A Robust Parity Test for Extracting Parallel Vectors in 3D

Tao Ju, Minxin Cheng, Xu Wang, and Ye Duan, *Member, IEEE*

Abstract—Parallel vectors (PV), the loci where two vector fields are parallel, are commonly used to represent curvilinear features in 3D for data visualization. Methods for extracting PV usually operate on a 3D grid and start with detecting seed points on a cell face. We propose, to the best of our knowledge, the first provably correct test that determines the parity of the number of PV points on a cell face. The test only needs to sample along the face boundary and works for any choice of the two vector fields. A discretization of the test is described, validated, and compared with existing tests that are also based on boundary sampling. The test can guide PV-extraction algorithms to ensure closed curves wherever the input fields are continuous, which we exemplify in extracting ridges and valleys of scalar functions.

Index Terms—Parallel vectors, feature curve extraction, ridges and valleys, parity test

1 Introduction

The parallel vector (PV) operator is a general-purpose line feature representation proposed by Peikert and Roth [8] for data visualization. It is defined as the loci where two vector fields u, w are parallel or one of the two vectors is zero. In 3D, PV lines consist of those points x that satisfy

$$u(x) \times w(x) = 0 \tag{1}$$

Various types of line features can be expressed as PV with suitable choices of u, w, including ridges and valleys of a scalar field, streamlines of a vector field, vortices of a velocity field, and extremal curves of a tensor field.

As surveyed in [8], PV is usually computed on a spatial grid and proceeds in two stages. First, possible locations of PV on a grid cell face are computed. To differentiate them from true PV locations, we shall refer to these computed locations as *seeds*. Next, seeds are connected within a grid cell, either simply by straight line segments [16, 2, 11] or using more sophisticated but more accurate curve tracing methods [15, 13, 1, 7].

A key property that makes PV appealing is that they are generally made up of closed curves when u, w are continuous [8]. This is in contrast with discontinuous features such as local maxima or minima in a scalar field. To inherit this property in a computational algorithm, the total number of seeds identified by the algorithm over all faces of a cell should be *even*. Otherwise, any way of connecting the seeds into curves within the cell would create either an open curve or a junction with odd degrees, which cannot be separated into disjoint closed segments. Various methods have been proposed for computing seeds (see a review in the next section). However, none of them comes with a theoretical guarantee of the parity of the number of seeds either on a cell face or in a cell, except in the special case when u, w are linear over each face.

In this paper, we propose a provably correct test that, given *any* choice of continuous fields u, w over a cell face, determines the parity of the number of PV points on that face. The test mimics the classical zero-crossing test over an 1D interval based on signs at the ends of the interval: it reports "true" if and only if there are odd number of true zeros in the domain. Like zero-crossing, our test only needs to sample u, w on the boundary of the face, and does not require sampling or subdivision of the face interior. The idea behind the test is to consider the solution of Equation 1 as singularities in a tangent vector field over some auxiliary surface. We show that the parity of the number of stable singularities in this field can be computed using the classical Gauss-Bonnett and Poincare-Hopf theorems *without* explicitly constructing the auxiliary surface. A discrete implementation of the test is presented and validated using a synthetic input whose PV has a known and analytical form.

Using our test, a PV extraction algorithm can ensure closed curves wherever u, w are continuous by creating one seed on each cell face that our test returns odd. We exemplify such usage in a typical 2-stage PV algorithm applied to extract ridge and valley lines of a scalar field.

- *Tao Ju is at Washington University in St. Louis. E-mail: taoju@cse.wustl.edu.*
- *Minxin Cheng and Xu Wang are at University of Missouri at Columbia. E-mails: [mcdz2|xwf32]@mail.missouri.edu.*
- *Ye Duan (corresponding author) is at University of Missouri at Columbia. E-mail: duanye@missouri.edu.*

Manuscript received 31 Mar. 2014; accepted 1 Aug. 2014. Date of publication 11 Aug. 2014; date of current version 9 Nov. 2014. For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.
Digital Object Identifier 10.1109/TVCG.2014.2346412

2 Related Works

Given the scope of our work, we limit our discussion to published methods for identifying seeds. In particular, we will focus on the combinatorial aspect (e.g., number and parity), rather than the geometric location, of the seeds computed by these methods.

2.1 Analytical solution

When the face is a triangle and both fields u, w are defined by linear interpolation of vectors at the triangle vertices, the exact number and locations of PV points can be found analytically as an eigenvector problem [8]. However, this approach is not applicable for high-order, non-linear fields.

2.2 Iterative root-finding

For non-linear fields u, w, solutions to Equation 1 can be sought by using root-finding techniques over the cell face [11, 15, 13, 7]. These methods typically start at an initial location on the face, such as its center, and uses 2D Newton-Raphson technique either for a fixed number of iterations or until convergence. A seed is found if the iteration terminates at a PV point (within some tolerance) inside the face. However, root-finding does not guarantee to find any or all PV points on the cell face. There is also no guarantee on the parity of total number of seeds over all faces of a cell.

Pagot et al. [7] uses reduced affine arithmetic to conservatively estimate the presence of PV points in a cell and on a face. The test is used to recursively subdivide any cell (and any face) that might contain a PV point. While the method can reduce the chances of missing PV points, there is still no guarantee on the correctness of the total number or parity of the computed seeds. Also, recursive subdivision adds computational cost, which can be substantial since the test based on affine arithmetic is conservative. Finally, the technique is restricted to those u, w with known, polynomial forms.

2.3 Boundary sampling

A class of methods detect the presence of PV points over a cell face by examining u, w on the face boundary [2, 14, 5, 10]. The rationale

comes from the well-known fact that the parity of the number of first-order critical points in a 2D vector field z in a closed region can be obtained by the *winding number* (total number of 2π turns) of z as it travels along the region boundary. These methods typically start by projecting the 3D vectors whose zeros are being sought on the face boundary onto a common plane, and then analyze the winding of the projected 2D vector field.[1] The 3D vectors can be either $u \times w$ or $(u \times w) \times u$, the projection of w onto plane orthogonal to u. The projection plane could be the supporting plane of the face [2, 5] or the plane orthogonal to the averaged u vectors along the face boundary [14, 10].

An appealing property of this approach is that it does not need to sample the interior of the face. However, current tests only capture the zeros in the projected 2D vector field rather than those in the original 3D vector field v. As we will demonstrate in Section 4, the winding of the projected 2D field can be a poor indicator for the parity of zeros of v. In addition, since the choice of the projection plane in current methods varies for different faces of the cell, there is no guarantee on the correct parity of the entire set of seeds in a cell.

Similar to these methods, our method also samples the 3D vector v along the face boundary and examines its winding. The key difference is that we compute the winding on varying planes that are orthogonal to u at each boundary location. Since these planes always contain v, we avoid the inaccuracy associated with projection.

3 Background

As we shall see in the next section, we will formulate the problem of detecting PV points as identifying critical points of a tangent vector field over a surface. Our method builds upon classical results on differential geometry and vector field topology, which we shall briefly review first. We refer interested readers to introductory materials such as [3] for in-depth coverage of these concepts. Below we assume S is a differentiable, genus-zero surface in R^3 with a smooth boundary ∂S.

3.1 Vector fields along curves

We start by considering vectors that are tangent to the surface S and move continuously along a differentiable curve on the surface $\tau : [0,1] \rightarrow S$. An example is the (unit) tangent vector field along τ, noted as g_τ (shown in blue in Figure 1 (a)).

An operator used frequently for vector field analysis is how much one vector field v_1 *turns* with respect to another field v_2 along the curve. To formalize turning, note that the vector $v_1(t)$ and the surface normal defines a coordinate frame in the tangent plane at $\tau(t)$. The turning from v_1 to v_2 along τ on the surface S, denoted as $\mathrm{Turn}_{\tau,S}(v_1, v_2)$, is the total counter-clockwise angle traced by v_2 in this coordinate frame. Figure 1 illustrates the turning from a vector field v (red) to the tangent vector field g_τ (blue).

