Interactive analysis of the topology of 4D vector fields

by R. R. Dickinson

Interactive visualization methods are now evolving in response to a need to provide more immediate access to particular features of interest to analysts at particular points in the space and time of their data. This paper focuses on feature extraction methods relevant to the analysis of vector fields. In vector fields, "critical points" are those points at which the vector magnitude passes through zero. The word "topology" is used to describe the interconnection patterns between critical points. Topology is central to the understanding of vector fields. It provides very succinct and precise summary information, and can be used to subdivide large fields into well-defined subregions. In this paper, methods for interactively creating maps of vector-field topology are described. The advantages offered by interactive methods in comparison with automatic methods are also discussed.

Introduction

Much visualization research and system development in recent years has focused on the automatic creation of large quantities of pictorial information to represent the end results of analysis. This trend is rooted in an implicit assumption that visualization happens at the end of the computing component of a given project. As a result, a large amount of computing resources may be spent on producing images in which the user might only have a passing interest. This typically takes the form of a videotape archive. Once a tape has been made, there is no way of exploring different spatial regions of the underlying data from different viewpoints without restarting the videotape production cycle from scratch. Further, with currently available video playback equipment, there are limits on how quickly and easily a scientist or engineer can obtain information about a particular region of the time domain of a given problem. And with the implicitly fixed number of output frames that can appear in a video recording over a given portion of analysis time, there is no guarantee that a feature of particularly short duration will appear on the tape at all.

If the purpose of computing is insight, a feedback loop is clearly implied. In turn, visualization should be regarded not only as the climax of a process of enlightenment, but also as the beginning of a new analysis cycle. More immediate access to particular features of interest to analysts at particular points in the space and time of their data is clearly needed. Interactive visualization methods are now evolving in response to this need. This paper discusses some algorithms and user interface issues relevant to the interactive visualization of vector fields. When the accompanying videotape is viewed, it should be remembered that each scene was

recorded in real time. The narration explains what the user is doing with a mouse and a Spaceball* to drive the topology extraction process. In some cases, the user has chosen to trade off image quality and density with interactivity in order to focus interest on the particular phenomenon that is being demonstrated.

In this paper, the term *vector-field* refers to a process which associates a vector-valued quantity with each point in a region of space and time. The most common practical examples of fields that are fundamentally vector-valued (as opposed to derived vector fields) are the fluid velocity fields arising from computational fluid dynamics analyses and experimental fluid mechanics research. Other common applications are heat flows in heat transfer problems and magnetic flux in magnetics. Examples of derived vector fields of substantial interest to meteorologists are the gradients of scalar fields such as atmospheric pressure and temperature. The examples given in this paper are primarily drawn from these applications, but the theory is quite general.

The most common methods for visualizing vector fields typically involve stream-line tracking and particle traces. Several variations on these themes have appeared recently. For example, tubes have been used to more clearly communicate the 3D form of complex space curves arising from 3D fluid flows [1, 2]. "Strokes" (based on the head-to-tail shapes of pen strokes) are being used to reduce the clutter associated with the more traditional arrowheads for denoting vector direction [1, 3]. For particle traces, paper jets are being used as alternatives to the more traditional bubble shapes in order to show more clearly the torsion of complex space curves [2]. A method that uses surfaces rather than curves has also been developed [4]; this is based on the displacement of a cutting plane in the direction of the vector field at all points of the cutting plane. The magnitude of the displacement is everywhere proportional to the corresponding vector magnitude. This method provides an important alternative to curve-based approaches. But substantial interpretation difficulties are likely to be encountered when this method is used in the vicinity of singularities due to the rapid changes in velocity magnitude that typically accompany such features.

