeting at that vertex.

Weinkauf *et al.* [28] extended the idea in [23] to 3D vector fields. Similarly to the 2D case, the vicinity of a critical point is segmented into sectors with a regular behavior of the flow. They achieved the construction of higher-order 3D vector fields by designing their topological skeletons.

In [29], an approach is proposed to extract and classify higher-order singularities of 3D vector fields. This is achieved by extracting the topological skeleton of a 2D vector field on a convex surface around the area of interest. They demonstrated vector field simplification as an application for this method.

Mixed schemes and non-linear schemes:
In [19] Scheuermann *et al.* proposed a method to first detect higher-order singularities by computing the indices of the triangles in the non-linear singular regions and then approximating the singularities by using polynomials of high degrees. In the non-singular regions, they use piecewise linear interpolation to represent the field.

To improve the *approximation* of higher-order singularities in a given non-linear vector field, Scheuermann *et al.* [20] proposed a C^1 interpolation scheme (based on Powell-Sabin and Nielson's interpolants) for 2D vector field defined on a piecewise linear planar domain. Although higher-order singularities are still split into combinations of first-order ones, the overall topology of the vector field is better reflected by their C^1 interpolation scheme as compared to previous work.

In [32], Zhang *et al.* presented a non-linear scheme to represent non-linear singularities through the use of exponential maps (around the one-ring of a vertex) and parallel transport. The non-linear property makes the interpolation scheme quite complicated and computationally expensive. More specifically, the construction of the exponential map, which is basically a local parameterization of the one-ring of a vertex, impedes the acceleration by graphics hardware. They extended the definitions to tensor fields in [31].

1.2.2 Visualization of Vector Fields on Surfaces

Since the introduction of LIC (Line Integral Convolution) by Cabral and Leedom in 1993, the method has been widely used for the visualization of 2D vector fields. In this work, we have also chosen to use the concept of LIC to visualize our vector fields. After the invention of this method, much work have been done to accelerate the original algorithm. For instance, in [22] Stalling and Hege presented the FastLIC method, which employs simple box filters. This method minimizes the total number of streamlines and hence accelerates the original LIC by an order of magnitude.

Later, work have been done to extend the 2D LIC technique to visualize vector fields on surfaces. As said in [14], these methods can be classified into mainly three types according to the space in which LIC operates:

- *Parametric space:* Forssell and Cohen [7, 8] proposed a method that allows the operation of LIC through a parameterization [6] of the surface. However, a global parameterization of a general surface may not always be obtained easily. Battke *et al.* [2] and Carr *et al.* [4] proposed methods that allow the operation of LIC in parametric space per triangle packing algorithms. Triangle packing can be considered as the simplest local parameterization of the surface by triangle. Nevertheless, the drawback of the triangle packing is that it is quite sensitive to the quality of the mesh.
- *Object space:* Immersing the vector field on the surface into a 3D volume enables the field to be treated as 3D, which means that streamline tracing can be performed [18, 12].
- *Image space:* Recently two methods, ISA (Image Space Advection) [13] and IBFVS (Image Based Flow Visualization for Curved Surfaces), [26] have been proposed to enable high-performance visualization of flow on surfaces. A side-by-side comparison of the two methods can be found in [14]. Working in this space, no parameterization of the surface is needed. Moreover, LIC can be accelerated by taking advantage of graphics hardware [10]. Seeing the advantages of image space based methods, in this work, we have chosen to carry out LIC in image space as well.

2 Our Discrete Vector Field Representation

In this paper, bold letters ($\mathbf{n}$, $\mathbf{v}$, $\mathbf{u}$...) denote vectors in $\mathbb{R}^3$ defined on a connected oriented mesh $M = < \mathscr{V}, \mathscr{E}, \mathscr{F} >$, where $\mathscr{V}, \mathscr{E}, \mathscr{F}$ denotes the set of vertices, edges and facets respectively. By oriented, we mean facets and edges are oriented. The orientation allows the definition of the normal $\mathbf{n}$ on each facet. The vector field $\mathbf{v}$ we handle in this paper is embedded in the mesh, hence as a 3-vector, it needs to satisfy $\mathbf{v} \cdot \mathbf{n} = 0$. This section explains how to interpolate such a field on M.