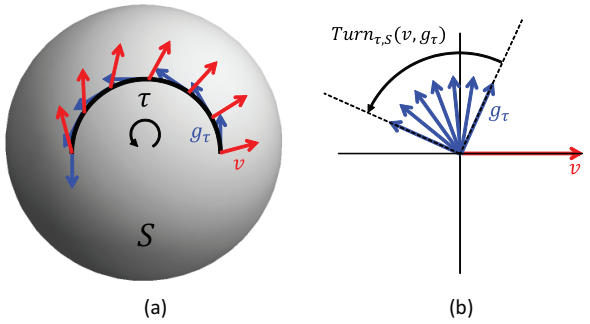

Fig. 1. (a) The tangent vector field g_τ (blue) and another vector field v (red) along a curve τ on a unit sphere S. (b) Plotting g_τ in the coordinate frame defined by v and the turn from v to g_τ.

[1] The approach used by Medioni and co-workers [2, 14], which is based on signs of the coordinate components of the projected v only at the four corners of the face, can be considered as a discrete way of computing the winding number.

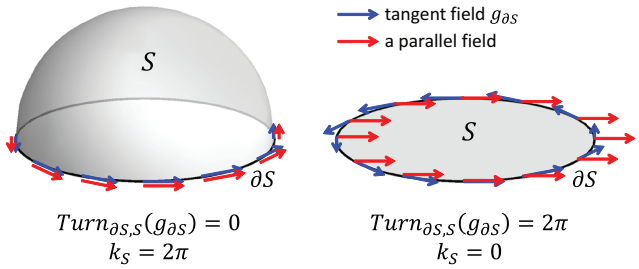

Fig. 2. Comparing the tangent fields (blue) and parallel fields (red) along the boundary of a hemisphere (left) and a flat disk (right). Noted under each picture are the total geodesic curvature of the boundary curve (defined as the turning from the parallel field to the tangent field) and the total Gaussian curvature of the surface.

3.2 Parallel transport and geodesic curvature

A vector field is *parallel* along τ if its derivative has zero component in the tangent plane along τ. The tangent field g_τ is also parallel only when τ is a geodesic on S. Unlike the tangent field, there are infinitely many parallel fields on a given curve. In fact, there exists a parallel field starting from any given vector at one end of the curve $\tau(0)$. This is known as the *parallel transport* of that starting vector.

An important property of parallel transport is that it only depends on the surface normal along τ and is independent of the geometry of τ. That is, parallel transport of a same initial vector along two curves τ_1, τ_2 will produce the same sequence of vectors if $\tau_1(t)$ has the same normal direction as $\tau_2(t)$ for all $t \in [0,1]$.

Figure 2 compares tangent fields and parallel fields on two examples. When τ is the equator of a sphere (left), the tangent field is also parallel, since τ is a geodesic. When τ is a circle on a plane (right), the tangent field turns a full circle around τ while a parallel field stays constant.

The turning from one parallel field to another is always zero. The turning from any parallel field to some fixed vector field v is the same, which we simply denote as $\mathrm{Turn}_{\tau,S}(v)$. The *total geodesic curvature* of τ is defined as the turning from a parallel field to the tangent field, or $\mathrm{Turn}_{\tau,S}(g_\tau)$. This quantity reflects how much the curve τ twists on the surface S. The total geodesic curvature of a geodesic curve is zero, since the tangent field is also parallel. In the examples of Figure 2, the total geodesic curvature of the sphere equator and the planar circle is respectively 0 and 2π.

3.3 Gaussian curvature

The geodesic curvature of the boundary curve is closely related to the curvature over the interior of the surface. Consider the Gauss map, which maps a point $x \in S$ to a point $n(x)$ on the unit sphere B such that $n(x)$ is the unit outward normal of S at x. The *total Gaussian curvature* of S, denoted by k_S, is the signed area of the image of S on B under the Gauss map. The sign of the area at a location x is determined by the Jacobian determinant of the Gauss map at x.

Gauss-Bonnet theorem states that the sum of total Gaussian curvature of S (that is, k_S) and the total geodesic curvature of the boundary ∂S (that is, $\mathrm{Turn}_{\partial S,S}(g_{\partial S})$) is 2π times the Euler characteristic of S. Since S being considered here has an Euler characteristic of 1, we have

$$k_S = 2\pi - \mathrm{Turn}_{\partial S,S}(g_{\partial S}) \tag{2}$$

We use the convention that ∂S is oriented such that S is on the "left" of the curve, or more formally, the cross-product of the normal of S and $g_{\partial S}$ always points towards the interior of S. In the examples of Figure 2, the total Gaussian curvature is respectively 2π and 0 for the hemisphere and the flat disk.

3.4 Vector fields on a surface

Consider a continuous vector field v over S that is tangential to S. We assume the generic situation where zeros of v are isolated and away

from ∂S. These critical points can be characterized using the Poincare index. Given any point $x \in S$, consider a disk neighborhood D of x small enough to not contain any other critical points of v except x. Given a continuous tangent coordinate frame over D, the *index* at x is the total counter-clockwise angle swept by v as it travels along ∂D in the counter-clockwise orientation divided by 2π (also known as the winding number of v).

The index is zero if x is a regular point, and can assume positive or negative integers if x is a critical point. While a critical point can have any integer index, only ones with index +1 (sources or sinks or foci) and -1 (saddles) are stable with respect to small perturbation in the field [12]. Higher-order critical points can be decomposed into lower-order critical points under slight perturbation.

If v is orthogonal to the boundary ∂S and points towards the exterior of S, the Poincare-Hopf theorem states that the sum of all critical point indices, denoted as $\mathrm{Ind}_S(v)$, is exactly the Euler characteristic of S, which is 1. Morse [6] generalized the theorem to an arbitrary vector field v by considering the turning of v from the tangent vectors along ∂S,

$$\mathrm{Ind}_S(v) = 1 + \frac{\mathrm{Turn}_{\partial S,S}(g_{\partial S}, v)}{2\pi} \tag{3}$$

As an example, consider the three vector fields in Figure 3 on a flat disk S. The three fields exhibit turnings of 0, -2π, and -4π from the boundary tangent, correctly calculating the total index of 1 (a source), 0 (no critical points), and -1 (a saddle) inside S.

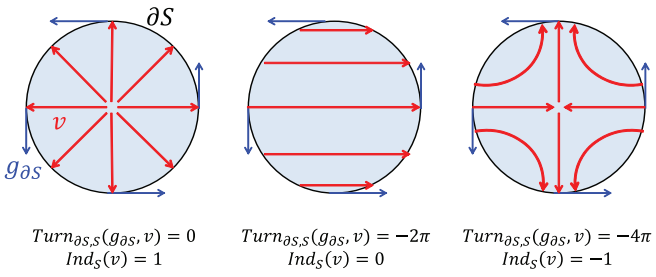

Fig. 3. Examples demonstrating the generalized Poincare-Hopf theorem, relating the total index $\mathrm{Ind}_S(v)$ of a vector field v over a disk surface S to the turning of v from the tangents $g_{\partial S}$ along the boundary ∂S.

4 The Parity Test

Our key idea is to treat the solution of Equation 1 as critical points in a tangent vector field over some auxiliary surface. The total index of the critical points, which has the same *parity* as the number of stable critical points, can then be obtained by observing the boundary behavior of u, w. We start with a theoretical derivation of our test in the continuous setting, and then present a discrete implementation. We end with validation of the test and comparison to previous methods.

4.1 Theory

Suppose we have a surface M in R^3 that is homeomorphic to a disk (e.g., a cell face) and two continuous vector fields u, w on M. We make several assumptions of generic inputs. First, at least one of the fields, say u, is non-zero on M. Second, the PV consists of 1-dimensional curves, and these curves intersect with only the interior of M at a finite set of points. Note that these generic assumptions may be violated in the discrete implementation, which we shall address in the next section.

4.1.1 Index of a PV point

We start by defining the index of a PV point on M. Each PV point is a zero of the cross-product field $v = u \times w$. Equivalently, it is also the zero of the field $v = (u \times w) \times u$, which has a geometric interpretation as the projection of w onto the plane orthogonal to u. These two choices of v are closely related; in fact they differ by a rotation of $\pi/2$ around u. We use the latter definition, which is adopted in existing tests based on boundary sampling [2, 14], but our results apply to either definition.