These traditional methods can lead to excessive visual clutter, especially if automatic graphics object creation methods are used. In fact, in many books and articles some of the most useful images are often schematic diagrams drawn by artists to highlight particular features that are known to exist but that are difficult to draw automatically. (For a good example of this, see Figure 5.1 of the notes from a recent SIGGRAPH course [5].) In contrast to this sort of clutter, vector-field topology provides a very succinct and precise summary of a given vector field. Topology describes the interconnection pattern of "critical points"—points at which the vector value passes through zero. In many applications, vector-field topology is central to the understanding of the underlying processes. Topology can provide both graphics programmers and users with ways of subdividing large and complex fields into well-defined and more easily understood regions. For example, topology edges can be used to define geometric boundaries within which tangent curves are constrained to lie. Structuring these into groups that the user can switch on and off according to regional interests follows in a natural way.

While the formal development of a method for automatically analyzing vector-field topology appeared quite recently [6, 7], the underlying mathematical components required for analyzing 2D vector-field topology have been available in books for some time [8]. In this paper we describe the implementation of these established mathematical concepts in an interactive system designed for the identification of selected parts of topology near given points in the space and time of a vector field.

2D vector-field topology

While 2D vector fields can be thought of as special cases of the 4D fields dealt with later in this paper, it is useful to begin with steady-state bivariate, bivalued vector fields,

$$F(u, v) = \begin{bmatrix} f_0(u, v) \\ f_1(u, v) \end{bmatrix}, \tag{1}$$

where f_0 and f_1 are the components of the vector value F, and u and v represent the field domain. Note that F can be either fundamentally vector-valued, representing a velocity field, for example; or it can be derived from some other function such as the gradient of a 2D pressure field. But for the purposes of what follows, we need only be concerned with the values of F itself, along with its derivatives with respect to u and v.

With respect to first-order derivatives, critical points can be classified according to the eigenvalues of the Jacobian of the vector value [7, 8],

$$J(u_0, v_0) = \begin{bmatrix} \frac{\partial f_0}{\partial u} & \frac{\partial f_0}{\partial v} \\ \frac{\partial f_1}{\partial u} & \frac{\partial f_1}{\partial v} \end{bmatrix}, \tag{2}$$

where u_0, v_0 is a point in the domain of F such that

$$F(u_0, v_0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{3}$$

In this paper the eigenvalues are written as

* Spaceball is a trademark of Spatial Systems Inc., Billerica, MA.

Table 1 Classification of critical points.

Name	λ_{r0}	λ_{r1}	λ_i
Saddle point	<0	>0	$=0$
Attracting node	<0	<0	$=0$
Repelling node	>0	>0	$=0$
Attracting focus	<0	<0	$\neq 0$
Repelling focus	>0	>0	$\neq 0$
Center	$=0$	$=0$	$\neq 0$

$$\boldsymbol{\lambda}_0 = \lambda_{r0} + \lambda_i \mathbf{i} \tag{4a}$$

and

$$\boldsymbol{\lambda}_1 = \lambda_{r1} - \lambda_i \mathbf{i}, \tag{4b}$$

using the subscripts r and i to distinguish between the real and imaginary parts of each eigenvalue. The classification of critical points using this notation is given in **Table 1** for completeness.

For the purposes of interactive topology extraction, the most important of these is the saddle point. At each saddle point, the tangent to the vector field is undefined, but the tangent is well defined some small distance away from each saddle point. Accordingly, there is always a distinct pair of curves that are each tangent to the vector field some small distance away from a given saddle point, as well as being tangent to an eigenvector of the Jacobian at the saddle point. By dividing each of this pair of curves into two distinct curves deemed to begin at the saddle point and head off in a positive and a negative direction, respectively, the user gets to interact with four distinct curves emanating from each saddle point. We use italics here to denote terms used in the *Graph Theory* sense: It is these four curves that form the *edges* connected to the *node* associated with the given saddle point in the *graph* that uniquely defines the topology of the given vector field. In an *oriented graph*, it is a simple matter to ensure consistency of orientation of *edges* by always assigning originating nodes (e.g., saddle points) as the origin of each connected *edge*.