2.1 Local Basis and Polar Coordinates

The first step in defining an interpolation for $\mathbf{v}$ is to define a local orthonormal basis $(\mathbf{x}(f), \mathbf{y}(f))$ on each facet $f \in \mathscr{F}$. A convenient way to do so is to choose $\mathbf{x}(f)$ to be a unit vector along one of the (oriented) edges of f, and take $\mathbf{y}(f) = \mathbf{n} \times \mathbf{x}(f)$. The main idea is that we interpolate in polar coordinates in the local basis. Hence on each facet, we write:

$$\mathbf{v} = \begin{pmatrix} r\cos\theta \\ r\sin\theta \end{pmatrix} = r\mathbf{u} \quad r = ||\mathbf{v}|| \quad \mathbf{u} = \frac{\mathbf{v}}{||\mathbf{v}||} = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \tag{1}$$

where r denotes the norm of $\mathbf{v}$, and $\mathbf{u}$ its direction. The norm and direction of $\mathbf{v}$ will be interpolated separately. Being a scalar, the norm can be interpolated easily. On the other hand, the direction interpolation will be more tricky. The direction $\mathbf{u}$ is parameterized by its angle $\theta \in \mathbb{R}$. The fact that we take the angle in $\mathbb{R}$ and not in $(-\pi, \pi]$ is very important. It allows the continuity of a direction on a triangle to be defined, even if the direction rotates by more than 2π within the triangle. This choice gives us more freedom for the interpolation, but generates an ambiguity. We will now explain how to handle such ambiguities in interpolating directions.

2.2 Direction Interpolation Ambiguity

Let us start with a simple example to explain direction interpolation ambiguity. Assume that we want to interpolate a direction $\mathbf{u}$ along an edge $[AB]$ knowing $\mathbf{u}(A)$ and $\mathbf{u}(B)$. If $\mathbf{u}$ is continuous, the angular variation along $[AB]$ satisfies:

$$\Delta\theta([AB]) = \int_A^B d\theta = ang(\mathbf{u}(B), \mathbf{u}(A)) + 2p\pi \tag{2}$$

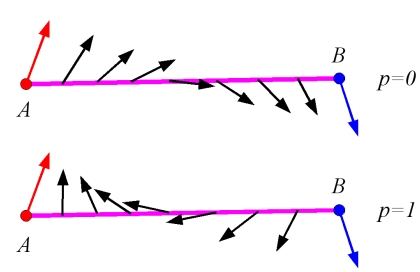

Fig. 4. Direction interpolation ambiguity: between two directions given at points A and B, we have different possible interpolations, which can be chosen using the variable p.

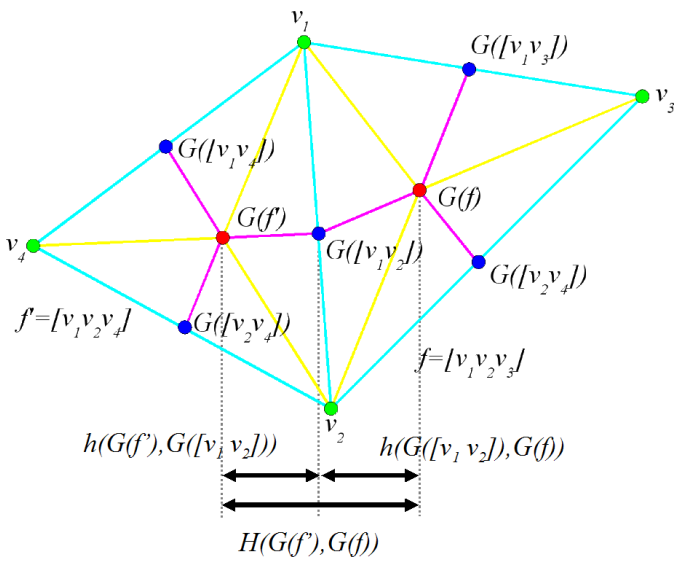

Fig. 5. Illustration on two adjacent (primal) triangles of a primal mesh (light blue) and its barycentric dual (magenta). Subdivision simplices are triangles based on a primal vertex (green), a dual vertex (red), and an edge middle (dark blue). Their edges are based on primal and dual edges, and edges between primal and dual vertices (yellow).

where we call $p \in \mathbb{Z}$ the period jump, and $ang(.,.)$ the angle in $(-\pi,\pi]$. The angular variation has infinitely many possible values for all the possible values of p (see Figure 4). Hence to interpolate the direction field on $[AB]$, we need both the values at A and B and an integer p (period jump), such that the linearly interpolated value at a point $P = (1-t)A + tB$ is written:

$$\theta(P) = \theta(A) + \Delta\theta([AB])t \tag{3}$$

We can take $\theta(A) \in (-\pi,\pi]$, but according to the choice for p, $\theta(B)$ can be outside this interval. The choice of the period jump p is of great importance, and we explain in Section 4 how to compute it when converting an existing vector field into our structure.

Using this idea of integer period jumps to remove the direction interpolation ambiguity, we will now explain how to interpolate a vector field on the mesh M in three steps of increasing dimension: facet centers (0D), dual edges (1D), mesh (2D). This interpolation may be thought of as a variant of the "side-vertex" interpolation scheme [15]. As the reader will see in the following, in our case, the value along the side is interpolated linearly while it is constant (identical to the side value) along a side-vertex path.