The index of a PV point is defined similarly as the Poincare index of a critical point on a surface. Given any point $x \in M$, consider a disk neighborhood $D \subset M$ around x small enough to not contain any other zeros of v except x. We establish a continuous coordinate frame on each orthogonal plane to $u(y)$ for all $y \in D$. This is possible because u is continuous and non-vanishing. The *PV index* at x is the winding number of $v(y)$ in this frame as y travels in counter-clockwise orientation along ∂D. The notations are explained in Figure 4.

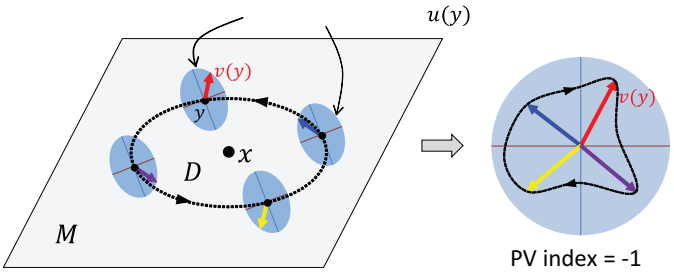

Fig. 4. Notations for defining the PV index.

Similar to the Poincare index, the PV index is zero if $v(x)$ is non-zero, and can assume positive or negative integers otherwise. By the same argument in [12], only PV points with indices +1 and -1 are stable with respect to small perturbation in the field, whereas other PV points can be decomposed into these stable ones.

Since a stable PV point has an odd index, the total index of all PV points on M has the same *parity* as the total number of stable PV points. In the following, we shall derive a formula for the total PV index on M.

4.1.2 Total index

To explain our formula intuitively, we will assume for now that there exists a mapping f from M to some auxiliary surface S such that, for every point $x \in M$, the outward unit normal at the mapped point $f(x) \in S$ is the unit vector $\hat{u}(x) = u(x)/|u(x)|$ (see Figure 5). As such, v becomes a tangent field on S, and the PV index at a point $x \in M$ is the same as the Poincare index of v at $f(x) \in S$. Now our task becomes computing the total index of v on S.

Since we do not know the geometry of ∂S, we cannot directly apply the generalized Poincare-Hopf theorem (Equation 3) to compute the total index of v. We need a different formula that would allow us to compute this index with only the knowledge of the normal field of S, which is u. To do so, we make note of the following property of turning,

$$\mathrm{Turn}_{\tau,S}(v_1, v_2) = \mathrm{Turn}_{\tau,S}(v_2) - \mathrm{Turn}_{\tau,S}(v_1) \tag{4}$$

That is, the turning from one vector field v_1 to another field v_2 along a curve τ is the difference between the turning from the parallel field to v_2 and the turning from the parallel field to v_1. Substituting this identity and the Gauss-Bonnet theorem (Equation 2) into the Poincare-Hopf theorem (Equation 3) yields

$$\mathrm{Ind}_S(v) = \frac{\mathrm{Turn}_{\partial S,S}(v) + k_S}{2\pi} \tag{5}$$

Note that the quantities on the right-hand side of this equation can be obtained by the Gauss map from S to the unit sphere B. The total Gaussian curvature, k_S, is the signed area of the mapped region on B. The turning of v from the parallel field along the boundary curve ∂S is the same as the turning of v from the parallel field along the mapped curve ∂S on B, because the two curves share the same normal field.

On the other hand, this Gauss map can be directly defined from M and u, without the need to construct the auxiliary surface S. In fact,

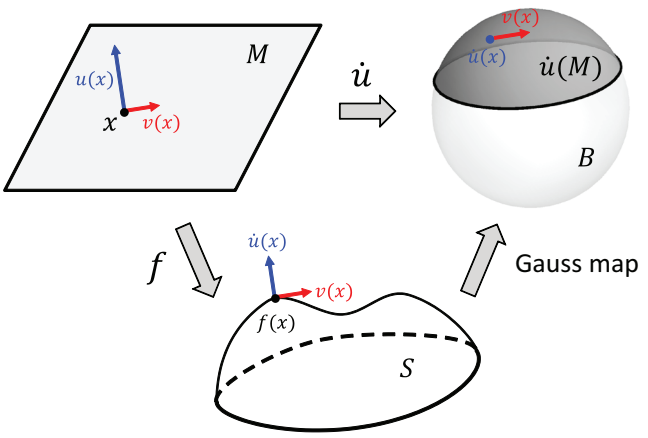

Fig. 5. Illustration of the mapping f from M to S, and the Gauss map from S to $\dot{u}(M)$ on the unit sphere B.

the image of a point $x \in M$ after first mapping to S (via f) and then to the unit sphere (via Gauss map) is exactly $\dot{u}(x)$. We can now write the image of S under the Gauss map as $\dot{u}(M)$, which is bounded by curve $\dot{u}(\partial M)$ (see Figure 5). Denoting the signed area of $\dot{u}(M)$ as $A_{\dot{u}(M)}$, Equation 5 translates to a practical formula for total PV index:

Lemma 4.1 *The sum of PV indices on M equals*

$$\frac{Turn_{\dot{u}(\partial M),B}(v) + A_{\dot{u}(M)}}{2\pi} \tag{6}$$

In words, the total PV index equals the sum of signed area covered by the spherical region $\dot{u}(M)$ and the turning of v along this region's boundary from a parallel vector field divided by 2π. While our derivation above assumes the existence of the auxiliary surface S, we can nevertheless prove without this assumption (see Appendix A).

4.1.3 Parity of PV points

Lemma 4.1 requires knowledge of u over the interior of M to compute the signed area $A_{\dot{u}(M)}$. To avoid such need, we make the following observation (see proof in Appendix B):

Lemma 4.2 *Let z be any continuous, non-zero vector field on M such that $z(x) = u(x)$ for all $x \in \partial M$, and let $\dot{z}(x) = z(x)/|z(x)|$. Then*

$$A_{\dot{u}(M)} = A_{\dot{z}(M)} \ (mod\ 4\pi) \tag{7}$$

We illustrate this result using an example in Figure 6. Here M is a unit disk on the XY plane centered at the origin. For a point on the disk with coordinates $x = \{a, b, 0\}$, define $u(x) = \left\{a, b, \sqrt{1-a^2-b^2}\right\}$ (top-left) and $z(x) = \left\{a, b, -\sqrt{1-a^2-b^2}\right\}$ (bottom-left). Note that the two vector fields are identical on the boundary of the disk. While $\dot{u}$ maps M to the top hemisphere of the unit sphere with a positive Jacobian determinant, $\dot{z}$ maps M to the bottom hemisphere with a negative Jacobian determinant (as revealed by the inverted color order at the pole). The signed areas of the two images, $\dot{u}(M)$ and $\dot{z}(M)$, are respectively 2π and -2π. Their difference is 4π, which agrees with Lemma 4.2.

The benefit of Lemma 4.2 is that we can replace $A_{\dot{u}(M)}$ in Equation 6 by $A_{\dot{z}(M)}$ without affecting the parity of the quotient. Since the only requirement of z is to match u on the boundary ∂M, such replacement avoids the need to probe the interior of M. Recall that the total PV index over M has the same parity as the number of stable PV points, we arrive at our main result,

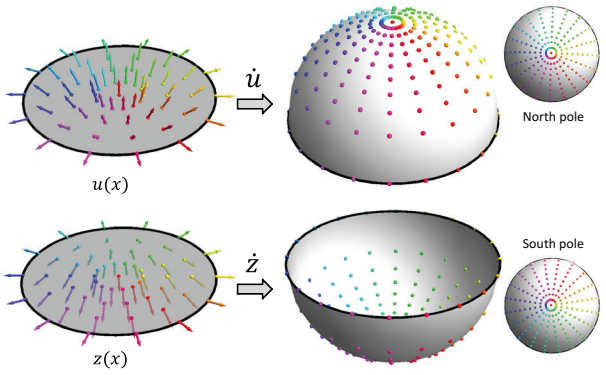

Fig. 6. An example illustrating Lemma 4.2.

Theorem 4.3 *The number of stable PV points on M equals, modulo 2,*

$$\frac{Turn_{\dot{u}(\partial M),B}(v) + A_{\dot{z}(M)}}{2\pi} \tag{8}$$

where z is any continuous, non-zero vector field on M such that $z(x) = u(x)$ for all $x \in \partial M$, and $\dot{z}(x) = z(x)/|z(x)|$.

