Computing Singularities of 3D Vector Fields with Geometric Algebra

Stephen Mann*
University of Waterloo

Alyn Rockwood†
Colorado School of Mines

ABSTRACT

Critical points of a vector field are key to their characterization. Not only their positions but also their indexes are crucial for understanding vector fields. Considerable work exists in 2D, but less is available for 3D or higher dimensions. Geometric Algebra is a derivative of Clifford Algebra that not only enables a succinct definition of the index of a critical point in higher dimension; it also provides insight and computational pathways for calculating the index. We describe the problems in terms of Geometric Algebra and present an octree based solution using the algebra for finding critical points and their index in a 3D vector field.

CR Categories: G.4 [Mathematical Software]: Algorithm design and analysis—Visualization

Keywords: Geometric Algebra, 3D Vector Fields, Singularities

1 INTRODUCTION

We define the vector field to be a continuous function $V : \mathbf{M} \rightarrow \mathbf{R^n}$, where $\mathbf{M}$ is a manifold in $\mathbf{R^n}$. If $\mathbf{x} \in \mathbf{M}$, then the vector at $\mathbf{x}$ is $V(\mathbf{x})$. A point $\mathbf{x}$ is a critical point of the vector field if $|V(\mathbf{x})| = 0$. For example, many useful vector fields are generated as gradient fields of differentiable potential functions $P : \mathbf{M} \rightarrow \mathbf{R}$, in which case the critical points occur where $|\nabla P(\mathbf{x})| = 0$. The Gauss map $\gamma : \mathbf{M} \rightarrow S^{n-1}$ (the sphere in n-dimensions) is defined by

$$\gamma(\mathbf{x}) = V(\mathbf{x})/|V(\mathbf{x})| \tag{1}$$

for all non-critical points $\mathbf{x}$. Consider an arbitrarily small ball $\mathbf{B}(\mathbf{c})$ about a critical point $\mathbf{c}$. The index of $\mathbf{c}$, ind($\mathbf{c}$), is given by the Gauss-Bonnet Theorem [7] that says

$$\int K d\gamma(\mathbf{B}(\mathbf{c}))/(\text{volume of } S^{n-1}) = \text{ind}(\mathbf{c}), \tag{2}$$

where K is the normal curvature of $\gamma(\mathbf{B}(\mathbf{c}))$. The normal curvature of a curve is the infinitesimal change of length at $\mathbf{x}$ on a curve compared to the change in $\gamma(\mathbf{x})$. Similarly, for a surface we compare the changes in surface areas. One would think that this would extend to higher dimensional volume changes, but this is not the case for historical reasons. Nevertheless it is the definition with which we will work (see Gottlieb [7]). Hence we compare the infinitesimal change of volume (length, area, etc.) of $\mathbf{B}(\mathbf{c})$ to $\gamma(\mathbf{B}(\mathbf{c}))$. In 2D the Gauss map records how many times the vector field on the circle $\mathbf{B}(\mathbf{c})$ cycles, or winds, as one follows the path once around the circle. The winding may be in the opposite direction of the path, or go multiple times around, but it will always be an integer when properly normalized by the volume of the $n-1$ sphere. See Figure 1 for examples.

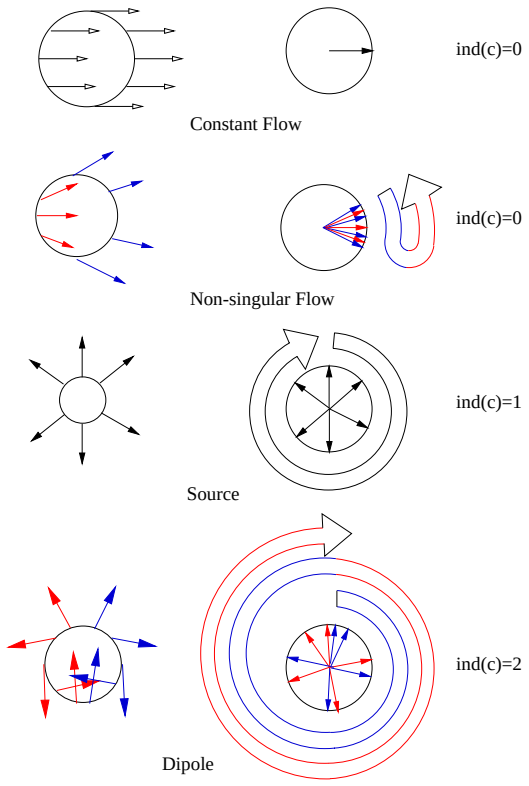

Figure 1: Vector fields on circles about critical points, their Gauss maps and the indexes.

