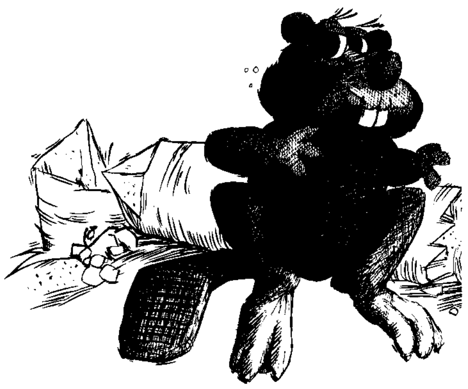
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT



Interactive 4-D visualization of fields

Robert R. Dickinson

CS-89-15

July, 1989

Interactive 4-D visualization of fields

by
Robert R. Dickinson

Visual Edge Software Ltd., 3870 Cote Vertu
St. Laurent, Quebec, Canada H4R 1V4

Acknowledgements

The atmospheric humidity field was provided by Ross Trafford from the Atmospheric Environment Service, Downsview, Ontario. The molecular electric field model was provided by Helmut Hayd from the University of Stuttgart. The analysis data for the liquid crystal defect was provided by Drs. Mitchell Luskin from the University of Minnesota and Robert Cohen from the University of North Carolina. The fluid flow analysis results for the airflow through the room was provided by Advanced Scientific Computing Ltd., of Waterloo, Ontario, Canada. The 2D solid mechanics analyses were provided by Peggy Baehmann from the Center for Interactive Computer Graphics of Rensselaer Polytechnic Institute. H.G. Engineering Ltd. of Toronto, Ontario, provided assistance with the 3D solid mechanics analysis using ANSYS software. The author would also like to thank Dr. Richard Bartels from the University of Waterloo for the many helpful discussions regarding the material presented in this report.

This work was primarily funded by Visual Edge Software Ltd., while the author was associated with the Computer Graphics Laboratory of the University of Waterloo. David MacDonald of Visual Edge Software Ltd programmed much of the interaction paradigms and animation history mechanisms. Partial funding for this research was provided by grants and a fellowship from the Natural Sciences and Engineering Research Council of Canada, and by donations from Digital Equipment Corporation and Silicon Graphics.

Interactive 4-D visualization of fields

Robert R. Dickinson

Visual Edge Software Ltd., 3870 Côte Vertu
St. Laurent, Québec, Canada H4R 1V4

ABSTRACT

The term 'field' is used herein for a process which associates a physical quantity with each point in a region of space and time. In general, fields can be scalar, vector, or tensor valued. Due to the typically large number of variables involved, many different visualization procedures are possible. To provide analysts with immediate accessibility to many alternatives, much development effort is required in the areas of user-interface design and advanced field visualization algorithms. This paper describes the design of a system in the context of generic field visualization. New visualization algorithms can be implemented in a way that is independent of the representation details underlying a particular set of analysis results. User interfaces and rendering methods can be easily shared across fields of differing order and dimensionality. The use of time as a visualization dimension is formalized and pertinent interface controls are discussed. These design attributes lead the way to many new possibilities in interactive visualization, permitting the exploitation of the fast processing, graphics pipeline speeds and multi-dimensional graphics input devices that are now obtainable with graphics workstations.

Key Words and Phrases: Scalar-, vector- and tensor-valued fields; piecewise analysis of continuum systems; visualization in scientific computing; interactive feature extraction; system design.

1. INTRODUCTION

For the purposes of this paper, a field is defined as a mapping from an n-dimensional space to the m-components of a corresponding field value. Fields can be scalar, vector, or tensor valued. For scalar fields, m is always 1. For vector fields, m is the dimensional length of the vector value. For tensor fields, m is the number components in the tensor value.

More formally, scalars and vectors can be thought of as zero and first order tensors respectively[1]. An example of a second order tensor field is the stress associated with a thin plate. In this case the stresses perpendicular to the plane of the plate at a given point are negligible. The complete stress state associated with a given point in the plate can be written in terms of a symmetric 2 by 2 matrix. Thus m is 3 for this class of field.

Mathematical operations on a field with m components produce new fields with a new m, which we denote as m'. An objective in field visualization is to display the continuous behaviour of the $(n+m+m'\ ..)$ dimensions unambiguously. More specifically, it is important to provide for the display of explicit interrelationships between selected components. This is because for high order fields, the independent display of any one of the m components is typically of little or no interest, relative to the continuously varying interrelationships among component values.

