Vortex Detection in Vector Fields Using Geometric Algebra

Stuart Pollock and Stephen Mann

Abstract We present a new method of detecting vortices in sampled vector fields by using Geometric Algebra, and tested three swirl-plane estimation methods for use within our algorithm. Our vortex-detection algorithm recursively looks at vector samples and, via an application of the two-dimensional version of the Gauss-Bonnet Theorem, extracts vortex cores.

1 Introduction

The visualisation of numerical fluid-mechanical and electro-dynamical simulations can be characterised by visualisation of their vortices. Loosely speaking, a vortex in a three-dimensional vector field is a curve around which the field's vectors appear to swirl. Learning where vortices appear in a vector field is important, and often critical. For example, in aerodynamics, vortices located in the wake of an object moving at a high velocity may cause structural damage to trailing objects.

Throughout the last three decades, much work has been done in vector field visualisation and vortex detection. One class of solutions involves dividing a set of points into small polyhedra and then determining whether the vortex core passes through a plane (roughly perpendicular to the vortex core) inside the polyhedron. Within this class of methods, the issues are: how to find such a plane; and how to find the vortex core in the plane.

In this paper, we review and compare three methods of finding such a plane: by using the Jacobian of the vector field; by using the curl of the vector field; and by

Stuart Pollock
Xtreme Labs Inc., 69 Yonge St. Suite 600 Toronto, Ontario M5E 1K3 CANADA e-mail: scepollo@alumni.uwaterloo.ca

Stephen Mann
University of Waterloo, 200 University Ave W, Waterloo, ON N2L 3G1 CANADA e-mail: smann@uwaterloo.ca

using the 'λ_2 algorithm'. We then adapt a Geometric-Algebraic method for finding singularities in the plane to aid in finding vortices passing through the swirl plane. We apply a recursive algorithm to localise the vortex detection and reduce running time. This method is compared to a recursive version of a related vortex-detection algorithm based on Sperner's lemma. While the results of the two algorithms are similar, our GA-based algorithm missed fewer cells containing the vortex than the Sperner based algorithm in the examples we tested.

2 Background and Related Work

A large amount of work has been done on vortices and on detecting vortices. We limit our discussion here to the work most closely related to ours.

The question of "what is a vortex" is not easily answered. Traditionally, physicists have used the magnitude of the curl of a vector field, which is twice the quantity known as the vorticity, at a sample point to perform vortex detection. However, Portela notes [11] that naïvely using the curl to find a vortex core may yield false-positive results. More generally, this means that point-wise algorithms using the curl of a vector field exclusively do not work for vortices with cores that are surfaces, and are not simple curves. A non-zero curl is a necessary condition for a vortex, but it is not sufficient.

Sujudi and Haimes [15] approximated the vector field's Jacobian at a point by taking the linear term of the Taylor series expansion. This tensor may have a single real eigenvalue and a pair of (non-real) complex-conjugate eigenvalues. The two complex-conjugate eigenvalues of a linear system correspond to a plane in which there is a swirl of some sort. Closed streamlines correspond to complex-conjugate eigenvalues that are purely imaginary. If the real part of the complex-conjugate eigenvalue pair is greater than zero, then the singularity is a source; otherwise, it is a sink (see Figure 1). Any of these singularities corresponds to a vortex core by Portela's definition. Sujudi and Haimes applied this idea to a tetrahedralisation of the sample space and processed each tetrahedral cell independently of the others.

Jeong and Hussain [7] gave an algorithm in which they claim that the swirling plane for a vortex corresponds to the plane defined by the eigenvectors correspond-

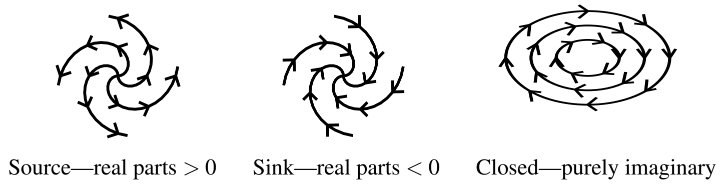

Fig. 1 Three types of streamlines representing different values of the pair of complex-conjugate Eigenvalues of the vector field's Jacobian.

ing to a pair of negative eigenvalues of $S^2 + \Omega^2$, where S and Ω are the symmetric and anti-symmetric parts of the Jacobian, respectively. Because eigenvalues are often listed as λ_1, λ_2, etc., and the two negative eigenvalues indicate a plane of swirling, the algorithm is sometimes referred to as the λ_2 algorithm. Portela pointed out that this method fails in some conditions, and showed why it works well in other conditions [11].