Note that since the tangent is undefined at a saddle point, the assignment of positive and negative direction lies solely with the assignment of the direction of the eigenvector. This is nontrivial because the direction of a given eigenvector is meaningless mathematically. For 2D steady-state fields it turns out that it can be considered as being arbitrary. For 2D transient fields, we need to reconsider what we mean by arbitrary, as discussed in the next section.

Now, to interactively locate saddle points, and in turn track topology edges, we need to be able to find the nearest saddle point from a given user-specified point, and then track the relevant tangent curve(s) emanating from that saddle point. Ideally, this process and the overhead of interacting with the user should be achievable at speeds approaching screen refresh rates. The following numerical procedure has been found to be quite adequate to achieve this goal on typical graphics workstations.

The following pseudocode gives an outline of the algorithm that we use to locate saddle points:

```
Find_nearby_saddle_point:

Classify_nearby_singularity (Table 1)
if (nearby_singularity is a saddle point)
{
        Step_to_nearby_saddle_point
}
else
{
        Find_nearby_saddle_like_region
        if (found) Step_to_nearby_saddle_point
}
```

The algorithm "*Classify_nearby_singularity*" simply involves taking the Jacobian of the vector field at the (potentially arbitrary) user-specified point, and classifying the form of the field at that point using the criteria in Table 1. In doing this, we implicitly assume that first-order information is generally sufficient to predict the form of the nearest singularity to the user-specified point. Since we are dealing with an interactive system, the consequences of this assumption being incorrect are trivial because the user is able to observe the behavior of the underlying algorithms at all times, and can simply move the user-specified point if the behavior is not what was expected.

If the class of the nearby singularity is a saddle point, then stepping on to it simply consists of the following first-order iterative process. At any given point in the vicinity of the saddle point, a first-order estimate of its location is given by

$$\delta u = \left(f_1 \frac{\partial f_0}{\partial v} - f_0 \frac{\partial f_1}{\partial v} \right) \Big/ \Delta \tag{5a}$$

and

$$\delta v = \left(f_0 \frac{\partial f_1}{\partial u} - f_1 \frac{\partial f_0}{\partial u} \right) \Big/ \Delta, \tag{5b}$$

where

$$\Delta = \frac{\partial f_0}{\partial u}\frac{\partial f_1}{\partial v} - \frac{\partial f_0}{\partial v}\frac{\partial f_1}{\partial u}. \tag{5c}$$

By iteratively stepping a distance δu in u and δv in v, and re-evaluating the Jacobian at the new point, finding the precise location of the saddle point to within some tangent-magnitude and/or location tolerance follows in the obvious manner.

If the class of the singularity near a given user-specified point is not a saddle point, we obviously need to take some other course of action. *Find_nearby_saddle_like_region* takes care of this. The objective is to move out of the region occupied by the user-specified point, into a region that is likely to contain a saddle point. If the user-specified point is deemed to be near a node or a focus, we simply march along the tangent curve passing through that point in the direction away from the node or focus, in search of a saddle-like region. If the user-specified point is deemed to be near a center, indicating that the tangent curves in this area are nearly circular or elliptical, we use the geometry of the osculating circle of the tangent curve passing through respective points in an iterative process that is designed to quickly move directly away from the nearby singularity, in search of a saddle-like region. The geometry of the osculating circle to the tangent curve passing through a given point is given by the curl of the unit tangent vector:

$$\text{curl}(\hat{\mathbf{t}}) = \left[\left(-f_0 f_1 \frac{\partial f_0}{\partial u} - f_1^2 \frac{\partial f_0}{\partial v} + f_0^2 \frac{\partial f_1}{\partial u} + f_0 f_1 \frac{\partial f_1}{\partial v}\right) \Big/ |F|^3\right]\mathbf{k}. \quad (6)$$

The magnitude of the inverse of the coefficient of **k** gives the radius of the osculating circle, and its sign indicates the side of the tangent curve on which the center of the circle is located.

Given either of the above initial states, if the saddle-point-searching algorithm ends up at a point outside the domain of the field, our system currently does nothing visually, returning directly to the input event loop to wait for the user to move to a new point from which to try again.