From this background, the task at hand can be thought of as being $(n+m+m'\ ..)$ dimensional, where $(n+m+m'\ ..)$ can be typically as high as ten or more. Since the number of dimensions available for actual display on a workstation at one time is limited, there are many alternative display methods to be chosen from, each with a distinct selection from or combination of the $(n+m+m'\ ..)$ dimensions of the task at hand. This leads to many interesting user interface design issues that need to be dealt with.

Another important attribute of field visualization that distinguishes it from other forms of scientific data visualization is that there is an infinity of information within the continuous domain of a given field. Hence field visualization is inherently a *summarizing* process in which there are many different methods of summarizing various aspects of a field that might be of interest. Clearly, a systematic approach is needed to make these methods interactively accessible to users, permitting the selection of the visualization and feature extraction techniques that are most appropriate for a given problem.

A major practical consideration in the design of field visualization software involves the diversity of field-analysis methods. These include finite difference methods, finite volume methods, boundary and finite element methods, spectral methods, and special purpose variational methods[eg:2]. All of these methods are approximations to the solution of continuum systems. At the end of the analysis, the output is characterized by numbers (field values) associated with discrete points (nodes) or regions (elements) of the domain. For some methods, such as the finite element method, an interpolation function that is consistent with the underlying mathematical formulation might also be available. Similarly, for empirically measured fields, data values at discrete points are typically used along with a relevant interpolation method.

Until recently, the post-processing software associated with common analysis codes have tended to carry this discrete data structure all the way through to what the analyst sees on the screen of a graphics workstation. The geometry in the underlying data structures was typically inflexible, and dedicated to a given analysis method. Providing analysts with immediate access to a selection of old 'summarizing' procedures was difficult, let alone adding new ones.

This paper describes the design of a system that is based on the above generic concept of a continuous field. New visualization algorithms can be implemented in a way that is independent of the representation details underlying a particular set of analysis results. User interfaces and rendering methods can be easily shared across fields of differing order and dimensionality. The ability to map one of the n generic independent variables to real time can be provided in a natural way. This use of time as a formal visualization dimension provides analysts with easy access to the best real-time performance that a given workstation can deliver, without compromising on software portability.

2. OVERALL SYSTEM DESIGN

The overall structure of the system is illustrated schematically in Table 1. The crux of this design lies in the data flow between the *tools* and the *generic field* source. The only information that the *tools* need to perform their assigned tasks efficiently is m values (and optionally, selected low order derivatives) at arbitrary points in the n-D domain of the field. The tools produce geometry, scalar values associated with coordinates of that geometry, and colour and timing information, all of which is sent to the *renderers* which form the interface between the tools and a graphics hardware pipeline. Because the tools are not constrained by the data structures of a given field analysis method, they have complete freedom to adapt to field behaviour, and permit simple user control of the trade-offs between image quality and precision on the one hand, and interaction speeds and response times on the other.

Table. 1. The primary modules and their interconnections

GENERIC FIELDS
Evaluators for each supported field representation
An interface between generic field value requests and the actual evaluators, including a mechanism to permit the user to indicate a 'current field' out of a collection of fields

(a) Requests for field values (b) Requests for local domain boundary information ↑	↓ (a) Values (and derivatives) (b) Boundary intersections and local curve segments

TOOLS
Numerical algorithms for tracking curves and sweeping out surfaces that satisfy specific conditions relevant to a given feature extraction process
Adaptive and flexible volume/surface intersection procedures
User interfaces capable of monitoring system performance and adapting accordingly, and of exploiting the ease with which users can 'walk around' field space with the aid of graphical input devices

Geometry, scalar values, colour and timing data ↓	↑ System performance data w.r.t the use of time

RENDERERS
Mechanisms for associating field coordinates with screen coordinates
Mechanisms for associating scalar values with colour
Mechanisms for controlling the use of time as a formal visualization dimension

3. GENERIC REPRESENTATIONS OF FIELDS

The overall design concept described in Table 1 is applicable to large classes of field visualization problems. But in order to provide a framework for discussion of some specific tools in section 4 and to pragmatically define the scope of the present paper, the following limits are placed on n for the remainder of this paper.