Banks and Singer [1] gave a predictor-corrector method to detect vortex cores in a velocity field for a fluid via a vector-predictor and a pressure-corrector. By starting at a point in the velocity field, a stream-curve is traced in space by stepping from point to point based on the interpolated sample velocity and corrected by the fluid's local pressure. Many such streams are mapped in the field to find the vortices.

Roth and Peikert [13] devised a method based on higher-order partial derivatives to do vortex detection in their standard problem, the bent helical vortex.

Jiang et al. [8] introduced a new geometric algorithm to find vortices: their algorithm employs Sujudi's and Haimes's [15] Jacobian-based characterisation to extract the real eigenvector in a small neighbourhood around the centre of a sampled volume, then uses ideas from Sperner's Lemma to find vortices. The geometric algorithm is inexpensive; however, it sometimes falsely detects vortices in regions without vortices, and sometimes misses vortices in other regions. Like Sujudi's and Haimes's algorithm, this algorithm is not recursive: all sampled cells are checked for vortices.

2.1 Geometric Algebra

We shall present an algorithm based on Geometric Algebra to perform vortex detection, and present here the basics of Geometric Algebra required to understand our results. For a more complete introduction, see [4, 5, 2].

A Geometric Algebra (G.A.) is an oriented, co-ordinate-free algebra defined over R^n, which has 2^n unit basis elements of different dimensionality. The basis elements are oriented k-dimensional subspaces of R^n, although a particular element of the algebra may have components of differing dimensionality.

In Geometric Algebra, the dimensionality of each subspatial element is called its '*grade*'. Thus, scalars are '*grade*-0' elements or '0-*blades*', vectors are '*grade*-1' elements or '1-*blades*', and so forth. Without loss of generality, we shall refer to the orthonormal basis unit vectors of R^n as $\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$, etc.

The exterior product, denoted '$\wedge$' is a grade-raising operation. Given two linearly-independent vectors $\mathbf{a}$ and $\mathbf{b}$, each of which is a grade-1 subspace of R^n, the quantity $\mathbf{a} \wedge \mathbf{b}$ is a 2-blade. The exterior product is anti-symmetric: for two vectors $\mathbf{a} = a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + a_3\mathbf{e}_3$ and $\mathbf{b} = b_1\mathbf{e}_1 + b_2\mathbf{e}_2 + b_3\mathbf{e}_3$ in R^3, $\mathbf{a} \wedge \mathbf{b} = -\mathbf{b} \wedge \mathbf{a} = (a_2b_3 - a_3b_2)\mathbf{e}_2 \wedge \mathbf{e}_3 + (a_3b_1 - a_1b_3)\mathbf{e}_3 \wedge \mathbf{e}_1 + (a_1b_2 - a_2b_1)\mathbf{e}_1 \wedge \mathbf{e}_2$. The magnitude of this product is $|a||b|\sin\theta$, where θ is the angle between the vectors. Geometrically, the magnitude is the area of the parallelogram defined by the vectors.

One set of basis elements for a Geometric Algebra built on R^3 is $\{1, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_2 \wedge \mathbf{e}_3, \mathbf{e}_3 \wedge \mathbf{e}_1, \mathbf{e}_1 \wedge \mathbf{e}_2, \mathbf{e}_1 \wedge \mathbf{e}_2 \wedge \mathbf{e}_3\}$. The first element is the unit scalar, the next three elements are 1-blades, followed by three 2-blades, with the final basis element being an oriented volumetric element (a 3-blade). The highest-grade element in a G.A. defined on R^n is called the pseudoscalar, and is denoted I_n.

2.2 Singularity Detection

Mann and Rockwood presented a G.A.-based algorithm for detecting singularities [9]. Their algorithm is a discrete application of the G.A. version of the Gauss-Bonnet theorem, the continuous version of which was presented for Geometric Algebra by Hestenes and Sobczyk [6]. The algorithm works in any n-dimensional space in which a Geometric Algebra resides but within the context of this paper, only the two-dimensional version of this algorithm is required.