2.3 Step 1: 0D

From now on, we call $[v_0 ... v_i]$ the simplex (edge or triangle) based on points $v_0 ... v_i$ and $G(s)$ the gravity center of a simplex, s. The first step of the interpolation is to define the vector $\mathbf{v}(f) = r(f)\mathbf{u}(f)$ at the gravity center $G(f)$ of each facet $f \in \mathscr{F}$. $r(f)$ is simply a scalar, and $\mathbf{u}(f)$ at $G(f)$ can be defined by its angle $\theta(f) \in \mathbb{R}$, which is the angle between $\mathbf{u}(f)$ and the facet reference vector, $\mathbf{x}(f)$. Notice that $\mathbf{v}(f)$, $\mathbf{u}(f)$, $r(f)$ and $\theta(f)$ are values given at $G(f)$. They will be interpolated, so they are not constant on the whole facet, but we omit the G for the sake of brevity since it is not ambiguous. Moreover, it corresponds to the implementation where a couple $\theta(f)$, $r(f)$ is stored for each facet.

2.4 Step 2: 1D

The second step of the interpolation is to define the direction field on the edges of the barycentric dual of M (see Figure 5). For an edge $e = [v_1v_2]$ between a pair of adjacent triangles $f = [v_1v_2v_3]$ and $f' = [v_2v_1v_4]$, we can geometrically define the barycentric dual edge:

$$e^* = [G(f)G([v_1v_2])] \cup [G([v_1v_2])G(f')] \tag{4}$$

Given a period jump $p(e^*)$ (see Section 4 for the choice of $p(e^*)$), the angular variation along e^* is given by:

$$\Delta\theta(e^*) = \int_{e^*} d\theta = ang(\mathbf{x}(f'),\mathbf{x}(f)) + \theta(f') - \theta(f) + 2p(e^*)\pi \tag{5}$$

where the angle $ang(.,.)$ is measured after flattening the pair of triangles f, f' along their common edge e. The main difficulty of this step is to split the angular variation along the two parts of the dual edge. We chose to split according to the height ratio above the common edge (see Figure 5):

$$\alpha(G(f),G([v_1v_2])) = \frac{h(G(f),G([v_1v_2]))}{H(G(f'),G(f))} =$$

$$\frac{||\overrightarrow{v_1G(f)} \times \overrightarrow{v_1v_2}||}{||\overrightarrow{G(f')G(f)} \times \overrightarrow{v_1v_2}||} = \frac{||\overrightarrow{v_1v_3} \times \overrightarrow{v_1v_2}||}{||\overrightarrow{v_4v_3} \times \overrightarrow{v_1v_2}||} \tag{6}$$

This gives a natural linear interpolation for $P \in [G(f)G([v_1v_2])]$ given in barycentric coordinates $P = (1-t)G(f) + tG([v_1v_2])$:

$$\theta(P) = \theta(f) + \Delta\theta(e^*)\alpha t \tag{7}$$

For the norm, the same considerations lead to a linear interpolation of the form:

$$r(t) = r(f) + \Delta r(e^*)\alpha t \tag{8}$$

with $\Delta r(e^*) = r(f') - r(f)$

2.5 Step 3: 2D

The third step is to interpolate the field over the whole mesh, by a piecewise defined interpolation on the subdivision simplices $S_{i,j,k}$ of the mesh M. The subdivision simplices are simply defined as the triangles

$$S_{i,j,k} = [v_iG([v_iv_j])G([v_iv_jv_k])] \quad \forall (i,j,k) | [v_iv_jv_k] \in \mathscr{F} \tag{9}$$

Notice that because of the possible permutations for (i,j,k), there are six subdivision simplices per facet of the mesh (see Figure 5). On each subdivision simplex of a facet $f = [v_iv_jv_k]$, the field is known on the edge $[G([v_iv_j])G(f)]$. As for the remainder of the triangle, it is interpolated such that the vector is constant along each segment between a point of this edge and the primal vertex v_i. Another way of saying that is that we obtain the vector at a point P as the vector at the intersection P' between (v_i,P) and $[G([v_iv_j])G(f)]$ (see Figure 6). This interpolation may be thought of as a variant of the "side-vertex" interpolation scheme [15]. In our case, the side value is interpolated linearly while along a side-vertex path, the value is constant, which equals to the side value. This gives a very simple expression in barycentric coordinates. If we write:

$$P = (1-t')v_i + t'((1-t)G(f) + tG([v_iv_j]))$$

then we have:

$$P' = (1-t)G(f) + tG([v_iv_j]) \tag{10}$$

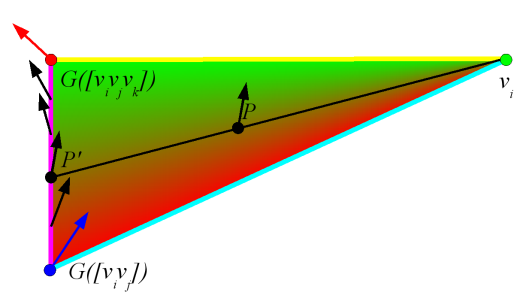

Fig. 6. Our "side-vertex" interpolation over a subdivision simplex: we obtain the interpolation value at P as the value at its projection P' on the (magenta) dual edge.

and we obtain the vector direction and norm by the interpolation described in Equations 7 end 8.