We explicitly deal herein with fields in which n is either three or four. For fields in which $n = 3$, we denote the three independent variables as (u,v,t), and use u and v as spatial dimensions of the output geometry and map t to real time. 2-D steady state fields form a special case in which t is invariant. Similarly, for fields in which $n = 4$, we denote the four independent variables as (u,v,w,t), and use u, v and w as spatial dimensions and t as time. (Similarly, 3-D steady state fields are a special case in which t is invariant.)

The software implementation of the above generic concept of fields follows readily via well-known information hiding techniques. The user interacts with an array of fields, where all visualization operations are performed on the currently selected field. Consider the simplest case of a scalar field visualization tool that only needs the value of the field at requested points in the field domain in order to perform its task efficiently. Such a tool can simply send requests to the field evaluators giving the three (or four) data elements comprising an arbitrary point in n-space, and receive an indication of whether or not that point is inside the domain of the current field, and if so, what its value is.

Such a tool need not know any of the details about its representation — it might be a tensor product B-spline in four variables; a uniform grid of values that are to be quad-linearly interpolated; or sets of connected networks of irregular polyhedra, along with relevant spatial and time interpolating functions. In all of the algorithms referred to in section 4, we assume that these representational details are indeed hidden, leaving the corresponding software immune to changes or augmentations of the actual field evaluators. The major generic requirement of a given evaluation scheme is that field values are expected to be finite and piecewise continuous, (and in some cases, depending on the algorithm, first and second order differentiable). At run-time, the use of a given tool is restricted until such time as the data for a relevant field exists somewhere that is accessible to the relevant field evaluators.

As m increases, the number of data items that need to get passed from the evaluators to the tools increases. This increase is exaggerated in cases where tools require partial derivatives in addition to the m field values. But the number of data items remains manageable and does not threaten the feasibility of the above approach for a large class of practical field problems. Consider such a class defined in Table 2 by the limits placed on m for vector and tensor fields.

Table 2. Limits on number of values associated with a large class of practical fields.

n	vars.	Vector	2nd order Symm. Tensor
3	u,v,t	$m = 2$	$m = 3$
4	u,v,w,t	$m = 3$	$m = 6$

Practical examples of vector fields in this class include transient fluid flow fields in 2 and 3 spatial variables. Examples of tensor fields include transient symmetric stress tensors.

A more general set of field visualization tools can be served by the field value and derivative requests that are summarized in Table 3. Each entry in this table corresponds to a unique request. The number in each entry indicates the number of members in the returned data structure. All of the tools described later in this paper use structures of these forms.

Table 3. The numbers of data items including derivatives associated with selected field-value requests.

	Scalar		Vector		2nd order S.Tensor	
$n \rightarrow$	3D	4D	3D	4D	3D	4D
value	1	1	2	3	3	6
deriv_order1	4	5	8	15	12	30
deriv_order2	10	15	-	-	-	-

It is a simple task to add more data structures as the need arises. The important concept is that there exists a manageable number of data structures that cover large classes of visualization needs.

There are many benefits of adopting this approach. For example, all of the branching statements required to perform a given evaluation using the correct 'current field' are localized in an analogous manner to collections of objects in an object-oriented programming environment. More generally, requests for field values need not even be restricted to local function calls. A natural generalization would be to permit requests across networks via remote procedure calls in a distributed computing environment. This would eliminate the apparent need to pass entire sets of memory-consuming analysis results to the visualization system before it can get 'off the ground'.

Another benefit of this generality is that special cases fall into place in natural ways. For example, 'steady state' fields can be readily accommodated by simply adopting the convention that all values are constant over some interval $t_0 \leq t \leq t_1$, and non-existent elsewhere. Thus the data associated with a steady state field that corresponds to the termination of a damped transient field problem can be smoothly joined to the transient field data without the user having to switch current fields.

Further, even though t is always mapped to the real time, T, of the display device, we need not be restricted to fields in which the last variable is physically 'time'. (The mechanism for mapping t to T is described in section 6.). That is, we can alter the actual use of the four dominant visualization dimensions by simply altering the order of values in the relevant field value data structure prior to handing them to the tools.

4. FIELD VISUALIZATION TOOLS

The simplest tool, but least useful in terms of getting an overall impression of the behaviour of a given field, is a 'numeric cursor' digital readout of field values at points indicated by 'walking around' the spatial domain of a field with the aid of various graphics input devices. While this operation is clearly of limited use on its own, we have found that it fulfills an apparent desire to see actual numbers when a user has identified and zoomed in on a given feature of interest. It thus has an important role to play in complementing the other tools.