We wish to choose a field z whose resulting signed area $A_{\dot{z}(M)}$ can be easily evaluated. In this work, we use linear interpolation in a radial parameterization of M. Define some mapping from the unit disk to M so that the origin of the disk maps to some location $c \in M$. Let $z(c) = z_0$ where z_0 is an arbitrary unit vector. Each ray of the unit disk is mapped to a curve on M that starts from c and ends on some boundary point $x \in \partial M$. For each point y on this ray, define the vector $z(y)$ by interpolating $z(c)$ and $u(x)$ along the ray. By this definition, the image $\dot{z}(M)$ on the unit sphere is made up of great arcs from z_0 to unit vectors $\dot{u}(x)$ for all $x \in \partial M$. As we shall see next, the signed area of $\dot{z}(M)$ can be computed discretely over the spherical curve $\dot{u}(\partial M)$.

4.2 Discrete computation

The parity test in Equation 8 can be easily discretized and computed using basic spherical geometry. We assume that the surface boundary ∂M is sampled by a finite sequence of points, $\{p_0, \ldots, p_n\}$ such that $p_0 = p_n$. We use $u_i, v_i, \dot{u}_i$ to denote the vectors $u, v = (u \times w) \times u, \dot{u} = u/|u|$ at a sampled location p_i.

To discretize Equation 8, we approximate the spherical curve $\dot{u}(\partial M)$ by a piecewise smooth curve made up of great arcs of B connecting successive samples in the sequence $\{\dot{u}_0, \ldots, \dot{u}_n\}$. We next give details on computing the two quantities, $Turn_{\dot{u}(\partial M),B}(v)$ and $A_{\dot{z}(M)}$. Note that our computation gives the *exact* result on this piecewise spherical curve, and hence the sum of the two quantities will still be an exact multiple of 2π (subject to numerical imprecisions). We end this section with a discussion on choosing the sample points.

4.2.1 Discretizing turning

The first quantity, $Turn_{\dot{u}(\partial M),B}(v)$, can be computed as the sum of turning over each arc in the piecewise approximation of $\dot{u}(\partial M)$.

To compute the turning on the arc between $\{\dot{u}_i, \dot{u}_{i+1}\}$, we first parallel-transport vector v_i along the arc from $\dot{u}_i$ to $\dot{u}_{i+1}$, producing a new vector v_i^* at $\dot{u}_{i+1}$ (see Figure 7 left). There are several ways to compute v_i^*. One can rotate v_i around the axis orthogonal to $\dot{u}_i, \dot{u}_{i+1}$ by the angle between $\dot{u}_i$ and $\dot{u}_{i+1}$. However, the computation of the rotation axis requires a cross-product, which can be numerically unstable when the angle between $\dot{u}_i$ and $\dot{u}_{i+1}$ is small. We take the more stable approach of Wang et al. [17], which involves only dot products.

The formulation given in [17] is used to transport normal vectors along a spatial curve. We first re-formulate our problem as normal transportation as follows. Let g_0, g_1 be the unit tangent vector of the great arc from $\dot{u}_i$ to $\dot{u}_{i+1}$ at the two ends of the arc, and consider now the great arc on the sphere from g_0 to g_1. Note that $\dot{u}_i$ and $\dot{u}_{i+1}$ are

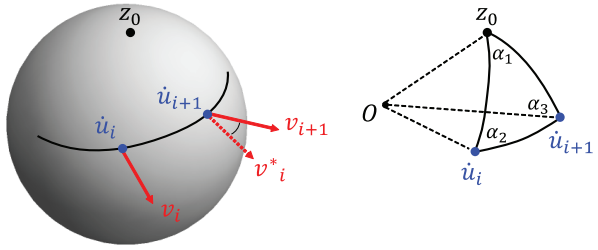

Fig. 7. Notations for discretizing the parity test.

tangent vectors of this arc, and v_i is a normal vector of the arc at g_0. Our goal is to transport v_i to another normal vector v_i^* at the other end of the arc, g_1.

Following [17], v_i^* is obtained by two reflection transformations. The first reflection uses the bisecting plane between g_0, g_1 (Figure 8 (a)). Observe that the unit normal of this plane can be obtained by $n = (\dot{u}_i + \dot{u}_{i+1}) / |(\dot{u}_i + \dot{u}_{i+1})|$. The reflection of any vector ξ by this plane can be written as

$$R(\xi) = \xi - 2(\xi \cdot n) n \tag{9}$$

After reflection, $R(v_i)$ remains orthogonal to $R(\dot{u}_i)$, and the latter is identical with $-\dot{u}_{i+1}$. The second reflection uses the bisecting plane between $R(\dot{u}_i)$ and $\dot{u}_{i+1}$ (see Figure 8 (b)). Since $R(v_i)$ is parallel to this reflection plane, it is unchanged by the reflection. Hence we obtain $v_i^* = R(v_i)$.

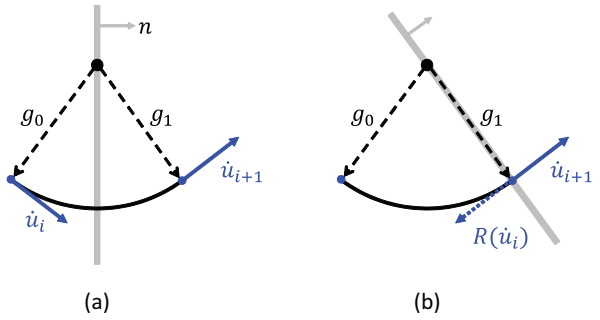

Fig. 8. Parallel transport from $\dot{u}_i$ to $\dot{u}_{i+1}$ by two reflections.

The turning on the arc between $\{\dot{u}_i, \dot{u}_{i+1}\}$ is simply the angle from v_i^* to v_{i+1} in the counter-clockwise orientation in the tangent plane at $\dot{u}_{i+1}$.

$$\angle(v_i^*, v_{i+1}) = \cos^{-1}\left(\frac{v_i^* \cdot v_{i+1}}{|v_i^*|\,|v_{i+1}|}\right) \delta_i \tag{10}$$

Here, δ_i is 1 (resp. -1) if $\dot{u}_{i+1} \cdot (v_i^* \times v_{i+1})$ is positive (resp. negative).

The summation of the angles over all arc segments gives the desired turning over the entire curve,

$$\text{Turn}_{\dot{u}(\partial M),B}(v) = \sum_{i=0}^{n-1} \angle(v_i^*, v_{i+1}) \tag{11}$$

4.2.2 Discretizing spherical area

Using the choice of z as discussed earlier, the region $\dot{z}(M)$, bounded by $\dot{u}(\partial M))$, is made up of n spherical triangles. Each triangle is formed by z_0, a randomly chosen unit vector, and unit vectors $\dot{u}_i, \dot{u}_{i+1}$ for $i = 0, \ldots, n-1$.

The signed area of each triangle can be obtained by

$$\triangle(z_0, \dot{u}_i, \dot{u}_{i+1}) = (\alpha_1 + \alpha_2 + \alpha_3 - \pi)\,\eta_i \tag{12}$$

where $\alpha_1, \alpha_2, \alpha_3$ are the dihedral angles between the planes forming the cone spanned by the spherical triangle and the origin (see Figure 7 right), and η_i captures the orientation of the triangle. Specifically, define n_k as the unit normal of the plane opposite to angle α_k for $k = 1, 2, 3$,

$$n_1 = \frac{\dot{u}_i \times \dot{u}_{i+1}}{|\dot{u}_i \times \dot{u}_{i+1}|}, n_2 = \frac{\dot{u}_{i+1} \times z_0}{|\dot{u}_{i+1} \times z_0|}, n_3 = \frac{z_0 \times \dot{u}_i}{|z_0 \times \dot{u}_i|} \tag{13}$$

The angles α_k are computed as

$$\alpha_1 = \cos^{-1}(n_2 \cdot n_3), \alpha_2 = \cos^{-1}(n_3 \cdot n_1), \alpha_3 = \cos^{-1}(n_1 \cdot n_2) \tag{14}$$

The sign η_i is 1 (resp. -1) if $\dot{u}_{i+1} \cdot (n_1 \times n_2)$ is positive (resp. negative). The summation of the signed areas over all triangles gives the desired total signed area over $\dot{z}(M)$,

$$A_{\dot{z}(M)} = \sum_{i=0}^{n-1} \triangle(z_0, \dot{u}_i, \dot{u}_{i+1}) \tag{15}$$

4.2.3 Boundary sampling

Our discrete algorithm correctly computes the parity of PV points given the discrete samples on ∂M. However, it may fail to capture the true parity on M if u or v exhibits excessive variation between the sampled locations. To keep the variations low, we use a simple adaptive sampling strategy. Starting from a face edge, we calculate the angular difference between the u (and v) vectors at the two ends and create a sample at the mid point of the edge if the difference is greater than a user-defined threshold (we used 10 degrees). The new sample divides the edge into two segments, and the same process is repeated for each segment until either no more samples need to be created or a maximum subdivision depth is reached (we used 10).