Our interpolation scheme has two advantages:

1) It is straightforward to implement on a GPU (see Section 5).
2) It allows arbitrary singularities to be represented at vertices of the mesh.

In the following section we describe and classify these singularities, and explain how our framework enables singularities of arbitrary index to be represented.

3 Vector Field Singularities

3.1 Singularity Classification

Vector field singularities in the plane have been much studied. For a 2D vector field $\mathbf{v} : (x,y) \in \mathbb{R}^2 \mapsto (v_x, v_y) \in \mathbb{R}^2$, a point q where the vector field vanishes is called a singularity or critical point. The first classification of singularities is done using the Jacobian matrix J of $\mathbf{v}$ at q:

$$J = \begin{pmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} \end{pmatrix} \tag{11}$$

The singularities are then classified by the two eigenvalues of J. If they are non-zero, the singularity is said to be linear or first-order. Otherwise the singularity is said to be non-linear or higher-order. A finer classification can be made according to other criteria (eigenvalues are real, complex or purely imaginary, real and imaginary parts are of same or opposite sign).

However, the Jacobian matrix is not always well behaved around q, and does not allow to classify higher order singularities. For instance, a dipole generated by a magnetic field has a singularity which "looks" different from the singularities classified above. To classify a wider class of singularities, we can use the *index* of a singularity, typically defined as:

$$2\pi I_{\mathbf{v}}(q) = \int_{\gamma(q)} d\theta \tag{12}$$

where $\gamma(q)$ is a closed curve making one counterclockwise turn around q and containing no other singularity, and θ is the angle between the direction $\mathbf{v}$ and a reference vector, such that $v_x = r\cos\theta$ and $v_y = r\sin\theta$. This integral is always an integer multiple of 2π, such that the index of a singularity is always an integer. Notice that it depends only on the direction of the field and not on its norm, hence the choice of polar coordinates to study singularities is justified.

It can be shown that the singularities for which the Jacobian is well defined at q can only have index -1, 0 or 1, whereas the dipole has a singularity of index 2. This comes from the fact that using the Jacobian at q leads to making a linear approximation of the field at q. This also explains why classical linear interpolations do not capture high index singularities. We will now show how our framework allows arbitrary indices to be represented very easily.

3.2 Singularities with Integer Indices

We have a very simple closed curve around a vertex v to compute its index: the set of all edges dual to the edges emanating from v. This curve is the boundary of the dual facet so we call it ∂v^*. However, because we are not in the plane anymore, the integral in Equation 12 does not necessarily lead to an integer value. This is easily corrected by adding the angle defect $A_d(v)$:

$$2\pi I_{\mathbf{v}}(v) + A_d(v) = \int_{\partial v^*} d\theta = \sum_{e^* \in \partial v^*} \Delta\theta(e^*) =$$

$$\sum_{e^* \in \partial v^*} ang(\mathbf{x}(f'), \mathbf{x}(f)) + 2p(e^*)\pi \tag{13}$$

Note that when summing along ∂v^*, the $\theta(f') - \theta(f)$ terms of the dual edges cancel out. As the angle defect, and the $ang(\mathbf{x}(f'), \mathbf{x}(f))$ solely depend on the geometry of the field, one can express Equation 13 in the following form:

$$I_{\mathbf{v}}(v) = I_0(v) + \sum_{e^* \in \partial v^*} p(e^*) \tag{14}$$

where $I_0(v) = 1/2\pi(\sum ang(\mathbf{x}(f'), \mathbf{x}(f)) - A_d(v))$ is a "geometric" index that solely depends only on the choice of the basis vector for each facet. Hence after computing I_0, the index of any vertex on M can be obtained very easily by summing period jumps around the vertex.

3.3 Singularities with Fractional Indices

For many applications, we are interested in directions in the weak sense, without orientation (there is a $\pm$ sign ambiguity). In that case, angles are only given in $(-\pi/2, \pi/2]$. For instance, an eigenvector field of a second-order tensor field needs such flexibility. Our framework allows such fields to be handled in a very simple manner by authorizing period jumps to be multiples of $1/2$, which will allow representation of more general singularities of index multiples of $1/2$ (see the bottom vector field in Figure 1). Authorizing period jumps to be multiples of $1/4$ allows even more general singularities to be captured, which applies for vector fields with higher levels of symmetry, referred to as "cross" fields in [11, 27, 16].