Theorem 1. *Let I_2 be the pseudoscalar (unit-area element), let f be a vector-valued function defined on R^2, let V be the vector field defined by f, and let C be a closed, continuous, piecewise differentiable curve in the domain of f. If the total index of singularities of f contained within the boundary C is m, then*

$$\oint_C \frac{\mathbf{V} \wedge dV}{|\mathbf{V}|^2} = 2mI_2\pi. \tag{1}$$

This integral may be approximated numerically. Given vector samples w_i at points p_i lying on a closed planar curve C in R^3, the w_i may be projected onto the plane in which C lies and normalised to yield unit vectors $\hat{v}_i$. If, for n samples, one assigns a new sample, $p_n = p_0$ and $\hat{v}_n = \hat{v}_0$, then the integral may be approximated discretely as

$$S = \sum_{i=1}^{n} \hat{v}_{i-1} \wedge \hat{v}_i. \tag{2}$$

If n is large enough then $|S|$ is approximately $2m\pi$. The two-dimensional version of Theorem 1 applies not only to R^2, but also to any plane in R^n.

Geometrically, the normalised vectors are mapped onto the unit circle and the areas of half-parallelograms defined by consecutive vectors are summed. Half of the magnitude of the wedge product for each consecutive pair of vectors corresponds to one of the coloured triangles in Figure 2. The m corresponds to the number of times the vectors go around the unit circle for each circuit of C, the factor 2 corresponds to dividing each parallelogram's area by 2 to obtain the area of a triangle, and π corresponds to the area of the unit circle.

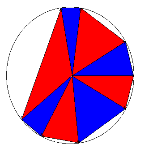

Fig. 2 An example of the discrete approximation to (1), illustrating (2) for a simple vector field containing a single singularity in the sampled region.

3 A G.A. Vortex Core Detection Algorithm

Researchers have visualized vortices in many different ways. One class of solutions involves dividing a set of points, at which vector samples are taken, into small polyhedra and finding for each polyhedron a plane roughly perpendicular to the vortex core, and then determining whether the vortex core passes through the plane inside the polyhedron. Within this class of methods, the issues are: how to find such a plane; and how to find the vortex core in the plane.

We compare three methods of finding such a plane. We also adapt the Mann-Rockwood Singularity-Detection Algorithm to find a vortex in the plane using Geometric Algebra. We apply a recursive algorithm to localise the vortex detection, thereby reducing the running time by two orders of magnitude (Table 3). This method is compared to a recursive adaptation of Jiang's et al.'s vortex-detection algorithm.

Our algorithm is deterministic, based only on the data samples provided. Unlike Sujudi's and Haimes's algorithm and Jiang's et al.'s algorithm, our algorithm is recursive. Unlike Banks's and Singer's algorithm, our algorithm does not step from cube to cube, but rather proceeds in an exclusively top-down fashion.

Our recursive vortex-core-detection algorithm (Figure 3) begins by looking at a cube of vector samples in space and determining whether a vortex core passes through the cube. If a vortex core is detected then, if our algorithm has not reached the bottom level of recursion, the cube is divided into eight octree sub-cubes and our algorithm is executed on each of the sub-cubes. At the bottom level of recursion, if a vortex core has been detected for this cube, then the cube is marked as having a vortex.

The main details missing from the code in Figure 3 are how to compute the swirl plane, and how to detect the vortex passing through this plane. We discuss these issues in Sections 3.1 and 3.2. This basic algorithm has several problems, which we describe in Section 4, along with modifications to improve the algorithm.

```
FindVortex(Ca[],depth)
  Foreach cube C in Ca
    Compute swirl plane Pl
    Project 3-D field onto Pl
    if vortex passes through Pl in C then
      if depth=max then
         Mark C
         return
      else
         Cs[] = 8to1split(C)
         FindVortices(Cs[],depth+1)
```

Fig. 3 Pseudo-code of our recursive vortex detection algorithm.

3.1 Calculating The 'Swirl' Plane

We have investigated three ways to estimate a plane in an octree cube perpendicular to and through which a vortex core may pass. The algorithms are one based on the sole real eigenvector of the Jacobian, one based on the curl, and the λ_2 algorithm. A basic assumption for detecting a vortex core by using any of these methods is that the partial derivatives of the vector field are constant throughout the current octree cube.