While we have not encountered such cases in our tests, it is possible that our generic assumptions (e.g., non-zero u and v on ∂M) may be violated at a sample location p due to numerical evaluations. To deal with such degeneracy, one can perturb the location of p by a small spatial amount and re-sample u, v there. As long as such perturbation is done consistently for all surfaces that use p as a boundary sample, the parity test can still guarantee closedness of the PV curves.

4.3 Comparison and Validation

Our parity test is similar in spirit to several existing methods for detecting zeros of v [2, 14, 5, 10] in that they also monitor the winding of v on the face boundary in some 2D coordinate system. The key difference is that these existing methods use a single plane to establishing the coordinate system. Since this plane almost never contains v, v has to be first projected onto this plane. In contrast, our method uses *variable* planes orthogonal to u, which always contain v, and hence no projection is needed. The 2D coordinate systems on these planes are established by parallel vectors along the spherical curve $\dot{u}(\partial M)$. Equation 6 essentially computes the winding of v in these coordinate frames ($\text{Turn}_{\dot{u}(\partial M),B}(v)$) corrected by the turning of the frames themselves ($A_{\dot{u}(M)}$).

To validate our method, we designed a synthetic example whose PV has a known form. The fields u, w are chosen so that the solution to Equation 1 can be expressed analytically but still has a non-trivial curve geometry. Specifically, for a spatial location $\{a, b, c\}$, we let w be a constant vector, and define u as a linear rotation field composed by successively rotating w around the X, Y, Z axes by angles a, b, c:

$$w(a,b,c) = \{1,1,1\}$$
$$u(a,b,c) = R_Z(c) \cdot R_Y(b) \cdot R_X(a) \cdot w(a,b,c)^T$$

where $R_\Pi(\alpha)$ is the 3D rotation matrix around a given axis Π by angle α (in radian). The solution to Equation 1 can be found using a symbolic package (e.g., *Mathematica*), and it is the union of straight lines and sinusoidal curves. For any real number t and integers c_1, c_2, there are 4 PV points of the following types:

- Type 1: $\left\{t, 2c_2\pi - \frac{\pi}{2}, -t + 2c_1\pi - \frac{\pi}{2}\right\}$

- Type 2: $\{t, 2c_2\pi + \frac{\pi}{2}, t + 2c_1\pi - \pi\}$
- Type 3: $\left\{t, 2\tan^{-1}\left(\frac{\cos(a)+\sin(a)+1}{\cos(a)+\sin(a)-1}\right) + 2c_2\pi, t + 2c_1\pi\right\}$
- Type 4: $\left\{t, 2 \bmod \left(\tan^{-1}\left(\frac{\cos(a)+\sin(a)-1}{-\cos(a)-\sin(a)-1}\right), 2\pi\right) + 2c_2\pi, -t + 2c_1\pi + \frac{\pi}{2}\right\}$

Points of the first two types lie on two groups of parallel lines, whereas points of the last two types lie on two groups of parallel sinusoidal curves. These PV curves are shown in Figure 9.

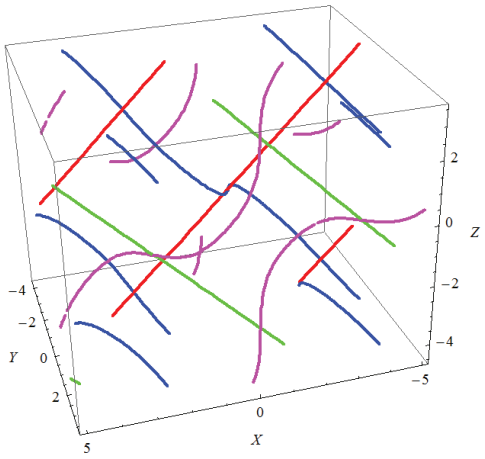

Fig. 9. PV curves of our synthetic vector fields. PV points of types 1, 2, 3, 4 are colored red, green, blue, magenta. The plot range is $[-5,5]$ in X, $[-4.3,3.7]$ in Y, and $[-4.5,3.5]$ in Z.

We test our method at different grid resolutions. At each resolution, we compare the output of our test T_{our} to the true parity of PV points on each grid face T_{true}. A face is colored gray, red, blue, or not shown if the pair $\{T_{\text{our}}, T_{\text{true}}\}$ has value {Odd, Odd}, {Odd, Even}, {Even, Odd}, or {Even, Even}. The result is shown in Figure 10, first row. Note that our test always matches the true parity even at an extremely coarse grid resolution, where the vector field on a cell face may vary significantly and there may be multiple PV points on one face.

We also tested existing methods based on boundary-sampling. In our implementation, we use the same boundary samples of the cell face M as in our method, and we compute the winding number of v at the samples after projecting to either the supporting plane of M [2, 5] (which we call the Face Plane test) or the plane orthogonal to the average of samples of u along ∂M [14, 10] (which we call the Average Plane test). The output of a test is the parity of the winding number. The results of these two tests, compared with the true parity, are shown in the second and third rows of Figure 10. Note that the Face Plane test consistently produces errors even at fine grid resolutions. The Average Plane test tends to make accurate decisions as a finer grid resolution, but fails at a low grid resolution.

We take a closer look at the failure cases of previous methods in Figures 11 and 12. Figure 11 examines a face with no PV point, but an odd winding number is reported by both Face Plane and Average Plane tests (i.e., a false positive). Figure 12 examines a face with one PV point, but an even winding number is reported by these tests (i.e., a false negative). Note that the choice of the projection plane in these tests has a significant impact on the winding of the projected 2D vectors (e.g., compare the lower pictures in (c,d) in each figure). On the other hand, our test does not require projection, and the correct parity is reported in both cases.

5 Application: Extracting Ridge and Valley Lines

Our test can be used in conjunction with a PV-extraction algorithm to ensure the closedness of the resulting curves wherever u, w are continuous. All that is needed for the algorithm is to compute seeds on a cell face whose parity matches the outcome of our test. For example, one can create one seed for each face where our test reports odd. For practical inputs, however, discontinuity of u, w generally exists. While the correctness of our parity test does not hold on faces containing such discontinuity, using our test still helps ensure well-connected curves elsewhere. We shall give an example of using our test in a typical 2-stage PV extraction framework for computing ridge and valley lines.

5.1 Method

The ridge and valley lines of a 3D scalar field $s : R^3 \rightarrow R$ can be defined using the gradient g and the eigenvectors of the Hessian H [9]. Let $\varepsilon_1, \varepsilon_2, \varepsilon_3$ be the three eigenvectors of H ordered such that their corresponding eigenvalues $\lambda_1, \lambda_2, \lambda_3$ have increasing absolute values. A point lies on the ridge or a valley if g is parallel to ε_1. The values of λ_1, λ_2 are both negative (resp. positive) if the point is on the ridge (resp. valley).

We can express ridges and valleys as PV by setting $u = \varepsilon_1$ and $w = g$. However, since ε_1 does not carry an orientation, it is technically a line field rather then a vector field. There are two approaches to resolve this difficulty. First, one can attempt to orient ε_1 locally on each cell face M by tracing along its boundary. The attempt will always succeed unless ε_1 is discontinuous somewhere on M [4], in which case our parity test would not be applicable anyway. Second, as done in several previous work [8, 9, 7], one can use an alternative PV definition of $u = Hg$, and select from the computed curves those that are parallel to only ε_1. The second approach avoids the need for orienting ε_1 and is more efficient because the eigen-problem is only solved during the selection process. However, the selection step may fail at locations on the curves where g and ε_1 are not aligned (e.g., due to numerical errors), which would lead to unnecessary disconnections in the resulting curves. Since our emphasis is on curve connectivity, our implementation takes the first approach.