Before explaining how to visualize vector field represented by our data structure (Section 5), we explain in the next section how to convert existing vector fields into our representation.

4 Encoding An Existing Vector Field

To manipulate existing vector fields (static or dynamic) represented in other forms, it is necessary to convert them into our data structure. This is possible as long as the vector field $\mathbf{v}(q)$ to be converted is known at each point q of the surface M. This is the case in FEM modeling where the vector field is described as a sum of basis functions defined on the whole mesh. If the vector field we wish to convert is only known at discrete samples, we suggest to simply linearly interpolate it between the samples. The conversion is done in four steps:

1) Singularity placement: Singularities are detected in the original vector field. If a singularity is not at a vertex, in order to preserve the exact location of singularities, we need to add a vertex at its exact position by a facet-split operation, which leads to splitting the corresponding facet in three.
2) Choose any one of the three oriented edges as the reference vector $\mathbf{x}(f)$ on each facet.
3) Compute the angle $\theta(f) = ang(\mathbf{v}(G(f)), \mathbf{x}(f))$ and norm $r(f) = ||\mathbf{v}(G(f))||$ at each facet barycenter $G(f)$.
4) Compute the period jumps such that the field rotation along dual edges is preserved. This rotation is given by the integral of the angular field variation along the dual edge:

$$\Delta\theta(e^*) = \int_{e^*} d\theta = \int_{e^*} \frac{v_x dv_y - v_y dv_x}{v_x^2 + v_y^2}$$

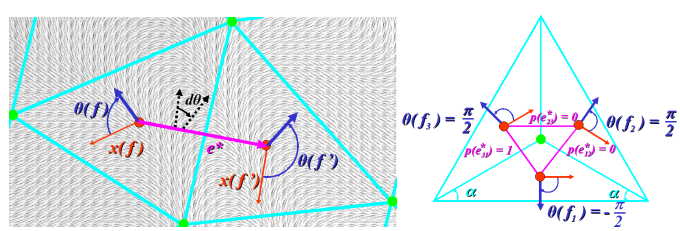

Fig. 7. Left: the period jump $p(e^*)$ must be set according to Equation 15 to ensure that the field rotation along the dual edge is the same as in the original vector field (visualized by the gray integral lines); Right: Converting a singularity of the type source. One can check that the index computed using Equation 14 equals to 1.

This value can be evaluated using the analytic representation of the field. Once the $\Delta\theta(e^*)$ is evaluated from the original field, one compute the period jump of the dual edge using Equation 5:

$$p(e^*) = \frac{1}{2\pi}(\Delta\theta(e^*) - ang(\mathbf{x}(f'), \mathbf{x}(f)) - \theta(f') + \theta(f)) \quad (15)$$

In Figure 7, we show an example of how to compute $\theta(f)$ and period jump $p(e^*)$. We illustrate as well a concrete example of converting a singularity of the type source (index=1) into our representation.

5 The GPU-based Visualization Algorithm

In this section we present an algorithm to visualize our vector fields. We adopted a GPU-accelerated LIC-based approach working in image space, which is inspired by the Image Space Advection method proposed in [13]. The algorithm is done in three passes (see Figure 8).

5.1 Pass 1: Depth Value for Geometric Discontinuities

In order to remove undesired visual continuity across the geometric discontinuities of surfaces when LIC is carried out in image space. We adopted the criteria to distinguish the geometric discontinuities proposed in [13] as follows.

$$\|z_{i+1} - z_i\| > \varepsilon \|q_{i+1} - q_i\| \quad (16)$$

where ε is a threshold chosen to be 0.15 for the results shown in this paper. The test compares the depth values in the object space z_i and z_{i+1} of two consecutive points q_i and q_{i+1} along the integral path in the image plane. A positive result of the test identifies a geometric discontinuity. To allow performing the above test in the LIC pass, the Z-buffer is rendered to the frame buffer and stored as a texture, T_{depth}.