For obvious reasons the index in 2D is often called the winding number, an unfortunate choice of terminology, since it becomes confusing in higher dimensions. A better choice would have been to call it a covering number. One can imagine the Gauss map of the ball $\mathbf{B}$ being some multiple in surface area compared to the original ball, i.e., it wraps it several times. Since the area is oriented, it can also be some negative multiple. This is the index. The index of a critical point is a crucial factor in determining the topology of the vector field in various well-known display algorithms (see [8, 17]) It is our aim to provide an alternative way to view the index using Geometric Algebra, which then leads to alternative ways to calculate. Ultimately this path of investigation will lead to a number of unique application issues; ones that are algorithmic, numerical and visual.

2 GEOMETRIC ALGEBRA AND THE INDEX

In brief, Geometric Algebra (GA) is a graded, non-commutative (Clifford) algebra, which is geometrically intuitive (introductions to GA can be found in [3, 4, 12]). Elements of the algebra are di-

*School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1 CANADA, smann@uwaterloo.ca

†Colorado School of Mines, Dept. Of Math and Computer Science, Golden, CO 80401, alynrock@mines.edu

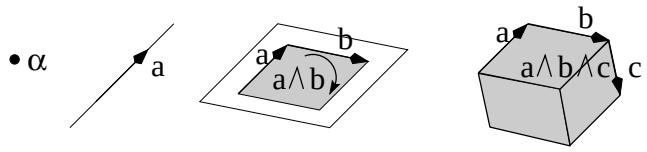

Figure 2: Spanning subspaces with the outer product.

mensionally homogeneous (grade = intrinsic dimension) such as the familiar scalars (0D), vectors (1D), as well as area oriented bivectors (2D), volume oriented trivectors (3D) and so forth. Multivectors are combinations of the k-vectors. The fundamental geometric product defines inner, i.e., grade lowering, products; and outer, i.e., grade raising, products. The outer product "$\wedge$" defines a k-vector $V = a_0 \wedge a_1 \wedge \ldots \wedge a_k$ where the a_i are vectors. "Wedging" a vector with a k-vector raises its grade one step. In GA, the k-vectors are the basic elements of computation, and can be added and combined with all the products. Examples of k-blades of dimensions 0,1,2,3 are shown in Figure 2. Although this figure indicates a particular shape for the outer product, a k-blade does not have a specific shape, only a signed magnitude and orientation.

GA also includes the pseudo-scalar I, which is the maximal grade, unit element. Multiplying an element by the pseudo-scalar yields an element that is dual in grade; thus an $n - i$ grade element becomes an i grade element and vice-versa. These few facts are needed to understand the GA version of the index theorem. Pauli Algebra is one model of GA although for purposes of insight and algebraic thinking it is a clumsy one, especially as the dimension increases. It is much more effective to use the axioms and the svelte body of theorems of GA [9]. Pauli Algebra does, however, offer one method by which GA can be mapped to current computer systems, one which we take advantage of in what follows computationally (see GABLE etc. [5, 6, 14]).

Hestenes ([10] p.275) gives a general formula for computing the index of a critical point of a vector field V on a manifold using GA. For 3D this formula reduces to

$$\text{ind}(\mathbf{c}) = \frac{C}{I} \int_{\mathbf{B}(\mathbf{c})} V \wedge dV / |V|^3. \tag{3}$$

The constant $C = 1/(6 \times 4\pi/3)$ of (3) contains the sphere normalization factor, with an additional factor of 6 to adjust the volume of the trivectors, whose magnitude is that of a parallelepiped having six times the volume of the desired tetrahedron. The radius part of the sphere factor is absorbed into the denominator inside the integral, $|V|^3$. The extended differential dV is a bivector perpendicular to V [10]; thus $V \wedge dV$ represents an infinitesimal volume element. This differs from (2) which uses surface elements for comparison as approximated by dV, but the difference between the two are one of proportionality. Moreover, the discretized volume elements are perceived to be a closer approximation than the surface case for the same discretization. Dividing by the pseudoscalar I converts the volume element generated by the integral to a scalar.

3 Finding Critical Points

Formula (3) is a straightforward roadmap for computing the index of a critical point in GA. It is intuitive and conceptually easy to program in a GA system (not to minimize some of the implementation details given later). Another advantage is that the sense, or sign, of the elements is automatically tracked within GA. We do not have to keep track of surface orientations, i.e., is it back facing or not?