Generally, tools are comprised of numerical algorithms that perform operations on the current field, and the user interfaces used to control when and how these algorithms are executed. The algorithms include operations on a sequence of fields of the form $f(u,v,w,t_0)$, $f(u,v,w,t_1)$, . . . , $f(u,v,w,t_N)$, where t_i forms a monotonically increasing sequence over an interval of 'model time'. As well as performing operations on a given field, $f(u,v,w,t_i)$, the continuity of specific features with respect to time can also be tracked via numerical techniques. From the user's point of view, the task of discretizing the continuous time of a given field into a monotonic sequence of images should be a hidden detail. Mechanisms for doing this automatically involve two-way communication between the renderers and the tools.

As shown in Table 1, the primary item of information that the renderers expect is geometry. The basic geometric primitives that are dealt with here are 3-D curves, tubes, and surfaces. Curves are handed to the renderers as simply connected polylines. Tubes are passed as polylines, along with radii and surface properties. Other surfaces are represented by irregularly connected triangular patches. While these simple primitives are far less compact than those obtainable using splines, they are well suited to the assignment of colour to individual geometric elements according to a given scalar value and colour interpolation scheme. Some applications of each of these primitives are now given.

4.1 Curves

4.1.1 2-D contours and streamlines

Algorithms for tracking contours of bivariate scalar fields of the form $f(u,v,t_i)$, and contours on section planes of trivariate scalar fields of the form $f(u,v,w,t_i)$, can be readily designed to exploit low order partial derivatives[3]. Similarly, 2-D gradient streamlines on section planes of $f(u,v,w,t_i)$ can be used to highlight point singularities[4]. While continuous-tone colouring of surfaces provides good overall impressions of field behaviour, curves provide excellent local directional information.

4.1.2 2-D Tensor field lines through finite element models

An example of the result of interactively tracking tensor field lines (curves which are everywhere parallel to the eigenvectors of a tensor field) through a theoretically interesting stress field is illustrated in Figure 1. In this field, a pair of forces of equal magnitude and opposite direction are acting near the centre. The way in which force is transmitted from the source to the sink, and the influence that this has on the surrounding media, is clearly evident from the map of tensor fields lines. The interaction paradigm used to create these curves was roughly as follows. Having requested a particular curve type from a menu, the user moves the cursor around the domain of the field. A pair of curves is drawn 'uphill and downhill' about the moving cursor at speeds approaching refresh rates. A button press at a given location records the current curve pair. Hence analysts can essentially paint pictures interactively, highlighting the features required for either analysis feedback, future records or simply evidence to use in communicating concepts to colleagues.

Figure 2 illustrates a more practical example — a wall-mounted bracket, in which there is no point source or sink. In this example, the way in which the applied distributed force is transmitted in tension across the top member, and in compression down the diagonal strut, is clearly evident from the map of tensor field lines. The constriction-like features in the top left also give the analyst clues as to how the geometry might be modified to reduce the maximum shear stresses in this region.

4.2 Tubes

4.2.1 The analysis of a stable defect pair in a liquid crystal

Strong visual clues about the 3-D form of space curves are observable when they are rendered as tubes wrapped around their centrelines. This effect can be exaggerated if lights are used to highlight features of special interest. An example is illustrated in Figures 3 and 4. The centrelines of these tubes are parallel to the orientations of molecules in a liquid crystal defect. In this example, scalar quantities such as the vector magnitude are of little significance, so colour and lights can be used to highlight and distinguish the two directional features of particular interest. A similar interaction paradigm to that described earlier was used to create this model.

4.2.2 An electric field associated with a macro molecular model

The same procedures can be applied to the gradient of scalar fields of the form $f(u,v,w,t_i)$, as illustrated in Figure 5. The placement of each distinct collection of tubes into a separate graphical sub-structure is straightforward, permitting the user to switch features on and off at will. In this case the tubes are colour coded according to gradient magnitude, but there is no lighting and shading used. On workstations with the ability to simultaneously display enough colours, both lights and scalar-value colour mapping can be used simultaneously.