The curl and the Jacobian of a vector field each contain first-order partial derivatives of the field. The partial derivatives are estimated using finite differencing on points at the centres of the faces of the octree cube on which the vortex core detection is performed.

Jacobian: If the Jacobian has exactly one real eigenvalue and two complex-conjugate eigenvalues, then the plane perpendicular to the real eigenvector corresponds to a plane with swirling [15]. The real eigenvector of the system is thus tangent to the vortex core at the singularity. In our implementation of this algorithm, if the complex portion of any value, including complex-conjugate eigenvalues, is less than $\varepsilon = 10^{-6}$, then the value is considered to be wholly real.

Our algorithm considers a system with a real eigenvalue λ with a multiplicity of more than one **not** to have any swirling (in this case, the eigenvalue is actually a pair of complex-conjugate eigenvalues in which the complex portion is zero).

Curl: Traditionally, physicists have used the local curl of a vector field for vortex detection. If the curl has a high enough magnitude, then it is considered to be part of a vortex core. Thus, for our algorithm, any octree cube with a non-zero curl at its centre is a good candidate for either the Sperner's-Lemma-based algorithm or the G.A.-based algorithm.

λ_2: In 1995, Jeong and Hussain [7] claim that the swirling plane for a vortex corresponds to the plane defined by the eigenvectors corresponding to a pair of negative eigenvalues of $S^2 + \Omega^2$, where S and Ω are the symmetric and anti-symmetric parts of the Jacobian, respectively. Our implementation of the λ_2 algorithm is based on the contents of the literature review in Roth's and Peikert's work [12].

3.2 Vortex Core Detection

After the swirl plane is determined, points lying in the intersection of the swirl plane and the cube's faces are sampled, and the vector samples are projected onto the plane and normalised. The two-dimensional version of the Mann-Rockwood Singularity-Detection Algorithm is then applied to the vectors at the points on the curve.

We computed the intersection of the swirl plane with the cube by using the *meet* of the swirl plane with each face of the cube [14], which produces a sequence of line segments. We sampled points on these line segments linearly, and sampled the vector field at these points. The sampled vectors are projected onto the swirl plane, and then (2) is applied to the projected vectors to numerically calculate the integral.

One known limitation of our algorithm (and all other such algorithms) is that oppositely rotating vortices might not be detected unless some minimum amount of spatial subdivision is performed in advance, and this minimum subdivision level may only be determined by trial and error on a case-by-case basis.

The Mann-Rockwood two-dimensional G.A.-based singularity detection algorithm is based on finding the area spanned by a sequence of unit vectors on a closed curve. If this area is within $\varepsilon = 10^{-6}$ of zero, then the total index of singularities within the closed curve is zero; this means either that there are no singularities contained within the curve, or that the indices of all the singularities cancel.

In our algorithm, the closed curve is the polygon defined by the intersection of the swirl plane with the cube. Three-dimensional vectors are computed at sample points on the polygon, and then are projected into the swirl plane and normalised, as illustrated in Figure 2, where the vectors at consecutive points on the curve have been mapped onto the unit circle. As in the Mann-Rockwood algorithm, the areas of the triangles defined by consecutive vectors are summed, and the result of that sum indicates how many singularities of what sign are within the curve.

To gauge the usefulness of our G.A.-based algorithm, we also implemented a variant of the Sperner's-Lemma-based algorithm described by Jiang et al. [8]. Vector samples at the centres of the faces of the cube are projected into the swirl plane. Their version of the algorithm steps around the polygon defined by the intersection of the cube with the plane and examines each of the sample vectors in turn, marking each with the quadrant in which it lies. If a complete circuit of the quadrants is made, either clockwise or counter-clockwise, then a vortex core goes through the sample cube.

4 Results

We compared our G.A. vortex-detection algorithm and a recursive version of the Sperner's-Lemma-based algorithm for both speed and accuracy. The computations compared herein were performed on a 2.0 GHz Pentium 4 with 2GB RAM, running Debian Linux. To reduce variation in timing, each test was run ten times and averaged. The Geometric Algebra package used has three- and four-dimensional algebras generated using Gaigen v0.982. Each singularity-check of the polygon resulting from intersecting an octree cube with its swirl plane involved approximately 48 samples, with each adaptive step adding eight more samples on each line segment in which refinement was needed.