Our goal is not only to compute such indexes, but also to find the critical points in the first case. To do this we will employ a

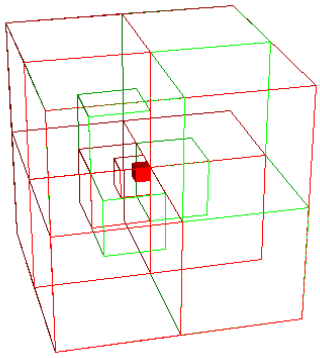

Figure 3: Octree search of $V((x, y, z)) = x\, e_1 + y\, e_2 + z\, e_3$.

cuberille subdivision of space. To understand how this will work we need to make several observations. The first is that a cube is a sphere - topologically speaking. Anything homeomorphic to the ball $\mathbf{B}$ will work in (3). We compute (3) over each cube, which now tiles the space and provides a complete, non-redundant covering not proffered by balls. If the index of a computed cube is 0, then it may be assumed that there is no critical point to within the resolution of the cube. It can happen that there are multiple critical points that sum to 0. Whenever critical points are close their aggregate behavior acts like a single critical point with index equal to the sum of indexes of the constituent critical points. For index 0, it will not be apparent that there is a critical point unless higher resolutions are investigated.

If the index of a cube is nonzero, then a critical point has been found, or perhaps a collection of critical points close to each other. Such collections of critical points are indistinguishable from a single critical point of the same index up to the resolution computed. This emphasizes the crucial role that setting the resolution has. It also indicates an important trade-off between computational cost and accuracy. In some applications one knows that a given distance isolates critical points. In such cases an octree algorithm can be used to precisely locate those points within cubes that are identified as containing critical points. The octree subdivides whenever the cube has nonzero index. It is analogous to the root finding problem. Knowledge that roots are separated greatly improves speed and precision. Figure 3 shows an octree example of locating the critical point for the field given by $V((x, y, z)) = x\, e_1 + y\, e_2 + z\, e_3$ (See section 6 for more details on this and other examples). In this, as in most of our figures, we will omit drawing the vector field and instead focus on the singularities in the vector field.

4 Related Work

There is a considerable amount of work on display of vector field topology, which employs linear expansions of the field in the neighborhood of critical points [8, 15], i.e., use of the eigenvalues of the Jacobian tensor of the field. These methods, however, fail to find the behavior of fields that contain critical points with index other than 1 or -1. As mentioned, Scheuermann et al. discuss methods for determining high order indexes, but they are limited to 2D [17, 18, 19]. Trotts et al. introduce the point of infinity as a critical point for infinite fields [20] .

The use of octrees for rendering implicit surfaces is closely related to our work [2], with the primary differences being the type of field being searched (scalar vs vector) and that with implicit surfaces, the value of the scalar function is computed at the corners of

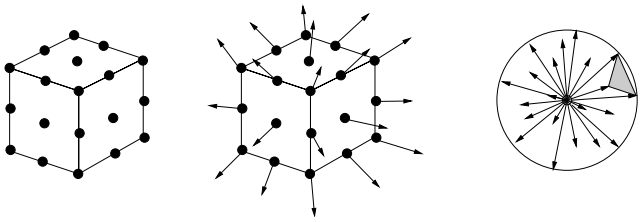

Figure 4: Sampling V over a cube and summing the trivectors.

the cube, while with our method we evaluate a vector field at a grid of samples on the faces of the cube.

5 IMPLEMENTATION

Moving from the theory to the basic implementation is straightforward (Figure 4). The integral (3) over the surface of the cube is approximated by sampling each face f of the cube on a regular grid $\mathbf{p}_{i,j}$, evaluating the vector field at each sample point giving vectors $p_{i,j} = V(\mathbf{p}_{i,j})$, normalizing these vectors, forming the trivectors

$$\begin{aligned} R_{i,j} &= \hat{p}_{i,j} \wedge \hat{p}_{i,j+1} \wedge \hat{p}_{i+1,j} \\ S_{i,j} &= \hat{p}_{i+1,j+1} \wedge \hat{p}_{i+1,j} \wedge \hat{p}_{i,j+1}, \end{aligned}$$

and summing the trivectors $R_f = \sum R_{i,j}$ and $S_f = \sum S_{i,j}$. Summing the R_f,S_f over all six faces of the cube and normalizing by a factor of $1/(6 \times 4\pi/3)$ should give a result close in value to an integer that is the index of a critical point inside the cube (if any).

However, several problems arise. First, as mentioned earlier if the cube size is too large, then critical points may be missed or misclassified. Second, if the sampling grid on the cube faces is too sparse, then the resulting sum might not be close to an integer (i.e., the approximation to the integral is too coarse). Third, the integral over the face of the cube will only detect point singularities; we must use other methods to detect curve and surface singularities.

We will not address the first issue in this paper; the remainder of this section will address the other two issues.