4.2.3 A computational fluid dynamics application

In many fluid flow problems, curvilinear singularities are of interest. The centrelines of the vortices illustrated by the streamline in Figure 6 provide an interesting example. Air flows from the inlet at high velocity over the baffle towards the outlet, where it then gets entrained in a pair of vortices. Due to the complex nature of these vortices, visualizing their form with streamlines alone is difficult because the resulting curves 'cross themselves' many times over when projected onto a 2-dimensional graphics monitor. Further, when the velocity in the direction of the vortex centreline is low, streamlines tediously circulate around the centreline, taking a long time to track and to draw on the screen. Hence it becomes interesting to focus specifically on curves that define the centrelines of the vortices. An example is illustrated by the additional curve in Figure 7 that is not shown in Figure 6. Figures 8 and 9 illustrate the 3-D nature of this curve. The velocity along this line is extremely low relative to the surrounding air. The blue spot in the middle indicates the location at which the velocity magnitude passes through zero, or equivalently, the point at which the velocity direction changes sign.

The interaction paradigm used to create the model of the vortex centreline is roughly as follows. The user moves a cursor around in 3-space in the vicinity of the vortex. With each screen update, the nearest point on the curve defining the centreline of the vortex is located, and the centerline is tracked about that point and displayed on the screen. At any given time during this interaction, the user can save a chosen centreline as a permanent tube in a manner analogous to that described earlier. For transient fluid flow visualization, the automatic tracking of such features with respect to time follows in a natural way.

4.2.4 3-D Tensor fields lines through finite element models

The use of tensor field lines to communicate the ways in which force is transmitted through 3-D media is illustrated in Figure 10. Various methods for displaying scalar functions of tensor fields have been been commonplace for some time. But the ability to display the coupling between all 6 components of a symmetric 3X3 tensor field in this way is new, and has only becoming feasible as a result of increases in the performance of graphics workstations. These curves were painted interactively using the same sort of user interface as that described in section 4.1.2. For 3X3 tensor fields, there are three distinct curve sets about any given point. Accordingly, the simultaneous tracking of any combination of these is provided for in the curve tracking menu for tensor fields.

4.3 Surfaces

4.3.1 Colour coded spherical icons

At the very beginning of this section, the use of a 'numeric cursor' was described. A natural progression from this sort of simplistic interface is to wrap a sphere around interactively steered points, colour-coding it according to the value of a relevant scalar field, or scalar function of some higher order field (eg. velocity magnitude, eigenvalues, Von Mises stresses, and so on). This gives analysts a fast means of locating 'hot-spots' in regions of particular interest. In transient fields, collections of such objects can be left at given points in the (u,v,w) domain, while the system automatically alters their colour with respect to time.

4.3.2 Continuous tone section planes through an atmospheric humidity field

A monochrome example of the use of continuous colour maps on volume sections is illustrated in Figure 11. In this example, the wave-like highs in the humidity field are of particular interest. A monochrome interpolation scheme was found to be more effective in highlighting these features than the full colour schemes used in the other examples. The level of subdivision that surfaces are discretized into prior to display is an editable parameter of the section plane renderer. For section planes, this is expressed in terms of limits on the data value variation across elements of the geometry. This sort of adaptive control of the trade-offs between image quality and drawing speeds is easy to provide for with the continuous nature of the generic function evaluators described in section 3.

Time plays an important role in this tool. The provision for interactively steering a section plane through a given volume provides an alternative to the apparent need to perform volume imaging. Similarly, by leaving section planes in a given spatial location, an impression can be gained of the full 4-D variation of transient scalar functions using the timing controls described in section 5.

4.3.3 Isotimic surfaces through the molecular electric field

An example of the use of isotimic (constant value) surfaces is illustrated in Figure 12. The three 'bubble-shaped' features on the left correspond to the three singularities in Figure 5. The simultaneous use of shaded gradient-streamlines and isotimic surfaces is illustrated in Figure 13. For isotimic surfaces, the relevant scalar value can be specified by using the 'numeric cursor' described earlier along with a mouse click, at which point there is a slight pause while the isotimic surface geometry around that point is created and then handed to the renderers. In transient fields, the evolution of the geometry of isotimic surfaces with respect to time can be used together with the use of section planes described above. The algorithm for this task builds on ideas from the marching cubes algorithm[56]. An initial virtual grid is used as a starting point. The resulting coarse polygonal information is adaptively refined until relevant geometric tolerances have been satisfied.

Another example of the use of isotimic surfaces is shown in Figure 14, which clearly shows the 3-D form of the low pressure zone associated with the vortices of the velocity field shown in Figures 8 and 9.