In seed-extraction stage, we produce samples along the face boundary in an adaptive manner to ensure low variability of u, v. This is done using a recursive binary splitting strategy that adds a sample point between two existing ones until either the angle between u (and v) at successive samples falls under a given threshold or the distance between the samples is shorter than another threshold. After applying our parity test using Equations 8,11,15, and if the test reports odd, we compute a seed by applying a few 2D Newton-Raphson iterations starting from the face centroid. In seed-connection stage, we follow the simple strategy of connecting seeds by straight segments within a cell (more complicated tracing methods can also be applied). To deal with different number of seeds in a consistent manner, we additionally create a vertex (at the centroid of the face seeds) within a cell that has more than one seed and connect this vertex to each seed.

5.2 Results

We first test our method on a smooth distance field defined on a point set. The initial scalar value at a grid point is the Euclidean distance to the nearest point in the set. The grid is then smoothed using Gaussian kernels, and the scalar function s is defined by tricubic interpolation. An example result of our method for a *Trefoil Knot* is shown in Figure 13. We also compare with the results using the Face Plane test and the Average Plane test instead of our parity test. Note that the Face Plane test results in a large amount of spurious branches. While the Average Plane test produces much fewer noise, the resulting valley curve still contains disconnection and branching, due to a low grid resolution (20^3). On the other hand, our parity test gives a continuous knot as the valley curve.

Figure 14 shows the result of our method applied to two real-world data sets. One is the cryo-electron microscopy scan of a protein (BVP8), where the high density region is the protein backbone. The other one is a CT scan of blood vessels in the brain. In both data, ridge lines capture the meaningful features. For practical data like these, there is usually a fair amount of spurious ridge and valley lines that need to be pruned. A number of strategies for pruning ridge and valley lines have been used in the past [9]. Here we applied simple pruning

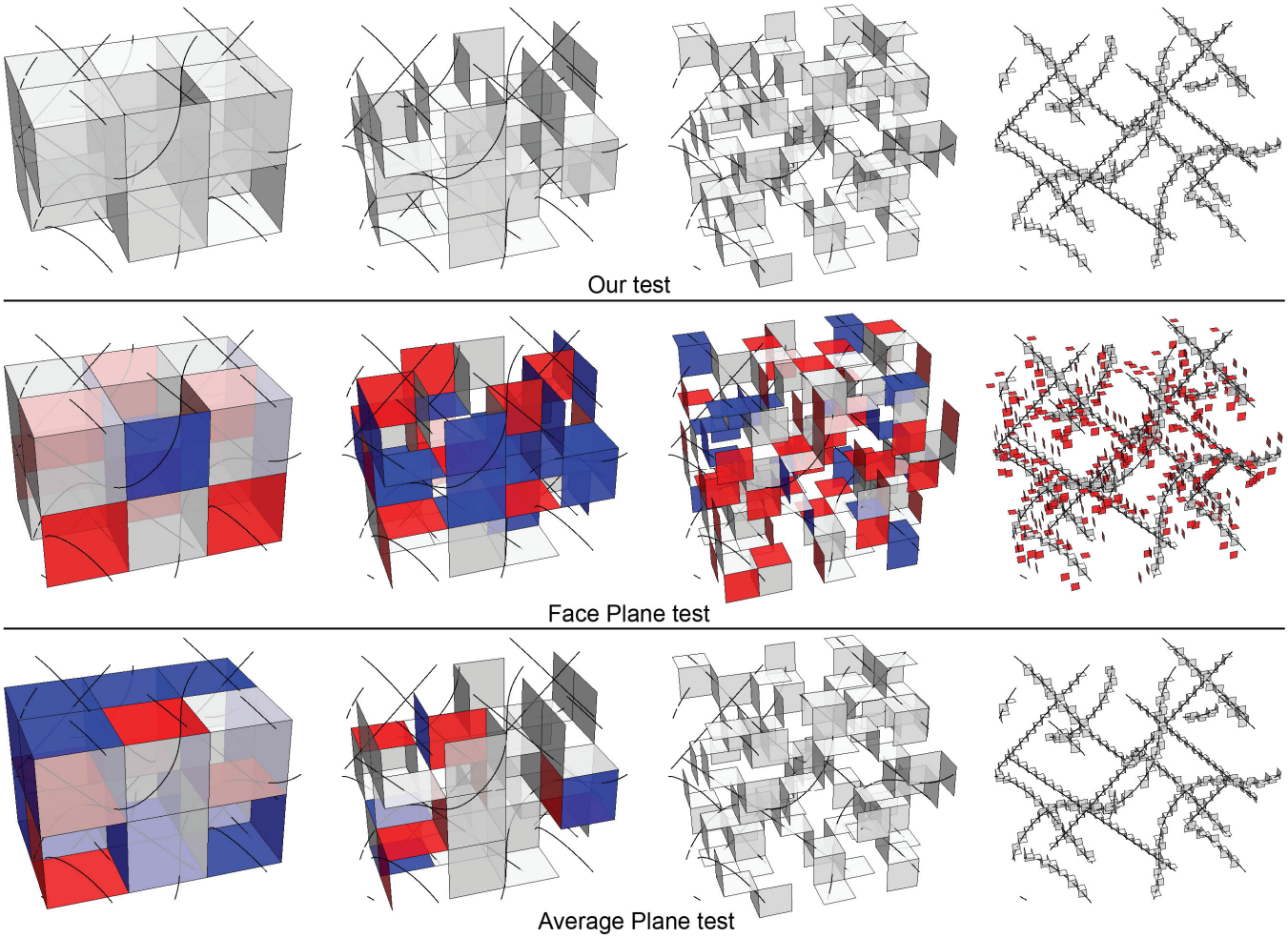

Fig. 10. Comparison of three parity tests on our synthetic data at four different grid resolutions (we used grid spacings 3.31, 2.31, 1.31, 0.31). Red (resp. blue) cell faces are where the test reports odd (resp. even) whereas the true parity is even (resp. odd).

by thresholding the scalar value at the extracted PV points and removing short curve segments.

Since our parity test involves independent operations on each cell face, it can be easily parallelized. Our tests are performed on an Intel Xeon E5-2440 machine with 8 core 2.40G Hz CPU. The computational time, including both the parity test and curve extraction, ranges from 10 seconds for the protein data (on a 96^3 grid) to 20 minutes for the vessel data (on a 512^3 grid).

6 Conclusion

We derive, for the best of our knowledge, the first robust test for the parity of the number of PV points on a genus zero surface with boundary in 3D. Similar to classical zero-crossing tests, our test only requires sampling along the boundary of the domain. A discretization of the test is described, validated, and compared with existing tests that are also based on boundary sampling. We also showed an application of the test for extracting continuous ridges and valleys.

Our parity test can be applied to arbitrarily shaped cell faces (e.g., triangles, hexagons, etc.) and even curved faces, and hence it is well suited for PV extraction in unstructured grids. Our test can also be used as an additional criteria for grid subdivision (e.g., subdivide if the parity is odd), which is particularly useful for tracing PV in higher-order fields [7].

While our test is restricted to disk-like surfaces in three dimensions, it would be interesting to explore its extension beyond this setting. As a starting point, we have verified that Lemma 4.1 should hold for surfaces of higher genus as well.

Acknowledgments

The work of the first author is supported by NSF grants (IIS-0846072, IIS-1302200, IIS-1319573, DBI-1356388). The work of other authors is supported in part by NSF grants (CMMI-1039433, CC-NIE-1245795).

References

[1] A. V. Gelder and A. Pang. Using pvsolve to analyze and locate positions of parallel vectors. *IEEE Trans. Vis. Comput. Graph.*, 15(4):682–695, 2009.

[2] G. Guy and G. G. Medioni. Inference of surfaces, 3d curves, and junctions from sparse, noisy, 3d data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(11):1265–1277, 1997.

[3] S. Kobayashi and K. Nomizu. *Foundations of Differential Geometry (volumes 1 and 2)*. Wiley, 1996.