Interaction paradigms

For the remainder of this section, we assume that nearby saddle points are locatable at speeds approaching or faster than screen refresh rates, and that a feedback line from the user-specified point to the saddle point is always displayed in a rubber-band-like manner. All that remains is to track the four curves emanating from the currently found saddle point and display them as temporary curves for the user to preview. Our system simply interprets a mouse click as an indication that the found edges are of interest, and saves them as permanent graphic structures.

A simple but very useful extension of this paradigm involves an edge search menu with the following choices:

- All four edges.
- Adjacent edges.
- Largest eigenvalue, positive sense.
- Largest eigenvalue, negative sense.
- Smallest eigenvalue, positive sense.
- Smallest eigenvalue, negative sense.

The second choice is the most intuitively satisfying one. With this selection, the particular edge that is oriented closest to the feedback line from the user-specified point to the saddle point is the only one that is tracked, displayed, and subsequently saved. The remaining four can be used to uniquely define any particular one of the four curves from a given point. This is useful for systematically dividing the topology creation process into parts that are smaller and easier to understand.

The opening segment of the accompanying video graphic shows some of these interaction paradigms in action. The first example involves a derived vector field that was produced by taking the gradient of a 2D atmospheric pressure field. The second example describes the physics involved in the transonic analysis of the flow of SF6 through an industrial circuit breaker, and then displays some results of an interactive analysis of the topology of these data.

Transient 2D fields

In this section, the topology extraction concepts described in the previous section are augmented for use with trivariate bivalued vector fields of the form

$$F(u, v, t) = \begin{bmatrix} f_0(u, v, t) \\ f_1(u, v, t) \end{bmatrix}, \quad (7)$$

where f_0 and f_1 are the components of the vector value F, u and v represent the spatial domain of the field, and t represents the time domain of the analysis or data used to define F.

For tracking topology with respect to time, at any given instant along the time domain of F the Jacobian and its eigenvalues can be computed as described above. That is, any given snapshot of the transient field is treated as a steady-state field independently of the values of F at any other point in its time domain. The only difference is that in a sequence of time steps used to animate the evolution of topology with respect to time, we need to ensure that the senses of direction of the four curves emanating from a given saddle point are consistent from one time to the next. This is because we would like to be able to give the user the ability to point to one particular edge at a given instant in time, and then ask the system to display its continuous evolution both before and after the specified time. To do this, we need to ensure that the computation of the sense of direction of the eigenvector associated with a given saddle point is consistent for small changes in the Jacobian associated with that saddle point, from one point in time to the next. We achieve this by symbolically solving the eigensystem of the Jacobian as follows.

For a compact representation of the symbolic operations that follow, we can rewrite the Jacobian at a

given point in space and time as

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{00} & \sigma_{10} \\ \sigma_{01} & \sigma_{11} \end{bmatrix} \tag{8}$$

The eigenvalues of this matrix are

$$\lambda_0 = 0.5[(\sigma_{00} + \sigma_{11}) + A] \tag{9a}$$

and

$$\lambda_1 = 0.5[(\sigma_{00} + \sigma_{11}) - A], \tag{9b}$$

or

$$\lambda_{0,1} = 0.5[(\sigma_{00} + \sigma_{11}) + s_A A], \tag{10}$$

where

$$A = (\sigma_{00}^2 - 2\sigma_{00}\sigma_{11} + \sigma_{11}^2 + 4\sigma_{01}\sigma_{10})^{1/2}$$

and s_A is the sign of A:

$$s_A = \begin{cases} +1 & \text{for } \lambda_0, \\ -1 & \text{for } \lambda_1. \end{cases}$$

Note that for foci, the magnitude of A is the imaginary part of the eigenvalues in Equation (4). The eigenvectors of $\boldsymbol{\sigma}$ are given by the following singular linear system:

$$[\boldsymbol{\sigma}] \cdot \mathbf{x} = \lambda \mathbf{x}, \tag{11a}$$

or, equivalently,

$$\begin{bmatrix} \sigma_{00} - \lambda & \sigma_{01} \\ \sigma_{10} & \sigma_{11} - \lambda \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0. \tag{11b}$$