4.3.4 Tensor fields surfaces through 3-D FEM models

Figure 15 shows a family of surfaces that are everywhere orthogonal to the tensor field lines depicted in Figure 10. In this way, complete information about the directions and magnitudes of the states of normal stresses are viewable throughout the tensor field medium at a given instant in time.

4.4 'Bubble traces'

The explicit use of the variable t as a visualization dimension is described in more detail in section 5. It is difficult to capture the importance of this dimension with static images alone, but 'stroboscoped' images provide some clues. An example of the use of animation with respect to time in a steady state analysis is illustrated in Figure 16. The bubbles in this image are spaced a constant time interval apart. The interaction is analogous to that described earlier except that two button presses are used — one to animate a given streamline and a second to save the geometrical description of the stroboscoped image.

A limitation of the traditional 'bubble' shape is that its simplistic form does not lend itself well to visualizing variations in the torsion of space curves. So in addition to bubbles, the user can choose other objects such as the 'paper jet' illustrated in Figure 17, permitting more effective use of lights and shading.

5. FIELD RENDERERS AND TIMING CONTROLS

As shown in Table 1, the renderers control the way in which geometry, timing and scalar data are used to display images on the screen of a workstation.

5.1 Screen space and colour

Mechanisms for associating the spatial portion of field coordinates (i.e. u, v, and w) with screen coordinates (X, Y, and Z) are well-known and need not be dealt with in detail here.

With respect to colour, our current implementation is restricted to mapping a given portion of a scalar domain to colour space. Interactive control of the use of particular portions of colour space is provided for, but we have chosen to limit the number of independent variables in the use of colour to one. Again, methods for performing this task are well-known and are not dealt with in detail here.

However, the use of time as a formal visualization dimension is of particular interest in interactive field visualization, and is described in more detail below.

5.2 The use of time — design objectives

In this section, the term *resolution* is used in reference to the number of distinct frames that are displayed per unit of real time, unless noted otherwise. Clearly, the major consideration with respect to the use of time as a formal visualization dimension is that resolution is dependent not only on screen refresh rates, frame buffer update speeds and the compute server performance of a given hardware environment, but also the size and complexity of the relevant display lists. Hence resolution can be expected to vary dramatically within a given hardware environment, as well as across different hardware systems.

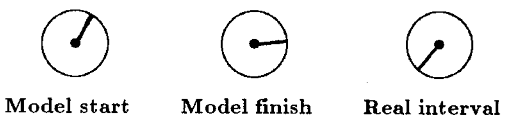
Work aimed at developing methods for controlling the faithful use of 'real-time' in computer-graphics has been underway for many years[eg:7]. More recently, real-time previewing of the effects of non-linear editing of the time-interpolation of inbetweens in film making has also been developed[8]. In these applications, the use of time is well-defined and the output sequence is fixed to a given number of frames per second, requiring display update speeds to be sufficient to be able to display the most complex image within some given amount of time. Similar constraints apply to the design of flight simulators.

But in contrast to these applications, the use of 'real-time' in scientific visualization is relatively abstract. As described earlier, the underlying independent variable that gets mapped to real-time need not even physically correspond to time at all.

From the point of view of a field analyst attempting to interpret quantitative results, consistency in the way a visualization system handles time is paramount. In a given static image, pixel colours in the vicinity of a given normalized screen coordinate are expected to be similar, regardless of device resolution. Similarly, the entire image in the vicinity of a given point in 'display-time' can be expected to be basically the same from one hardware environment to another, regardless of hardware performance. The only difference should be 'image quality' in so far as it is affected by the number of frames in between the first and last frames.

Consider extending this analogy to interactive alteration of clipping and 'zoom' parameters. If the projection of a given static image is such that the pixel resolution around a particular feature is insufficient to capture the essential characteristics of the feature, then the user gets to zoom in on the feature until the required information is clearly visible. Accordingly, if the real-time resolution of a given sequence is so poor as to miss a particularly short duration feature, then the user should be able to interactively 'zoom' in on the model-time interval of interest, obtaining more information over a given real-time interval, and 'clipping' information outside the 'time-zoom' limits. This is in marked contrast to simply controlling the wait interval (or frame buffer swapinterval in a double buffered system) in a fixed sequence with a fixed number of frames per unit of model time.