5.1 Curve singularities

When a curve singularity passes through the cube over which we are summing, the sum of trivectors over the faces of the cube will be 0. Thus, it will fail to detect the curve singularity. However, consider the 2D version for the computation of indexes using GA:

$$\text{ind}(\mathbf{c}) = \frac{C_2}{I_2} \int_{\mathbf{B}(\mathbf{c})} V \wedge dV/|V|^2. \tag{4}$$

Here, the differential dV is a vector perpendicular to V; C_2 is $1/(2\pi)$ (with a normalization factor of π for the area of the unit circle, and a normalization factor of 2 to account for bivectors having a signed magnitude equal to the area of a parallelogram that is twice the area of the desired triangle); and I_2 is the unit bivector in the plane. Returning briefly to our 3D problem, consider a planar slice Π of space, and define a new vector field V' where for $\mathbf{x} \in \Pi$, the field $V'(\mathbf{x})$ is the projection of $V(\mathbf{x})$ onto Π. Applying (4) to a closed curve $\mathbf{B}$ in Π over the vector field V', then a non-zero result indicates that there is a point singularity on Π inside $\mathbf{B}$ over V'. In turn, we know one of the following about V:

(a) a point singularity of V lies on Π inside $\mathbf{B}$;
(b) a 3D curve singularity of V passes through $\mathbf{B}$;
(c) for some point $\mathbf{x}$ on Π inside $\mathbf{B}$, $V(\mathbf{x})$ is non-zero while $V'(\mathbf{x})$ is zero.

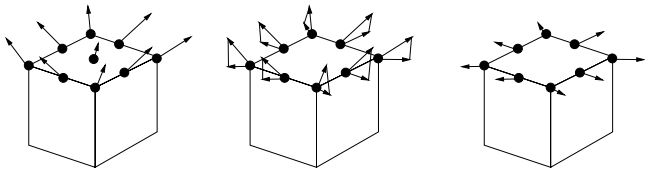

Figure 5: Projection of vector field onto one face of the cube.

Our interest is in case (b), that of curve singularities passing through the face of the cube. To distinguish case (b) from case (a) requires looking more globally at the vector field. Case (c) is a "false singularity," i.e., a singularity in the projected field that is not a singularity in the unprojected field. Such a singularity results when the projected vector is perpendicular to the face of the cube. To distinguish case (b) from case (c), we must localize the singularity (i.e., find a small $\mathbf{B}$), and test the vector field over this smaller region to determine if the vector field is non-zero over this region.

Applying these ideas to our implementation, for each cube face, we project the $p_{i,j}$ on the edge of the face into the plane of the face and renormalize, giving vectors $\hat{p}'_{i,j}$ (Figure 5). Along each edge, we form the bivector sum

$$\sum \hat{p}'_{0,j} \wedge \hat{p}'_{0,j+1}, \qquad \sum \hat{p}'_{i,n} \wedge \hat{p}'_{i+1,n},$$
$$\sum \hat{p}'_{n,n-j} \wedge \hat{p}'_{n,n-(j+1)}, \qquad \sum \hat{p}'_{n-i,0} \wedge \hat{p}'_{n-(i+1),0},$$

where $n+1$ is the number of samples along each edge. Adding the four sums and normalizing by C_2/I_2 yields the desired index. If the index is non-zero, then one of the three forms of singularities described above lie on the face.

To determine if the singularity is of type (b), we could allow the octree subdivision to run to completion, and then test one unprojected vector on the (small) face at the deepest level of the subdivision. If its magnitude is large, then we have a singularity of type (c), which we can disregard. To decide if the singularity is of type (a), we should test the other faces of the cube for a second such singularity; if there isn't one, then the singularity is of type (a), and should be treated as a point singularity. If there is a second face with such a singularity, then we have found a curve singularity.

Such an approach is inefficient, however, as singularities of type (c) (i.e., singularities in V' that are not singularities in V) will be common, but are of no use to us. Thus, allowing the octree subdivision process to localize these singularities is wasted computation. A heuristic to attempt to eliminate such false singularities earlier in the subdivision process is to check the length of all the unnormalized vectors in V sampled on the face of the cube: If we detect a potential curve singularity on a face, and none of the sampled unprojected vectors is close to 0, and the ratio of the largest unprojected vector to the smallest is small, then the potential singularity is likely of type (c) and should be disregarded (i.e., the octree cell should not be subdivided based on this test).

In some sense, this heuristic is performing a simple numerical search to localize the singularity. From this point of view, we could either (a) use the search to speed the octree subdivision, or (b) use the search instead of (4) to find the curve singularity. While both are reasonable variations of our method, note that (4) is still useful for determining the index of the curve singularity.