We tested our algorithm with two vortices: a straight-line vortex, which has been rotated so as not to be aligned with a co-ordinate axis, and the bent helical vortex. All of the partial derivatives have been approximated using finite differencing.

A minimum level and a maximum level of recursion was used in each test case. The maximum level in each case was 6. A minimum level of recursion of 3, for example, corresponds to subdividing the samples cube three times to obtain 8^3 cubes to be examined recursively (up to another three levels each).

Our initial tests with a rotated, straight-line helical vortex demonstrated that our algorithm could find a simple vortex core. For a more rigorous test, we used the bent helical vortex [12], which is defined as

$$\mathbf{V} = (-\frac{\gamma y}{r} - \frac{\omega z x}{r^2}, \frac{\gamma x}{r} - \frac{\omega z y}{r^2}, \omega(1-\frac{R}{r})), \quad r = \sqrt{x^2+y^2}, \tag{3}$$

with $R = 1.0$, $\gamma = 0.88$, and $\omega = 1.0$. The grid was centred at $[0.05, 0.05, 0.05]$.

From Table 1, we see that from a timing perspective, the Jacobian-based algorithm has the best timing with no recursion. For a minimum level of recursion of 0, the λ_2 algorithm takes longer to find a solution than either the curl or the Jacobian. The main reason why the λ_2 algorithm takes much longer than the Jacobian algorithm is that using the Jacobian causes our algorithm not to find swirl planes in the octree cubes, thus reducing computing time by computing a worse result. One can see by looking at Table 2 that, for a minimum level of recursion of 0, the number of swirl planes detected in the recursion while using the Jacobian is much lower than for the other algorithms.

Table 1 Running Times For Bent Helical Vortex, in seconds.

Singularity		Minimum Recursive Depth		
Detector	Swirl	0	3	6
Sperner	Jacobian	2.236	5.040	977.245
Sperner	curl	4.521	5.995	955.056
Sperner	λ_2	5.134	6.423	803.483
G.A.	Jacobian	3.025	5.322	962.140
G.A.	curl	4.831	6.428	930.339
G.A.	λ_2	5.962	7.312	892.704

The larger the minimum level of recursion in our algorithm, the better the λ_2 algorithm performs against its competitors (see Table 2) in terms of speed. The number of times any of the singularity-detection algorithms is called was discovered to be the source of this improvement, indicating that the λ_2 algorithm for determining the plane of swirl is a more accurate method than using either the curl or the Jacobian of the vector field.

Table 2 Number Of Calls To Singularity-Detection Algorithms For Bent Helical Vortex.

Singularity		Minimum Recursive Depth		
Detector	Swirl	0	3	6
Sperner	Jacobian	553	1232	259738
Sperner	curl	1121	1480	262144
Sperner	λ_2	1241	1543	215555
G.A.	Jacobian	705	1232	259738
G.A.	curl	1129	1504	262144
G.A.	λ_2	1361	1655	215555

For the bent helical vortex, the expected result is a closed circular vortex. Because the samples used in our testing come from an analytic solution, the solution may be plotted at the same time as the approximate solution. In Figures 4 and 5, the analytic solution has been plotted in blue, whereas the approximate solution for each test case is plotted in red.

Neither the Sperner's-Lemma-based algorithm nor the G.A.-based algorithm worked perfectly. Using the curl instead of the Jacobian yielded a circular vortex with a larger radius, which is expected due to the results of Roth and Peikert [12]. The λ_2 algorithm yields what seems to be the same curve generated by the curl-based algorithm, yet has fewer gaps with any minimum level of recursion than the curl-based algorithm with the same minimum level of recursion.

While both the Sperner's Lemma algorithm and our G.A. algorithm can find a closed or nearly closed curve, it required starting with a fine grid structure, and made little use of the recursive subdivision to reduce the costs. A further analysis revealed several things.

First, the singularity-detection part of our algorithm may fail for two reasons: there may be too few samples to get a good approximation to (1); and the grid may be aligned so as to make vortex cores pass too close to the faces of octree cubes. To circumvent the first problem, we used extra vector samples to better approximate the integral in Theorem 1. However, this resulted in little improvement.