If we wish to obtain normalized eigenvectors, we can impose the constraint that

$$x_1^2 + x_2^2 = 1. \tag{12}$$

To solve for the eigenvectors corresponding to λ_0 and λ_1 in a way that gives a consistent sign convention for small changes in $\boldsymbol{\sigma}$, the eigenvector corresponding to λ_0 is computed as follows. Solving the top row of (11) subject to (12) gives

$$x_0[0] = sense\ 2\sigma_{01} C^{-1/2} \tag{13a}$$

and

$$x_0[1] = -sense\ B\ C^{-1/2}, \tag{13b}$$

where

$$C = 4\sigma_{01}^2 + B^2,$$

$$B = \sigma_{00} - \sigma_{11} - s_A A,$$

and *sense* is ±1 denoting the sense of direction assigned by the user in the menu items listed in the previous section.

If both σ_{01} and σ_{10} are small relative to the diagonal values, σ_{00} is assigned to λ_0, and the vector $(0, -sense)$ is assigned to $\mathbf{x}$. The assignment of the eigenvector associated with λ_1 follows similarly from the lead given here.

A detailed illustration of interactive transient 2D topology extraction is not included as part of the accompanying video graphic because of space limitations imposed on the video publication.

3D vector-field topology

This section deals with steady-state trivariate, trivalued vector fields,

$$F(u, v, w) = \begin{bmatrix} f_0(u, v, w) \\ f_1(u, v, w) \\ f_2(u, v, w) \end{bmatrix}, \tag{14}$$

where the f_i represent the components of the vector value F, and u, v, and w represent the 3D Cartesian domain of the field.

With respect to first-order derivatives, critical points for 3D vector fields can also be classified according to the Jacobian of the vector value,

$$J(u_0, v_0, w_0) = \begin{bmatrix} \frac{\partial f_0}{\partial u} \frac{\partial f_0}{\partial v} \frac{\partial f_0}{\partial w} \\ \frac{\partial f_1}{\partial u} \frac{\partial f_1}{\partial v} \frac{\partial f_1}{\partial w} \\ \frac{\partial f_2}{\partial u} \frac{\partial f_2}{\partial v} \frac{\partial f_2}{\partial w} \end{bmatrix}, \tag{15}$$

where u_0, v_0, w_0 is a point in the domain of F such that

$$F(u_0, v_0, w_0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{16}$$

We are currently experimenting with a hybrid of symbolic and numerical techniques for solving the eigenvalues of J for the 3D case, using a generalization of the methods required for tracking tensor field lines through symmetric 3D tensor fields [2, 9]. To locate a particular critical point, the following first-order iterative scheme is used from an arbitrary point specified by the user. A first-order expansion of F gives

$$\begin{bmatrix} \frac{\partial f_0}{\partial u} \frac{\partial f_0}{\partial v} \frac{\partial f_0}{\partial w} \\ \frac{\partial f_1}{\partial u} \frac{\partial f_1}{\partial v} \frac{\partial f_1}{\partial w} \\ \frac{\partial f_2}{\partial u} \frac{\partial f_2}{\partial v} \frac{\partial f_2}{\partial w} \end{bmatrix} \cdot \begin{bmatrix} \delta u \\ \delta v \\ \delta w \end{bmatrix} = - \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix}. \tag{17}$$

An unrolled symbolic version of Gaussian elimination has been found to be both fast and robust for solving this system of equations for δu, δv, and δw at each step in an iterative process that searches for the nearest point satisfying Equation (16). As shown in the videotape, this makes interactive searching of critical points quite feasible.