Further note that our method for detecting curves of singularities is similar to the method of Jiang et al. for detecting a vortex core region [11]. Like us, Jiang et al. project the 3D vector field into a plane. However, they then use Sperner's lemma as a basis for finding where the vortex core passes through the plane rather than using geometric algebra to find a zero of the vector field. (Sperner's lemma basically says that if you start with a properly labeled n-simplex (i.e., one whose vertices have unique labels), then a labeled

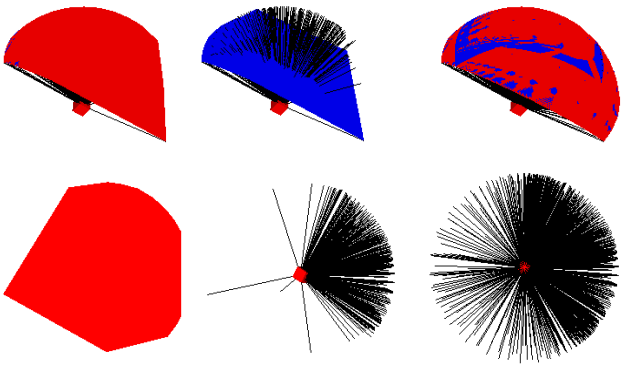

Figure 6: Refinement examples.

subdivision of this simplex will have an odd number of properly labeled simplices, where the labeling of the vertices of the subsimplices have to obey some minor constraints; see the Jiang et at. paper for details.)

5.2 Surface singularities

To compute a surface singularity, we essentially use the same idea as for computing curve singularities: for each sample point on the edge of the cube, project the corresponding vector onto that edge. The test for whether a surface singularity passes through the edge is simpler than in the case of curve singularities. No outer products are needed — if the projected vectors along an edge change orientation/sign, then there is a singularity in the projected vector field. But just as in the curve singularity case, further tests are needed to see if the singularity in the projected field corresponds to a surface singularity in the unprojected field. Note also that this test will only determine the existence of a surface singularity, without determining its index; see the Conclusions for further discussion of this issue.

5.3 Adaptive method

The method for computing point singularities omits some critical details for implementation, such as how many samples to take across the face of the cube. The trade-off is clear: more samples will give a more accurate estimate of the integral, but fewer samples are less expensive to compute. Ideally, we would take just enough samples so that the estimates are close to integer values. This leads to the idea of initially taking few samples, and if the result is not close to an integer value, then increase the sample rate.

We chose a variation of this idea. After sampling, when computing the outer products, we test the magnitude of each outer product. If the magnitude is large, then we have made a poor approximation to the sphere in between the three samples. In this case, we adaptively refine the region between the samples.

An example of refinement can be seen in Figure 6. On the top row is an example of a sampling that results in a positive region (red) that is a reasonable approximation to the sphere, while the negative region (blue) is a poor approximation to the sphere. The left image shows a display of all samples; the middle image shows just the negative region, but with all the sampled vectors drawn; the resulting sum yields an index of 0.3. The image on the right is after refinement, and has a computed index close to 0.0.

On the bottom row, left, we see a non-uniform sampling of the sphere, with the corresponding vectors shown in the middle image. The index computed here is 0.7. After refinement, we obtain the vectors on the right, with a computed index close to 1.0.

6 Examples, discussion

We tested our technique on a variety of functional vector fields. Table 1 lists the functions for the examples shown in this paper. In these functions, e_1, e_2, e_3 form an orthonormal basis for 3-space. Some of the examples are given as a vector function, while others are given as a scalar function whose vector field is the gradient of the scalar function.

In all examples, cubes containing point singularities are drawn in red, curve singularities in green, and surface singularities in blue. In all examples, we started with 9^2 samples per face. Only cubes containing a singularilty are shown. Cubes that did not contain singularities were also searched, but are not drawn in these figures; roughly speaking, this results in each drawn cube (except those at the maximum depth searched) being subdivided one additional level.

If a singularity lies on a cube corner, edge, or face, the integral over the cube (and its neighbors that share the singularity) will be 1, but the approximations to the integrals may require several levels of refinement to get close to an integer value. To simplify things, we offset the corners of the initial octree cube from an integer grid by 0.05 to avoid having the singularities land on a cube corner, edge or face.

Figure 7 illustrates our method searching a vector field that has a circle of singularities. On the left is the result when both line and surface false singularities are filtered. Note that this vector field has a point singularity at the origin in addition to a circle of singularities. In the middle, we see the result if we do not filter line singularities, while on the right is the result if we filter neither line nor surface singularities.

In Figure 8, we see an example of a double circle (eight) of singularities, a helix of singularities, and a sphere of singularities. Note that in the double circle of singularities and in the helix of singularities, there are cubes that are incorrectly identified as containing surface singularities. This is a result of the heuristics failing to remove the false surface singularities in these regions.

Finally, in Figure 9 we show an example of taking the union and intersection of two vector fields, one of which has a sphere of singularities and the other of which has a cylinder of singularities.