[4] R. Li, L. Liu, L. Phan, S. Abeysinghe, C. Grimm, and T. Ju. Polygonizing extremal surfaces with manifold guarantees. In *Proceedings of the 14th ACM Symposium on Solid and Physical Modeling*, SPM '10, pages 189–194, 2010.

[5] S. Mann and A. P. Rockwood. Computing singularities of 3d vector fields with geometric algebra. In *IEEE Visualization*, pages 283–289, 2002.

[6] M. Morse. Singular points of vector fields under general boundary conditions. *Proceedings of the National Academy of Sciences*, 14(5):428–430, 1928.

[7] C. A. Pagot, D. K. Osmari, F. Sadlo, D. Weiskopf, T. Ertl, and J. Comba. Efficient parallel vectors feature extraction from higher-order data. *Comput. Graph. Forum*, 30(3):751–760, 2011.

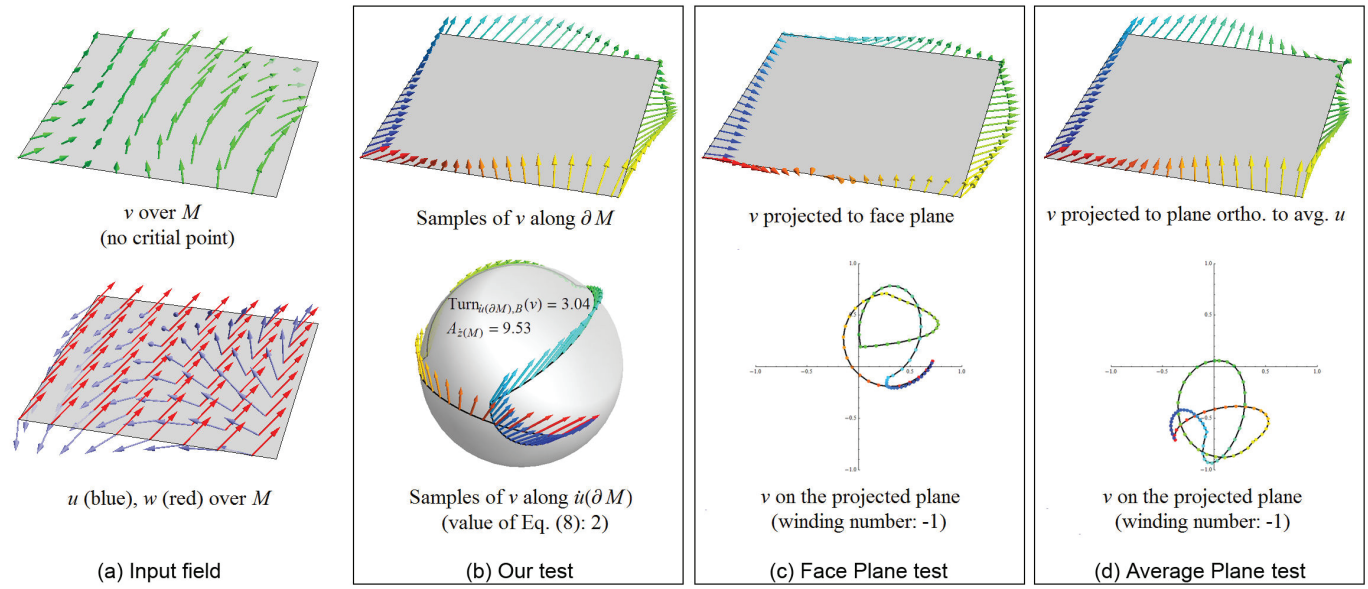

Fig. 11. A cell face in our synthetic input. This face has no PV point (as seen in (a)). Our parity test correctly reports even, while the other two tests both report odd. The coordinates of the two corners of the face are $\{4.24, 0.32, -4.5\}$ and $\{4.24, 2.63, -2.19\}$.

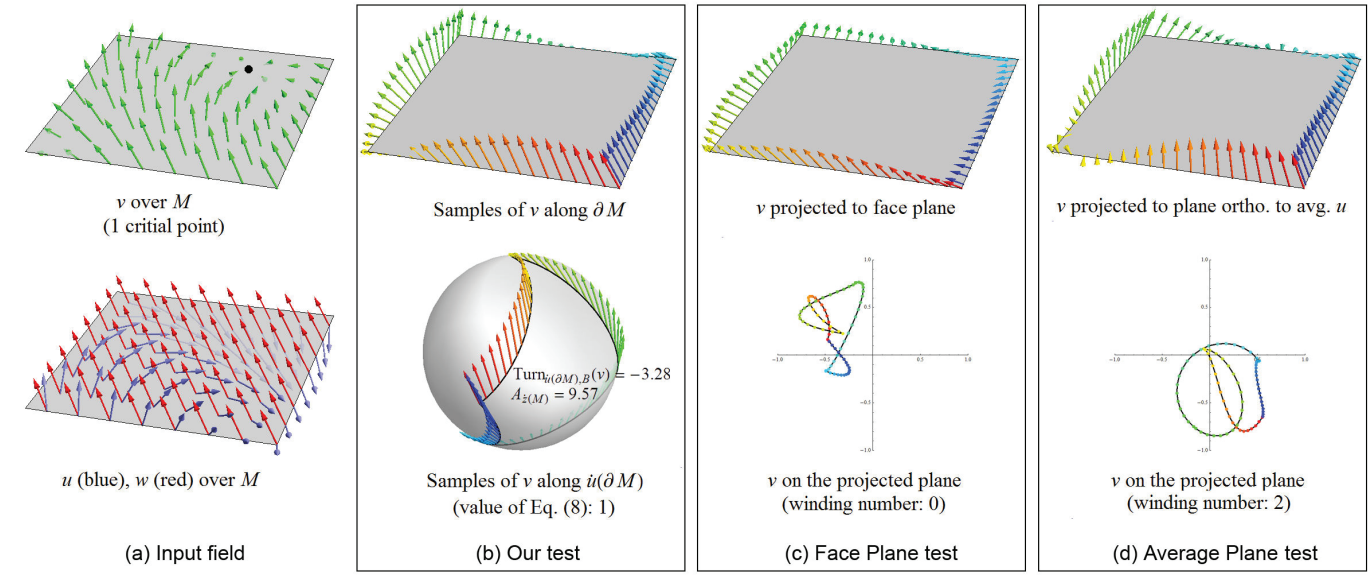

Fig. 12. Another cell face in our synthetic input. This face has one PV point (indicated by the dot in (a)). Our parity test correctly reports odd, while the other two tests both report even. The coordinates of the two corners of the face are $\{-5, 2.63, -2.19\}$ and $\{-2.69, 2.63, 0.12\}$.

[8] R. Peikert and M. Roth. The parallel vectors operator: A vector field visualization primitive. In *Proceedings of the Conference on Visualization '99: Celebrating Ten Years*, VIS '99, pages 263–270, 1999.
[9] R. Peikert and F. Sadlo. Height ridge computation and filtering for visualization. In *PacificVis*, pages 119–126, 2008.
[10] S. Pollock and S. Mann. Vortex detection in vector fields using geometric algebra. *Advances in Applied Clifford Algebras*, pages 1–20, 2013.
[11] M. Roth. *Automatic Extraction of Vortex Core Lines and Other Line-Type Features for Scientific Visualization*. PhD thesis, ETH Zurich, Institute of Scientific Computing, 2000.
[12] G. Scheuermann and X. Tricoche. *Topological Methods in Flow Visualization*, pages 341–358. Elsevier, Amsterdam, NL, 2005.
[13] J. Sukharev, X. Zheng, and A. Pang. Tracing parallel vectors. *Proc. SPIE*, 6060, 2006.
[14] C.-K. Tang and G. G. Medioni. Inference of integrated surface, curve, and junction descriptions from sparse 3d data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(11):1206–1223, 1998.
[15] H. Theisel, J. Sahner, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Extraction of parallel vector surfaces in 3d time-dependent fields and application to vortex core line tracking. In *IEEE Visualization*, page 80, 2005.
[16] J.-P. Thirion and A. Gourdon. The 3d marching lines algorithm. *Graphical Models and Image Processing*, 58(6):503 – 509, 1996.
[17] W. Wang, B. Jüttler, D. Zheng, and Y. Liu. Computation of rotation minimizing frames. *ACM Trans. Graph.*, 27(1):2:1–2:18, Mar. 2008.