In general, there are six curves emanating from each 3D saddle point, connecting adjacent nodes and foci. This many curves can lead to substantial visual clutter. For this reason it is useful to classify *edges* as well as *nodes* when attempting to extract topology from 3D vector fields. We call one such edge class a "vortex centerline." This involves the restriction that the curl of the vector field be parallel to the vector field itself, everywhere along the corresponding tangent curve. That is, we would like to find the particular curve that satisfies

$$F \times \text{curl}(F) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{18}$$

along its entire length. To find the nearest curve satisfying this condition from an arbitrary point specified by the user, we use the first-order expansion

$$[A] \cdot \begin{bmatrix} \delta u \\ \delta v \\ \delta w \end{bmatrix} = \begin{bmatrix} f_1 C_2 - f_2 C_1 \\ f_2 C_0 - f_0 C_2 \\ f_0 C_1 - f_1 C_0 \end{bmatrix},$$

where the coefficient matrix is computed as follows. We denote $\text{curl}(F) = (C_0, C_1, C_2)$ (i.e., $C_0 = \partial f_1/\partial w - \partial f_2/\partial v$ and so on), and write $\partial f_0/\partial u$ as f_{0u} and so on for the remaining components and derivatives:

$$A_{00} = f_{1u} C_2 - f_{2u} C_1 ,$$
$$A_{01} = f_{1v} C_2 - f_{2v} C_1 ,$$
$$A_{02} = f_{1w} C_2 - f_{2w} C_1 ,$$
$$A_{10} = f_{2u} C_0 - f_{0u} C_2 ,$$
$$A_{11} = f_{2v} C_0 - f_{0v} C_2 ,$$
$$A_{12} = f_{2w} C_0 - f_{0u} C_2 ,$$
$$A_{20} = f_{0u} C_1 - f_{1u} C_0 ,$$
$$A_{21} = f_{0v} C_1 - f_{1v} C_0 ,$$
$$A_{22} = f_{0w} C_1 - f_{1w} C_0 .$$

An unrolled symbolic Gaussian elimination solver is used at each iteration, in a manner analogous to that used for locating point singularities. For locating singular curves from poor initial guess locations, the above coefficient matrix often turns out to be numerically poorly determined until the iteration starts to zero in on the required curve. In such circumstances, we take a section through the 3D field that is perpendicular to the curl of the vector value and that passes through the initial point. At all points in this new 2D field, the 3D vector value is projected onto the plane. The out-of-plane component of the 3D vector value is ignored. In this new "virtual 2D vector field," we perform the same operations that were described earlier for 2D fields, to zero in on the point in this plane at which the vector value vanishes. This entire process is repeated in an iterative manner until the above three-dimensional convergence criterion is satisfied.

Scene 4 of the accompanying videotape shows this curve-searching algorithm in action. The interaction paradigm is similar to that described for saddle-point searching, except that the feedback line points from the user-specified point to the nearest point on the found curve, rather than at a singularity. The usefulness of these curves becomes particularly apparent in attempting to track paper jets in the vicinity of the curve. Such curves are clearly necessary in order to provide a reference feature for the unambiguous interpretation of 3D particle traces. Some examples of a second class of edge that we call a "saddle curve" are also given in the accompanying video graphic, but a detailed mathematical description of the algorithm for tracking this second class of curve is beyond the scope of the present paper.

Transient 3D vector fields

This section deals with quadvariate, trivalued vector fields of the form

$$F(u, v, w, t) = \begin{bmatrix} f_0(u, v, w, t) \\ f_1(u, v, w, t) \\ f_2(u, v, w, t) \end{bmatrix}, \tag{19}$$

where the f_i represent the components of the vector value F; u, v, and w represent the spatial domain; and t represents the time domain of the underlying analysis or data used to define F.

For tracking 3D topology with respect to time, we can again deal with each instant along the time axis as a steady-state problem of the form discussed in the previous section. Scene 5 of the accompanying videotape shows this process in action on the creation and evolution of the vortices in the airflow through a room. At the beginning of analysis time, the air is stagnant, and a forced flow velocity is imposed at the inlet. The movement of the vortices displayed here provides the analyst with essential visual feedback required to meaningfully interpret the flow results, without the excessive visual clutter that typically accompanies groups of particle traces. Note that as the user zooms in on a

given portion of analysis time, more information is displayed in the region of interest, and nothing is displayed outside the time clip limits. Users quickly learn to exploit this capability because it is analogous to the screen space clip limits of a static image.