5.2 Pass 2: Object to Image Space Projection

Since the LIC is performed in image space, the vectors at the points along the forward and backward line integral paths need to be known in advance. We achieve this by storing the vector field of the surface as follows. First, for each visible point of the surface, we compute its vector using the interpolation scheme as described in Section 2. We only interpolate the direction since we are only using the directional component of the vector for the LIC pass. The norm of the vectors can also be interpolated to give more effects of the visualization as done in [3]. More specifically, in our GPU implementation, we associate (1,0,0), (0,1,0) and (0,0,1) to the vertices of the subdivision simplex $G(f)$, $G([v_i v_j])$ and v_i respectively, where $f = [v_i v_j v_k]$, as their barycentric coordinates $(1-\lambda_1-\lambda_2, \lambda_1, \lambda_2)$. The direction of a point in the subdivision simplex (except v_i, where it is undefined) is given by the angle of rotation, θ measured with respect to the base direction $\mathbf{x}(f)$. Using Equations 10 and 7, the θ value is given by $\theta(f) + \Delta\theta\alpha\lambda_1/(1-\lambda_2)$. Then, the obtained vector in object space is projected onto the image plane, which is then normalized. The vector in image space is rendered to the frame buffer as color, which will be resampled as a texture, T_{vector}, in the third pass.

5.3 Pass 3: LIC in Image Space

Brief review: LIC (Line Integral Convolution) was proposed by Cabral and Leedom [3] to perform a texture synthesis for the visualization of 2D vector fields. By convolving an input white noise texture T_{noise} with a low-pass filter $\tau(w)$ along streamlines, the pixel intensity is highly correlated along individual streamlines but independent in the perpendicular direction. Considering a streamline $\gamma(s)$, line integral convolution gives the intensity for a pixel $x_0 = \gamma(s_0)$ as follows:

$$Intensity(x_0) = \int_{s_0-L}^{s_0+L} \tau(s-s_0) T_{noise}(\gamma(s)) ds \quad (17)$$

We carry out LIC in image space using a programmable GPU as follows: For a visible point on the surface projected onto the image space, starting from this point q_0, the streamline is traced in the positive and negative directions by coordinate advection. The advection is done by using vectors obtained by resampling the vector image T_{vector} stored in the first pass.

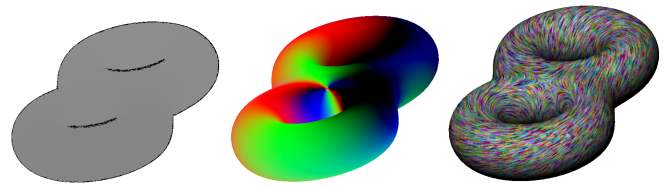

Fig. 8. The three passes of the LIC on GPU.Left: the geometric discontinuities identified; middle: vectors on the surface encoded in colors;right: result of the LIC process on a vector field with a singularity of index -2.

The time step Δs is defined such that $n \cdot \Delta s = L$, where n is the number of steps and $L = 1/10$, in the both directions. We use only the direction of the vector as done in [3] and we found that the results obtained are satisfactory. As for the convolution kernel, we chose to use a rather simple low-pass filter, which is of value $1/2L$ in $[-L, L]$ and zero elsewhere. Namely, the average of the $2n+1$ sample of the T_{noise} along the streamline. We chose this box filter basically for the reason of performance. To implement more complicated kernels such as the Hanning filter as done in [3], one needs more precomputation and extra storage (a one-dimensional texture as discussed in [9]).

The integral in a direction (positive or negative) stops at q_i whenever the position q_i gives a positive result to Equation 16 by resampling the depth information, T_{depth}, stored in the second pass. Random noise value is assigned to the unsampled points on the rest of the streamline. This ensures visual discontinuities at geometric discontinuities.

6 Results, Discussions and Conclusions

We have shown examples of vector fields that are represented by our construction on several types of surface. All the examples are visualized by the LIC-based algorithm that we presented, which is implemented on a machine with a 1.7 GHz Pentium processor and 1 GB of RAM equipped with a GeForce FX5650 graphics card. We achieved 15 $frames/s$ to visualize the Statue (50k facets) at resolution 800×600.

In every example that we have shown, using our representation, singularities can only appear at vertices. Figure 1 shows the comparison of our representation and the *classic* piecewise linear vector field. Note that our interpolation scheme is C^0 smooth except at vertices. It is not C^1, but our choices lead to continuity of the derivative across edges in direction perpendicular to the crossed edge.

We also show examples of vector fields defined on surfaces of various genuses. Figures 1, 2 and 9 show genus-0 surfaces, Figure 8 shows a genus-2 surface and Figure 10 shows a genus-3 surface with more geometric detail. Amongst the examples, Figure 2 is especially interesting. One can see in this figure that curvilinear sectors in the vicinity of a higher-order singularity correspond to the three basic sector types. Moreover, it shows that the complexity of the singularities that a vertex can represent is not limited by its connectivity. Finally, the index that we have defined in Equation 14 satisfies the Poincaré-

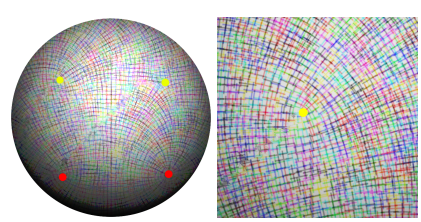

Fig. 9. One can have singularities with indices of multiples of 1/4. If the period jumps used are of multiples of 1/4. Left: indices of the singularities from left to right and from top to bottom are 1/4, -1/4, 1/2 and 1. Right: a zoom to the singularity of index 1/4.