From this background, the main design objective for the use of time as a formal visualization dimension becomes clear. The user requires 3 valuators (egs. knobs or slider bars) to control the mapping from field-model time, t, to real-time, T:



The three corresponding variables are denoted as t_{start}, the start time in field-model space, t_{finish}, the finish time in field-model space, and T_{real}, the real-time duration to which $t_{finish} - t_{start}$ is mapped on to. The automatic computation of an appropriate sequence of sets of geometry and scalar values for fields of the form $f(u,v,w,t_i)$ and the control of the faithful real-time display of them, is to be left up to the tools and renderers. The mechanism that we have adopted for this task is described below in pseudo code. It is assumed that the workstation is doublebuffered, and that control of when buffers get swapped is available from the applications software level. Note that t_i, $i = 0,..,N$ is in general, a non-uniform monotonic sequence, and that the resulting display sequence, T_i, corresponds to a subset of t_i. Also note that all of the geometry information is computed outside of the main loop, and stored in a display-list or analogous format. This approach is required to provide the maximum possible density of frames per unit time, because the time that it takes to compute the geometry, scalar values, and colour coding information is typically over three orders of magnitude greater than the time that it takes to display it.

```
$N_{estimate}$ ← an estimate of the number of frames that the
  hardware is capable of displaying in $T_{real}$.
$N$ ← $(N_{estimate} \times M) - 1$ †
Compute geometry for $f(u,v,w,t_i)$ for $i = 0,\ldots,N$.
do {
  Execute display list for $t_0$ in back-buffer
  $T_{start}$ ← the real time now
  Execute screen update now (swap buffers)
  while $(T < T_{start} + T_{real})$
  {
    Clear back buffer
    $T$ ← the real time now
    $i$ ← estimate_i_to_give_spare_time$(T - T_{start})$†
    Execute display list for $t_i$ in back-buffer
    wait until $\left(\frac{t_i - t_{start}}{t_{finish} - t_{start}} \geq \frac{T - T_{start}}{T_{real}}\right)$
    Execute screen update now (swap buffers)
  }
  if (performance can be improved) †
  {
    Re-evaluate $N$ using accumulated performance stats.†
    Compute geometry for $f(u,v,w,t_i)$ for $i = 0,\ldots,N$.
  }
} while no user interrupts
```

† The crux of this design lies in pre-computing display list information for more frames than the hardware is capable of delivering in T_{real}, and then actually displaying a selection from these that is adaptable to fluctuations in system response times. The actual density of the initial stored sequence is controlled by M. Low values of M can be used to give coarse initial guesses at optimal sequences, which can then be refined during the pause between cycles, accounting for possible inequalities in display speeds from one frame to the next. Responsibility for the faithful use of real-time is localized within the function:

"*estimate_i_to_give_spare_time*$(T - T_{start})$",

which can be designed to use statistics obtained from the last cycle to estimate the next frame that can be written into the backbuffer with time to spare. User feedback of how frequently frames get displayed late due to overly optimistic estimates is easy to provide for using graphs of model time vs real time. This function also takes responsibility for ensuring that the last frame is displayed on time, even if this means skipping (say) the second to last frame and waiting until $T = T_N - T_{start}$.

The main advantage of this design is the simplicity and consistency of the user interface. For a given setting of the three valuators, the same software running on two systems with differing performance capabilities will produce consistent first and last frames and will cycle over the same interval of real time, even though N might be totally different on each system. As described earlier, the difference is limited to image quality in so far as this is affected by N. The consistency of the content of the display with respect to time from one cycle to the next is also kept as high as possible.

Special cases of timing requests also fall into place naturally. For example, 'forwards and backwards' is simply controlled by permitting t_{start} to be either greater or less than t_{finish}, and constraining T_{real} to be positive. Further, the use of 'stills' at arbitrary values of t (as distinct from discrete 'frames' in a finite sequence), can use the same interface by simply adding a button that constrains t_{start} to be equal to t_{finish}.

The same algorithm can be applied to using time as a visualization dimension for steady state fields. For example, to animate the streamlines in Figures 16 and 17, the total time that a mass-less particle takes to travel the length of the streamline is used to define a domain in t. The user then uses the three knobs described above to control the relationship between t and T, and the portion of the streamline that gets animated.

Further work is planned for fields in which a long playback duration (say $(T_{finish} - T_{start}) > 15$ secs) is of interest. In such cases, a fourth valuator to interactively control non-linear relationships between t and T is required. This would permit the user to skip over uninteresting regions of t, and concentrate on more interesting regions at the touch of a valuator.