Our experiences with these tools to date clearly show that transient 2D topology extraction is very much easier for the user than transient 3D topology. Accordingly, it is desirable to be able to help the user even more by being able to search for particular critical points and topology edges along the time axis, as well as along the three spatial coordinates. For critical points, this task can be expressed in terms of a first-order expansion as

$$\begin{bmatrix} \frac{\partial f_0}{\partial u} & \frac{\partial f_0}{\partial v} & \frac{\partial f_0}{\partial w} & \frac{\partial f_0}{\partial t} \\ \frac{\partial f_1}{\partial u} & \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial w} & \frac{\partial f_1}{\partial t} \\ \frac{\partial f_2}{\partial u} & \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial w} & \frac{\partial f_2}{\partial t} \end{bmatrix} \cdot \begin{bmatrix} \delta u \\ \delta v \\ \delta w \\ \partial t \end{bmatrix} = - \begin{bmatrix} f_0 \\ f_1 \\ f_2 \end{bmatrix} . \quad (20)$$

While various numerical methods are available for solving poorly determined systems of equations such as these, there is a problem with user feedback. For feedback of the spatial offset from the user-specified point to the found point, we simply rubber-band a line between the two. For providing feedback of the found point along the time domain, it is clearly undesirable to shift the time associated with the display on behalf of the user. This would be analogous to moving the cursor on behalf of the user for spatial feedback, which has been found to be very undesirable. Clearly, there are many user interface problems to be dealt with as we move toward faster and more complex visual feedback capabilities, but the rewards for solving them can be expected to be substantial. The more we as system designers can simplify complex tasks for analysts, the more they will enjoy using the tools that we create.

Conclusions

In the work presented in this paper, we have clearly established the feasibility of interactive vector field topology extraction on today's state-of-the-art workstations. The potential benefits and advantages for scientists and engineers using these tools are clear. Older methods of creating automatic abstract visualizations of objects tend to produce copious amounts of imagery to be examined in attempting to extract meaning from vector-field data. In contrast, direct interaction with maps of the data to create features of particular interest in particular parts of the domain of the data provides a powerful subdivide-and-conquer approach to data interpretation.

While we are pleased with the progress we have made to date, the tools we have created are new, and the underlying mathematics is not trivial. This leads to a user learning curve that is difficult for novice users, particularly those who are not familiar with vector-field theory. Accordingly, it is strongly recommended that further work be undertaken with respect to the development of a "style guide" for feature extraction using multidimensional graphics input devices, based on a formal psychology study of various proposed approaches. This will provide a basis for freeing up more of the user's thinking time for the underlying field behavior by making the viewing and 3D cursor controls more automatic.

Acknowledgments

David MacDonald, a graduate student at McGill University, Montreal, Quebec, Canada, coded much of the object-oriented mechanisms for performing multidimensional multivalued field evaluations while he was with Visual Edge Software in Montreal. These mechanisms make the implementation of the ideas presented herein very straightforward. Without them, it would have been easy to become lost in the details of each field data representation used in the examples presented here. The transient room airflow data was provided by Advanced Scientific Computing of Waterloo, Ontario, Canada. Spatial Systems Inc., Billerica, MA, provided a Spaceball for the development of the multidimensional graphics input interface for 3D topology extraction. The underlying research was supported by the Natural Sciences and Engineering Research Council of Canada through an Industrial Research Fellowship. The videotape production was partially funded by NSERC Operating and Strategic Grants, through the Computer Graphics Laboratory of the University of Waterloo.