We also tested our method on vector fields having double point and double line singularities. The function having a double line singularity we tested was $V_{dl}(x, y, z) = (A/B)A$, where $A = x\,e_1 + y\,e_2$ and $B = \sqrt{x^2 + y^2}\,e_1$. In this expression for V_{dl}, we are using the vector multiplication of geometric algebra; note that all vectors in the field V_{dl} will have a 0 component in the e_3 direction due to the properties of the geometric multiplication. Figure 10 shows a rendering of this vector field, together with the double line singularity that our algorithm found.

One of the double point singularity functions we tested was $V_{dp}(x, y, z) = (A/B)A + z\,e_3$, where A and B are defined as in the previous paragraph. The double point singularity octree figure looks identical to Figure 3. In the vector fields V_{dl} and V_{dp}, we detected the singularities as having index 2. However, as we approached these singularities of higher order, refinement of the sampling of the cube faces became more important for computing accurate estimates of the integral.

Table 6 gives the times required to compute the examples in this paper. These timings were made on a Celeron 400MHz PC under Linux.

Type	Scalar Function	Vector Function
Point		$E_p(x,y,z) = x * e_1 + y * e_2 + z * e_3$
Circle	$f_c(x,y,z) = (x^2 + y^2 - 1)^2 + z^2$	$E_c(x,y,z) = -y * e_1 + x * e_2$
Eight		$E_e(x,y,z) = E_c(x-1,y,z) * E_c(x+1,y,z)$
Helix	$f_h(x,y,z) = 1/[(x - \cos(z))^2 + (y - \sin(z))^2]$	
Sphere	$f_s(x,y,z) = (x^2 + y^2 + z^2)^2$	
Cylinder	$f_C(x,y,z) = (x^2 + y^2 - 1)^2$	
Sphere Union Cylinder	$f_s(x,y,z) * f_C(x - .5, y, z)$	
Sphere Intersect Cylinder	$f_s(x,y,z) + f_C(x - .5, y, z)$	

Table 1: Test functions.

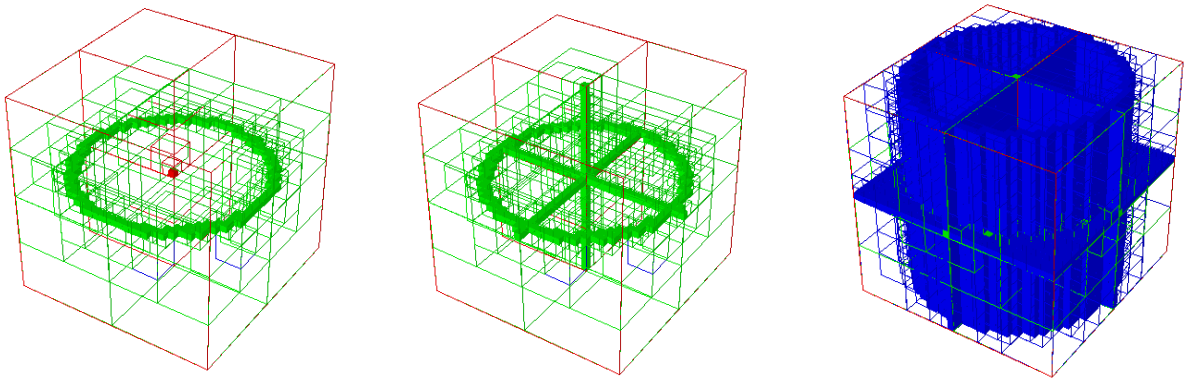

Figure 7: Circular curve of singularities. Left: all false singularities filtered; middle, no line filtering; right: no line or surface filtering.

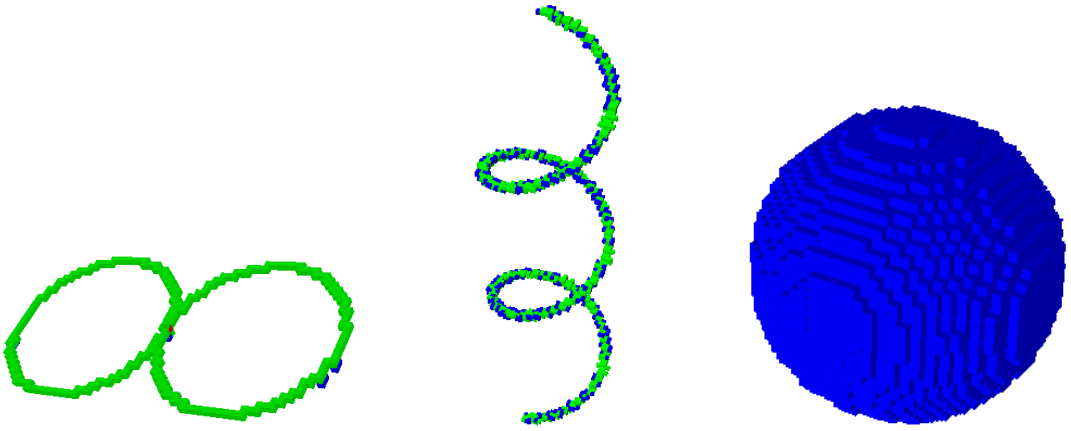

Figure 8: Double circle (eight) curve singularity, helix singularity, and sphere of surface singularity.