Hopf index theorem as we have proved in [17]. One can check for instance in Figure 2 that the indices of the two singularities of the sphere sum to $5-3=2=\chi(g=0)$, where g is the genus and χ is the Euler characteristic of the surface.

Our representation also enables singularities with fractional indices to be represented. This is achieved by using period jumps multiple of 1/2 or 1/4 instead of integers. Examples are shown in Figure 2, 1-bottom and 9 for the three cases respectively.

In terms of memory consumption, vertex-based vector field encoding schemes on surfaces require three real numbers (4 bytes) to be stored per vertex. Our facet-based encoding requires two real numbers to be stored per facet corresponding to the angle and norm, and one integer (1 byte) per edge corresponding to the period jump. If we denote V the number of vertices, and considering that the number of facets (resp. edges) of a mesh is around 2 (resp. 3) times the number of vertices, our approach requires around $(2\times4\times2+2\times3)V=22V$ bytes to be stored while it needs around $(3\times4)V=12V$ bytes in the classic vertex-based encoding scheme. This difference is explained by the fact that our sampling is on facet barycenters, which are around twice as dense as vertices.

Limitations

We have adopted the side-vertex interpolation scheme not only for efficiency and simplicity considerations, but also for a nice visualization of singularities of arbitrary index. The drawback of this choice is that field discontinuities may appear at non-singular vertices. However, the visual quality is affected only when one zooms very closely to a vertex or on a coarse mesh where a vector field with high variations is defined. In such cases, one can use other types of interpolation at non-singular vertices to make the field smoother. This will be one of the directions in the future work.

With our representation, singularities can only occur at vertices, whereas in piecewise linear vector fields, first-order singularities may appear anywhere on facets. This is more an advantage than a flaw since it gives direct control over the singularity placement. Moreover, we can allow a singularity anywhere on the mesh by simply creating a new vertex at the desired singularity position and splitting the corresponding facet in three.

Conclusions and Future Work

In this paper, we have introduced a concise yet complete representation of vector fields on triangulated surfaces with arbitrary topology. The representation enables the user to capture or represent higher-order singularities in a given continuous vector field, for example using a FEM function basis. The vector field is described in polar coordinates on each facet, with a facet edge being chosen as the reference to define the angle. We have introduced an integer called the period jump, which is associated to each edge of the triangulation to remove the ambiguity in interpolating the angle between two facets sharing an edge. The field is interpolated over the whole surface by using a variant of side-vertex interpolation, which combined with the notion of the period jump removes the bound on the complexity of singularities that a vertex can represent imposed by its connectivity. Since the singularities can only be located at vertices and their indices can be found combinatorily, tracking of singularities over time may become easier as compared to traditional methods that use only numerical detection. To visualize the so-defined vector field at interactive frame rate, we have presented a LIC-based GPU-accelerated algorithm operating in image space.

There are several directions for the future work. Most naturally, the representation may be extended to represent 3D vector fields with higher-order singularities. Besides, by adopting our representation, one can carry out visualization of dynamic flow on surfaces. Finally, as discussed in the previous section, a different interpolation scheme can be studied for non-singular vertices.

Fig. 10. A vector field defined using our representation on a genus-3 surface. Yellow dots are first-order singularities with indices -1 and 1. Red dots are higher-order singularities of indices both -2.

Acknowledgements

The authors are greatly indebted to the reviewers for their excellent feedback. Special thanks to Eric Kow for the proof-reading of the manuscript.

References

[1] A. Andronov, E. Leontovich, I. Gordon, and A. Maier. Qualitative theory of second-order dynamic systems. *Israel Program For Scientific Translation*, 1973.

[2] H. Battke, D. Stalling, and H. Hege. Fast line integral convolution for arbitrary surfaces. In *Visualization and Mathematics*, pages 181–195. Springer-Verlag, Heidelberg, 1997.

[3] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. In *Proceedings of ACM SIGGRAPH*, pages 263–272, 1993.

[4] N. A. Carr and J. C. Hart. Meshed atlases for real-time procedural solid texturing. *ACM Transactions on Graphics*, 21(2):106–131, April 2002.

[5] P. Firby and C. Gardiner. *Surface Topology*, chapter 7, pages 115–135. Ellis Horwood Ltd., 1982.

[6] M. S. Floater and K. Hormann. *Surface Parameterization: a Tutorial and Survey*, pages 157–186. N. A. Dodgson, M. S. Floater, and M. A. Sabin (eds.), Springer-Verlag, Heidelberg, 2004.