References

1. W. L. Hibbard, "4-D Display of Meteorological Data," *Proc. 1986 Workshop on Interactive 3D Graphics*, Department of Computer Science, University of North Carolina, Chapel Hill, 1986, pp. 23–36.
2. R. R. Dickinson, "Interactive 4D Visualization of Fields," *Technical Report CS-89-15*, Department of Computer Science, University of Waterloo, Ontario, Canada, 1989.
3. D. Fowler and C. Ware, "Strokes for Representing Univariate Vector Field Maps," *Proceedings of Graphics Interface '89* (Canadian Information Processing Society), London, Ontario, Canada, June 19–23, 1989, pp. 249–253.
4. G. D. Kerlick, "A Level Surface Cutting Plane Program for Data Visualization," *Extracting Meaning from Complex Data: Processing Display and Interaction, Proc. SPIE* **1259**, 2–13 (1990).
5. C. Upson and G. D. Kerlick, "Volumetric Visualization Techniques," in *ACM SIGGRAPH '89 Course Notes*, No. 13, 1989.
6. J. Helman and L. Hesselink, "Automated Analysis of Fluid Flow Topology," *Proc. SPIE* **1083**, 144–152 (1989).

7. J. Helman and L. Hesselink, "Representation and Display of Vector Field Topology in Fluid Flow Data Sets," *IEEE Computer* **22,** 27–36 (1989).
8. W. E. Boyce and R. C. DiPrima, *Elementary Differential Equations and Boundary Value Problems*, John Wiley & Sons, Inc., New York, 1986.
9. R. R. Dickinson, "A Unified Approach to the Design of Visualization Software for the Analysis of Field Problems," *Proc. SPIE* **1083,** 173–180 (1989).

Received November 6, 1989; accepted for publication December 1, 1990

List of videotape captions

The videotape that accompanies this paper has been designed to be complementary to the printed material, as well as being informative and visually interesting independently of the printed paper. The narration is used to communicate what the user is doing with various graphics input devices at various points in time, as well as to provide a qualitative overview of the need for interactive topology extraction. The viewer is referred back to this paper for details on the mathematical concepts behind the system's responses to given user interactions.

Scene 1 Some examples of excessive visual clutter from the use of automatic graphics object creation systems.

Scene 2 Interactive 2D topology extraction. The red and blue curves respectively denote +1 and −1 for s_A in Equation (10).

Scene 3 Interactive 3D topology extraction. Vortex centerlines are selected from user-specified points, and tracked in both directions about the found point until either a critical point or the domain boundary is encountered.

Scene 4 Interactive transient 3D topology creation paradigms. This scene shows the creation and evolution of various vortex centerlines associated with the air flow through a room. The inlet velocity is steadily increased from the start of analysis time.

Robert R. Dickinson *Pacific Visualization Systems, 51 Pacific Ave., Senneville, Québec, Canada H9X 1B1.* Dr. Dickinson received his B.E. degree in civil and agricultural engineering from the University of Melbourne, Australia, in 1979. He worked with GHD Pty. Ltd., consulting engineers, prior to obtaining an M.Sc. degree in engineering from the University of Guelph, Ontario, Canada, in 1982. He subsequently worked with Jenike and Johanson Ltd. and H. G. Engineering Ltd., Toronto-based consulting engineers, on the analysis of large-scale bulk handling systems and on finite element analysis system software. Dr. Dickinson obtained his Ph.D. in systems design engineering from the University of Waterloo, Ontario, in 1987. For his Ph.D. thesis, he developed a unified approach to the analysis of large stochastic systems and stochastic continuum systems. While working as a postdoctoral researcher in the Computer Graphics Laboratory of the Department of Computer Science at the University of Waterloo in 1988, he developed advanced vector and tensor field curve tracking algorithms. From 1989 to 1990, Dr. Dickinson supervised the design and implementation of the Via interactive field exploration system at Visual Edge Software Ltd., Montreal, in association with the Computer Graphics Laboratory of the University of Waterloo. The Via software system was used to produce the video graphic that accompanies this paper. He started Pacific Visualization Systems in 1991. Dr. Dickinson is an Adjunct Professor at the École Polytechnique de l'Université de Montréal; he is a member of both ACM SIGGRAPH and the Association of Professional Engineers of Ontario.