Function	Depth	Seconds	Cubes	Sec/Cube
point	4	1.6	105	.015
circle	5	20	1177	.016
eight	6	45	2945	.015
helix	7	145	9289	.016
sphere	5	158	10433	.015
sphere Union cyl	6	270	17673	.015
sphere Intersect cyl	6	38	1833	.020

Table 2: Timings for examples

7 VARIATIONS

Our implementation was a simple proof-of-concept prototype. Obvious improvements exist. For example, when drawing curve singularities, we drew the box containing the curve singularity. Instead, one could search the faces of the cube for close-to-zero values, and connect them with lines. A similar thing could be done for the surface singularities, although this is harder, as something similar to the surface extraction of Marching Cubes is required [13].

Further, while our method did an excellent job of finding point singularities, it had more troubles with curve singularities, and still more difficulties with surface singularities. In both cases, the problem is that the projected field may have singularities in it that are not in the original field. Filtering out these false singularities is problematic. One improvement would come from using a less local search method. Our method considers each octree cell independently. While a reasonable approach for point singularities, this method is wasting information it could effectively use to determine whether potential curve/surface singularities are true singularities.

In this paper, we applied our method to functional vector fields. Our method could also be applied to sampled vector fields. With a sampled field, one of the important issues is how to interpolate between the sampled vectors to fill space with a (continuous) vector field. Regardless of the interpolation method, our method would find the singularities within the interpolated sampled field. Furthermore, in interpolated fields, it is not uncommon to have point singularities that are quite close to one another [17]. Such near singularities cause problems for many methods, which may detect only a single point singularity. With our method, if the octree subdivision around the cluster singularity is not fine enough, it will detect a point singularity of index that is the sum of the cluster. With near singularities, that summed index represents the character of the vector field properly to within the resolution given.

8 CONCLUSIONS

In summary, we have presented a method for finding singularities in 3D vector fields. Our approach uses Geometric Algebra for computing a volume integral over the surface of a cube to detect point singularities, and uses octree subdivision to refine the location of a point singularity. In addition, our method can find curves of singularities, and surfaces of singularities, and it determines the index of the singularity. While finding any singularities by visually inspecting a 3D vector field can be a difficult task, computer assistance is especially needed for determining the index of singularities.

We note here several things of particular interest with our method. First, we found the use of Geometric Algebra to be a straightforward blueprint in coding the algorithm. While other methods could be used to estimate the integral, the trivector computation of Geometric Algebra automatically handles some of the geometric details, such as polarity and backfacing issues, simplifying the programming job.

Second, our method attempts to find curves of singularities and surfaces of singularities. Our approach here should be considered a first attempt. Much work remains to be done both (1) on the

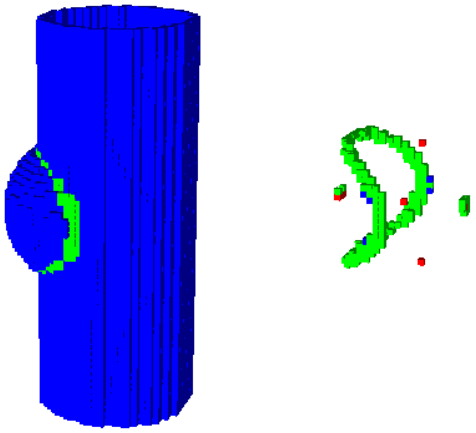

Figure 9: Surface singularities of sphere union cylinder and curve singularities of sphere intersect cylinder.

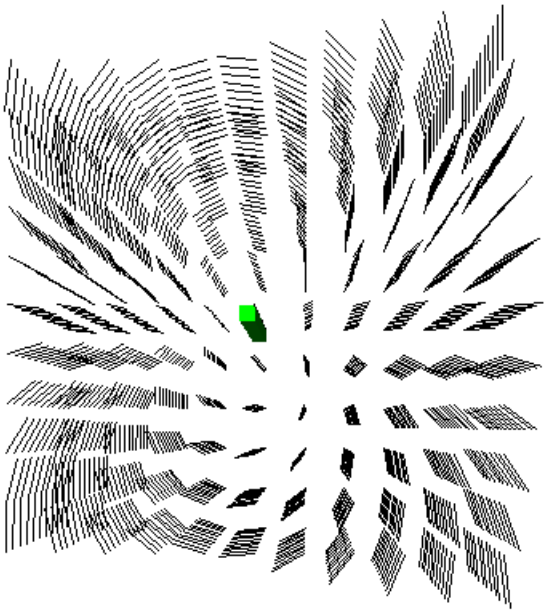

Figure 10: Vector field having a double line singularity.

theory of what a curve and surface singularity is, and (2) in methods for searching for such singularities. In particular, in searching for surface singularities, we project the 3D vector field on to lines, and search these 1D fields for singularities (which amounts to little more than searching for a change in orientation of the vectors along the line). This leads to the interesting question: simple sources and sinks in 1D vector fields are well known [1], but is there such a thing as a singularity of higher index in a 1D field? Our simple tests will detect only that there is a singularity on the line, but are unable to determine its index.