Looking at the octree in regions where the vortical curve is not closed, we found places where the detection fails when the singularity is close to a cube's face (e.g., the empty octree cubes in the red curve's gaps, Figure 6). Our algorithm was failing because the computed swirl plane is only *close* to perpendicular to the vortex core. The problem occurs when the vortex core (as determined by the swirl plane extraction algorithm) passes close to the boundary of the current octree cube, then the discrepancy between the estimated swirl plane and an actual swirl plane was enough that the vortex core passed through the detected swirl plane outside the octree cell (Figure 7). If this happened in both of two adjacent cells, then a gap would appear in our vortex core.

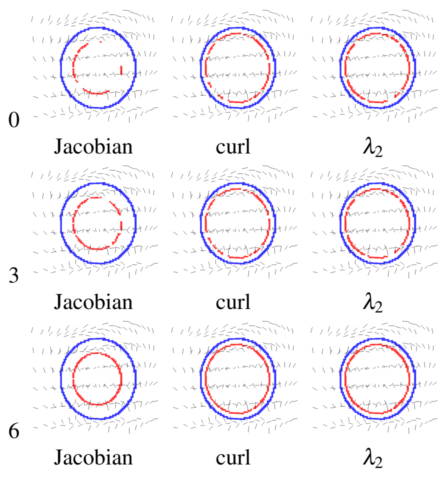

Fig. 4 Sperner's Lemma with minimum recursion of 0,3,6

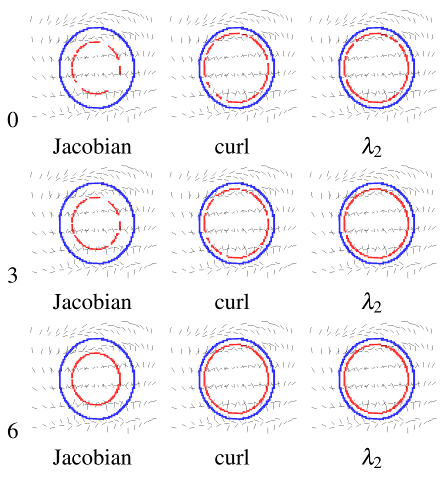

Fig. 5 Geometric Algebra with minimum recursion of 0,3,6

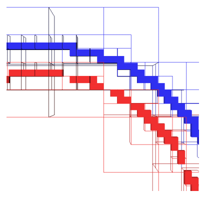

Fig. 6 The failure of the singularity detection

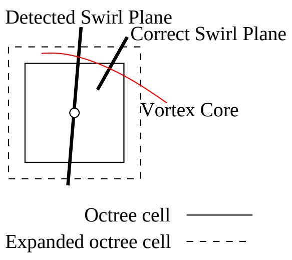

Fig. 7 Failed detection and solution

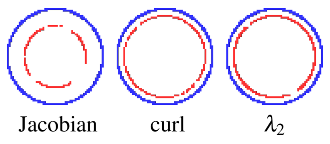

Fig. 8 Sperner's Lemma extended octree cells; minimum recursion of 0. Compare to Fig. 4.

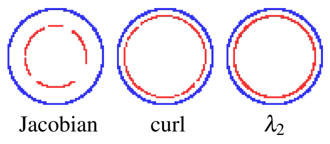

Fig. 9 G.A. extended octree cells; minimum recursion of 0. Compare to Fig. 5.

To overcome this problem, rather than intersect the detected swirl plane with the actual octree cell, we intersected the swirl plane with a slightly larger octree cell. This change to the algorithm resulted in significant improvements. As shown in Figure 9, using an octree scaling factor of 1.25 and a minimum recursion level of 0, we obtain a closed curve for the Geometric Algebra algorithm using the λ_2 swirl plane extraction method, a nearly closed curve for the curl swirl plane extraction method, with only slight improvements in the Jacobian swirl plane extraction method. Scaling the octree cubes also results in similar improvements in the Sperner's Lemma algorithm (Figure 8).

By increasing the size of each octree cube by 25 percent, the time to execute the algorithm roughly doubled. This is expected as the algorithm detects a vortex core in more cubes. However, a small number of false positives, approximately 5 percent of the original number of cubes, was added to the detected curve, mostly in places where the detection of the curve was initially strong.

Table 3 Running Times For Bent Helical Vortex, in seconds, using the λ_2 vortex characterisation.