[7] L. Forssell. Visualizing flow over curvilinear grid surfaces using line integral convolution. In *Proceedings of IEEE Visualization*, pages 240–247, 1994.

[8] L. Forssell and S. Cohen. Using line integral convolution for flow visualization: Curvilinear grids, variable-speed animation, and unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):133–141, June 1995.

[9] M. Grabner and R. S. Laramee. Image space advection on graphics hardware. In *Proceedings of the 21st Spring Conference on Computer Graphics and its Applications*, pages 75–82, 2005.

[10] W. Heidrich, R. Westermann, H. Seidel, and T. Ertl. Applications of pixel textures in visualization and realistic image synthesis. In *Proceedings of Symposium on Interactive 3D Graphics*, pages 127–134, 1999.

[11] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In *Proceedings of ACM SIGGRAPH*, pages 517–526, 2000.

[12] V. Interrante. Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution. In *Proceedings of ACM SIGGRAPH*, pages 109–116, 1997.

[13] R. S. Laramee, J. Schneider, and H. Hauser. Image space based visualization of unsteady flow on surfaces. In *Proceedings of IEEE Visualization*, pages 131–138, 2003.

[14] R. S. Laramee, J. van Wijk, B. Jobard, and H.Hauser. ISA and IBFVS: Image space-based visualization of flow on surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):637–648, 2004.

[15] G. Nielson. The side-vertex method for interpolation in triangles. *Journal of Approximation Theory*, 25(4):318–336, 1979.

[16] N. Ray, W. Li, B. Levy, A. Sheffer, and P. Alliez. Periodic global parameterization. *ACM Transactions on Graphics*, to appear.

[17] N. Ray, B. Vallet, W. Li, and B. Levy. N-symmetry direction fields on surfaces of arbitrary genus. Technical report, INRIA-ALICE, January 2006.

[18] C. Rezk-Salama, P. Hastreiter, T. Christian, and T. Ertl. Interactive exploration of volume line integral convolution based on 3D-texture mapping. In *Proceedings of IEEE Visualization*, pages 233–240, 1999.

[19] G. Scheuermann, H. Kruger, M. Menzel, and A. Rockwood. Visualizing non-linear vector field topology. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):109–116, 1998.

[20] G. Scheuermann, X. Tricoche, and H. Hagen. C1-interpolation for vector field topology interpolation. In *Proceedings of IEEE Visualization*, pages 271–278, 1999.

[21] W. Schroeder, F. Bertel, M. Malaterre, D. Thompson, and P. Pebay. Framework for visualizing higher-order basis functions. In *Proceedings of IEEE Visualization*, pages 43–50, 2005.

[22] D. Stalling and H. Hege. Fast and resolution independent line integral convolution. In *Proceedings of ACM SIGGRAPH*, pages 249–256, 1995.

[23] H. Theisel. Designing 2D vector fields of arbitrary topology. *Computer Graphics Forum (Proceedings Eurographics 2002)*, 21(3):595–604, 2002.

[24] X. Tricoche. *Vector and Tensor Topology Simplification, Tracking, and Visualization*. PhD thesis, Schriftenreihe Fachbereich Informatik (3), University of Kaiserslautern, 2002.

[25] X. Tricoche, G. Scheuermann, and H. Hagen. Higher order singularities in piecewise linear vector fields. In *The Mathematics of Surfaces IX, Springer, London*, pages 99–113, 2000.

[26] J. van Wijk. Image based flow visualization for curved surfaces. In *Proceedings of IEEE Visualization*, pages 123–130, 2003.

[27] L.-Y. Wei and M. Levoy. Texture synthesis over arbitrary manifold surfaces. In *Proceedings of ACM SIGGRAPH*, pages 355–360, 2001.

[28] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Topological construction and visualization of higher order 3d vector fields. *Computer Graphics Forum (Proceedings Eurographics 2004)*, 23(3):469–478, 2004.

[29] T. Weinkauf, H. Theisel, K. Shi, H.-C. Hege, and H.-P. Seidel. Extracting higher order critical points and topological simplification of 3d vector fields. In *Proceedings of IEEE Visualization*, pages 559–566, 2005.

[30] H. H. X. Tricoche, G. Scheuermann. A topology simplification method for 2D vector fields. In *Proceedings of IEEE Visualization*, pages 359–366, 2000.

[31] E. Zhang, J. Hays, and G. Turk. Interactive design and visualization of tensor fields on surfaces. Technical report, Oregon State University, 2005.

[32] E. Zhang, K. Mischaikow, and G. Turk. Vector field design on surfaces. Technical report, GVU 04-16, Georgia Institute of Technology, 2004. Accepted by ACM Transactions on Graphics, pending revision.