A Proof of Lemma 4.1

Proof: If the Jacobian determinant of $\dot{u}$ is positive everywhere on M, $\dot{u}(M)$ is a smooth surface whose unit normal vector at a point $\dot{u}(x)$ is exactly $\dot{u}(x)$. Hence Lemma 4.1 holds by the derivation in Section 4.1.2, letting $S = \dot{u}(M)$.

If the Jacobian determinant of $\dot{u}$ is negative everywhere, $\dot{u}(M)$ is still a smooth surface but the normal at a point $\dot{u}(x)$ is $-\dot{u}(x)$. Since v is tangent to $\dot{u}(M)$, Equation (5) still holds for $S = \dot{u}(M)$. However, due to the flipping of normal orientation, the PV index at a PV point $x \in M$ is the negative Poincare index of v at $\dot{u}(x) \in S$. Similarly, we have $\mathrm{Turn}_{\dot{u}(\partial M),B}(v) = -\mathrm{Turn}_{\partial S,S}(v)$, and $A_{\dot{u}(M)} = -k_S$. Hence Lemma 4.1 still holds.

Now consider the scenario where the Jacobian determinant of $\dot{u}$ has mixed signs over M. In general, the loci where $\dot{u}$ has zero Jacobian

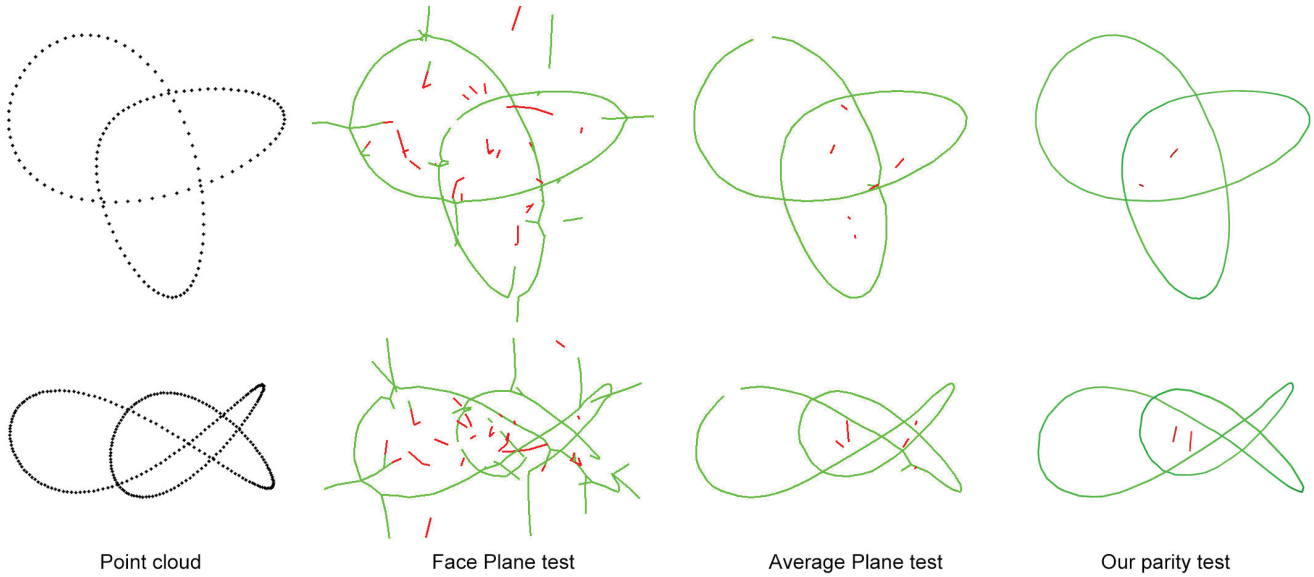

Fig. 13. Ridge (red) and valley (green) lines extracted from a smoothed distance field to a point cloud (representing a Trefoil Knot) using different parity tests (grid resolution: 20^3). The two rows view the same data from different angles.

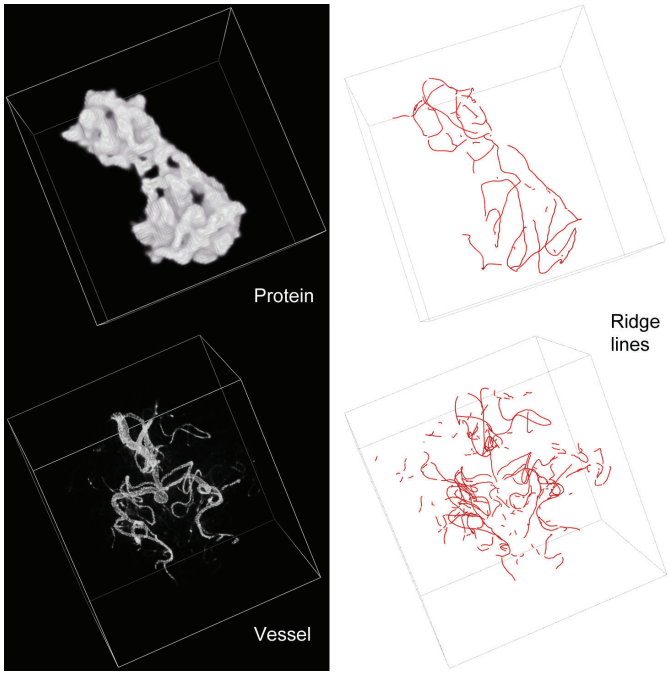

Fig. 14. Ridge lines extracted from real-world data.

determinant forms a graph, which partitions M into patches $\{M_i\}$ such that the Jacobian determinant of $\dot{u}$ has the same sign within a patch. Since Lemma 4.1 holds within each M_i, the total PV index over M, which is the summation of total index over each M_i, equals

$$\frac{\sum_i \mathrm{Turn}_{\dot{u}(\partial M_i),B}(v) + \sum_i A_{\dot{u}(M_i)}}{2\pi}$$

To prove Lemma 4.1 over entire M, it suffices to establish the following two identities

$$A_{\dot{u}(M)} = \sum_i A_{\dot{u}(M_i)},$$

$$\mathrm{Turn}_{\dot{u}(\partial M),B}(v) = \sum_i \mathrm{Turn}_{\dot{u}(\partial M_i),B}(v)$$

The first identify holds because of the integral nature of the signed area. To show the second identity, consider the graph of the curves that partition M into patches M_i. Each edge in the graph is either shared by the boundaries of two patches (traversed in opposite directions by the two boundaries) or lying on ∂M. Now, write $\mathrm{Turn}_{\dot{u}(\partial M_i),B}(v)$ for each patch M_i as the sum of turnings over edges in the graph (note that a parallel vector field is continuous even at C^0 corners of a curve). In the summation $\sum_i \mathrm{Turn}_{\dot{u}(\partial M_i),B}(v)$, the turnings on an edge shared by two patches cancel each other out, and hence only the turnings along the edges that are on ∂M remain, which is exactly $\mathrm{Turn}_{\dot{u}(\partial M),B}(v)$. □

B Proof of Lemma 4.2

Proof: Consider the surface $\bar{M}$ that is identical with M but with an opposite orientation (so that the normal field of $\bar{M}$ is opposite to that of M). Create a glued surface $H = M \cup \bar{M}$. H is a closed surface homeomorphic to a sphere. Define mapping $h : H \to B$ (where B is the unit sphere) as

$$h(x) = \begin{cases} \dot{u}(x), & \text{if } x \in M \\ \dot{z}(x), & \text{if } x \in \bar{M} \end{cases}$$

Since z, u are identical along the boundary of M and $\bar{M}$, h is continuous. It is a well-known fact that the signed area of $h(H)$ is the area of the unit sphere, 4π, times the topological degree of the mapping h, which is an integer. Hence we arrive at Equation 7 by noting that $A_{h(M)} = A_{\dot{u}(M)} + A_{\dot{z}(\bar{M})}$ and $A_{\dot{z}(M)} = -A_{\dot{z}(\bar{M})}$. □