Further, our approach in finding curve and surface singularities is to project the vector field on to a lower dimensional space. This projection introduces new singularities into the field, for which we propose certain heuristics to remove. These heuristics are simple, and more work remains to improve them.

9 ACKNOWLEDGMENTS

Many thanks to Daniel Fontijne, who ported our initial Matlab/GABLE [5] prototype to a far faster C++ version using Gaigen [6].

REFERENCES

[1] Ralph Abraham and Christopher Shaw. *Dynamics : the geometry of behavior*. Addison-Wesley, second edition, 1992.

[2] Jules Bloomenthal. Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355, November 1988.

[3] Leo Dorst and Stephen Mann. Geometric algebra: a computation framework for geometrical applications: part i (algebra). *Computer Graphics and Applications*, 22(3), 2002. to appear.

[4] Leo Dorst and Stephen Mann. Geometric algebra: a computation framework for geometrical applications: part ii (applications). *Computer Graphics and Applications*, 22(3), 2002. to appear.

[5] Leo Dorst, Stephen Mann, and Tim Bouma. GABLE: a geometric algebra learning enviornment. `www.science.uva.nl/~leo/GABLE/`.

[6] Daniel Fontijne. Gaigen: Geometric algebra implementation generator. `carol.wins.uva.nl/~fontijne/gaigen/`.

[7] Daniel Henry Gottlieb. Functions and the unity of mathematics. Unpublished. `www.math.purdue.edu/~gottlieb/Papers/papers.html`.

[8] James Helman and Lambertus Hesselink. Visualizing vector field topology in fluid flows. *Computer Graphics and Applications*, 11(3):36–46, May 1991.

[9] David Hestenes. *New foundations for classical mechanics*. Reidel, Dordrecht, 2nd edition, 2000.

[10] David Hestenes and Garret Sobczyk. *Clifford Algebra to Geometric Calculus*. Kluwer, 1984.

[11] Ming Jiang, Raghu Machiraju, and David Thompson. A novel approach to vortex core region detection. In I Navazo D Ebert, P Brunet, editor, *Proceedings of Symposium on Visualization '02*. EIROGRAPHICS-IEEE TCVG, May 2002.

[12] Joan Lasenby, Anthony N. Lasenby, and Chris J.L. Doran. A unified mathematical language for physics and engineering in the 21st century. *Phil. Trans. R. Soc. Lond.*, 358:21–39, 2000.

[13] W Lorenson and H Cline. Marching cubes: A high-resolution 3D surface construction algorithm. *Computer Graphics (Proc. of Siggraph)*, 21(4):163–169, July 1987.

[14] Pertti Lounesto. *Clifford Algebras and Spinors*. Cambridge University Press, Cambridge, UK, 1997.

[15] Greg Nielson, Il-Hong Jung, and Junwon Sung. Haar wavelets over triangular domains with applications to multiresolution models for flow over a sphere. In *Proceedings of Visualization '97*. ACM Press, 1997.

[16] A. Perry and M.S. Chong. A description of eddying motions and flow patterns using critical point concepts. *Annual Review of Fluid Mechanics*, 19:125–155, 1987.

[17] Gerik Scheuermann. *Topological Vector Field Visualization with Clifford Algebra*. PhD thesis, University of Kaiserslautern, 2000.

[18] Gerik Scheuermann, Hans Hagen, Heinz Krger, Martin Menzel, and Alyn Rockwood. Visualization of higher order singularities in vector fields. pages 67–74. IEEE Computer Society, 1997.

[19] Gerik Scheuermann, Hans Hagen, Heinz Krger, and Alyn Rockwood. Visualizing critical points of arbitrary poincare-index. In F. Post H. Hagen, G. Nielson, editor, *Scientific Visualization - Dagstuhl '97*, pages 277–283. EEE Computer Society, 2000.

[20] Issac Trotts, David Kenwright, and Robert Haimes. Critical points at infinity: a missing link in vector field topology. In *NSF/DoE Lake Tahoe Workshop on Hierarhical Approximation and Geometrical Methods for Scientific Visualization*, October 2000.