Singularity	Minimum Recursive Depth	
Detector	0	6
Sperner	0.783	133.327
G.A.	1.074	154.012

Singularity	Normalised cost (ms/cube)	
Detector	0	6
Sperner	0.539	0.619
G.A.	0.692	0.714

The left half of Table 3 give absolute running timings of our algorithm using an increased cube size, not accounting for the number of octree cubes being processed. When the timings are normalised by the number of cubes processed, the relative performance of the G.A. algorithm improves relative to the Sperner's-Lemma algorithm.

We provide here a brief summary of the length of the source code and in-line comments for various steps in our algorithm. Rather than implement our own G.A. package, we used Gaigen (the Geometric Algebra Implementation Generator developed by Fontijne et al. [3]) to generate three- and four-dimensional Geometric Algebras for use with our implementation.

The code for calculating the swirl plane is approximately 350 lines long, approximately 230 lines of which is eigenvector-extraction code. The rest of that code is support code for approximating partial derivatives and determining which swirl-plane-extraction algorithm to use.

The code for calculating the intersection of the swirl plane with the edges of a cube is approximately 65 lines long. The code for organising the lines segments of the intersection polygon and calculating the sample points is approximately 90 lines long.

The code for the Sperner's-Lemma-based singularity-detection algorithm is approximately 80 lines long. In contrast, the G.A.-based algorithm is 10 lines long, including comments and function header.

5 Conclusions

We have presented a recursive algorithm for detecting vortices in sampled vector fields, the mathematics for which have been simplified by the use of Geometric Algebra. We have shown that the two-dimensional version of the Mann-Rockwood Singularity-Detection Algorithm, coupled with the use of the λ_2-swirl-plane-extraction algorithm, yields more detected vortex core segments than the algorithm of Jiang et al. Plainly, from the images generated and the processing time, the λ_2 plane-extraction algorithm was the best of the three swirl plane extraction methods we considered.

Using the λ_2 algorithm or the curl has allowed both the G.A.-based algorithm and the Sperner's-Lemma-based algorithm to work quite well on a recursive basis, rather than requiring too much time by processing data for finer spatial subdivisions.

By increasing by 25 percent the size of the octree cube from which our samples are taken, we have also been able to detect a closed curve when using the λ_2 algorithm with a minimum recursion level of 0.

For additional details on this work, see [10].

References

1. D. C. Banks and B. A. Singer. A predictor-corrector technique for visualizing unsteady flow. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):151–163, 1995.
2. Eduardo Bayro-Corrochano. *Geometric Computing: for Wavelet Transforms, Robot Vision, Learning, Control and Action*. Springer, 2010.
3. T. Bouma D. Fontijne and L. Dorst. Gaigen: Geometric algebra implementation generator, 2002.
4. Leo Dorst, Daniel Fontijne, and Stephen Mann. *Geometric Algebra for Computer Science*. Morgan-Kaufmann, 2007.
5. D. Hestenes. *Space–Time Algebra*. Gordon And Breach, 1966.
6. D. Hestenes and G. Sobczyk. *Clifford Algebra To Geometric Calculus*. Kluwer, 1984.
7. J. Jeong and R. Hussain. On the identification of a vortex. *Journal Of Fluid Mechanics*, 285:69–94, 1995.
8. R. Machiraju M. Jiang and D. Thompson. *A novel approach to vortex core region detection*. IEEE TCVG Symposium On Visualization, 2002.
9. S. Mann and A. Rockwood. Computing singularities of 3D vector fields with geometric algebra. In *Proceedings of IEEE Visualization 2002*, pages 283–289, 2002.
10. Stuart Pollock. Vortex detection in vector fields using geometric algebra. Master's thesis, University of Waterloo, 2003.
11. L. M. Portela. *Identification and Characterization of Vortices in the Turbulent Boundary Layer*. PhD thesis, Stanford, 1997.
12. M. Roth and R. Peikert. Flow visualization for turbomachinery design. In *Proceedings of IEEE Visualization '96*, pages 381–384, 1996.
13. M. Roth and R. Peikert. A higher-order method for finding vortex core lines. In *Proceedings of IEEE Visualization '98*, pages 143–150, 1998.
14. L. Dorst S. Mann and T. Bouma. *The Making of a Geometric Algebra Package in Matlab*. Research Report CS-99-27, University Of Waterloo, 1999.
15. D. Sujudi and R. Haimes. Identification of swirling flow in 3D vector fields. In *AIAA 12th Computational Fluid Dynamics Conference*, 1995.