6. SUMMARY AND CONCLUSIONS

In summary, the recent dramatic improvements of both processing power and graphics pipeline speeds provided in today's graphics workstations will lead to new ways of thinking about interactive visualization in the analysis of field problems. Based on the use of a prototype of the design described herein, it is becoming reasonable to suggest that interactive feature identification will become part of the analysis itself, rather than an afterthought as suggested by older terms such as 'post-processing'. As iterative design and analysis procedures evolve with more extensive use of feedback loops, it will become more and more important to provide analysts with fast, highly interactive methods of extracting the particular aspects of a given field that are of interest. This is in marked contrast to older 'batch processing' approaches that involve the automatic creation of large amounts of pictorial representations of analysis results. Just as 'visualization' has been invented to describe the process of providing more immediate access to very large amounts of analysis data, 'interactive visualization' will be 'invented' to describe the process of providing more immediate access to the particular features that are of interest to the analyst at particular points in both the spatial and time domains of a given field.

7. OTHER APPLICATIONS

Work is already underway on the application of the tools described herein to other phenomena, including 4-D wind fields from meteorology studies, and results of transient stress analyses. The latter includes 4-D auto crash simulations using DYNA3D software, and 3-D metalforming simulations involving automated re-meshing with respect to time[9]. With respect to tensor fields, applications beyond the realm of solid mechanics are also of interest. These include studies of the interaction between components of coupled vector processes, such as the flows of holes, electrons, and heat through semiconductor devices; and studies of the covariance of probabilistic vector fields.

Finally, while the emphasis in this paper has been on the use of tools that produce geometrical information suitable for 'geometry pipeline' oriented graphics workstations, it should be noted that the overall design is by no means incompatible with the use of tools that produce pixel information directly, for use with 'image computers'[10]. In fact an early implementation of some of the components of this design was used to volume render a sequence of atmospheric humidity fields[11], the results of which were presented at SIGGRAPH '88. An interesting extension of this work might involve performing 'ray-tracing'-like procedures on continuous scalar fields with colour-coded geometries of the styles described herein imbedded within the scalar field, potentially combining the best attributes of both graphics system designs.

References

1. A.P. Boresi and P.P. Lynn, *Elasticity in Engineering Mechanics,* Prentice Hall, Englewood Cliffs, New Jersey (1974).

2. D. Kinderlehrer, Recent developments in liquid crystal theory, pp. (to appear) in *Colloquium Lions, North Holland,* (1989).

3. R.R. Dickinson and R.H. Bartels, The interactive editing, contouring and sectioning of empirical fields, *IEEE's Computer Graphics and Applications* **9**(5)(1989).

4. R.R. Dickinson and R.H. Bartels, Fast algorithms for tracking contours, streamlines, and tensor field lines, *Updating the State-of-the Art in Civil Engineering Computing Tools, Third Intl. conf. on Computing in Civil Eng., CSCE, ASCE, CCES;* **2** pp. 495-502 Dept. of Civil Eng., Univ. of British Columbia, Vancouver, Canada V6T 1W5, (1988).

5. W.E., Lorensen and H.E. Cline, Marching Cubes: A High Resolution 3D surface reconstruction algorithm, *Computer Graphics, ACM SIGGRAPH* **21**(4) pp. 163-169 (1987).

6. J. Bloomenthal, Polygonalization of implicit surfaces, in *Course notes for course 16: The Modeling of Natural Phenomena*, ACM Siggraph (1987).

7. B. Ackland and N. Weste, Real time animation playback on a frame store display system, *Computer Graphics, ACM SIGGRAPH* **14**(3) pp. 182-188 (1980).

8. I. Hardtke, *Kinetics for key-frame interpolation,* M.Math Thesis, Univ. of Waterloo, Ont. Canada (1987).

9. P. Baehmann, M.S. Shephard, R.A. Ashley, and A. Jay, Automated metalforming modeling utilizing adaptive re-meshing and evolving geometry, *Computers and Structures* **30**(1-2) pp. 319-325 (1988).

10. R.A. Drebin, L. Carpenter, and P. Hanrahan, Volume rendering, *Computer Graphics, ACM SIGGRAPH* **22**(4) pp. 65-74 (1988).

11. P. Sabella, A rendering algorithm for visualizing 3D scalar fields, *Computer Graphics, ACM SIGGRAPH* **22**(4) pp. 51-58 (1988).

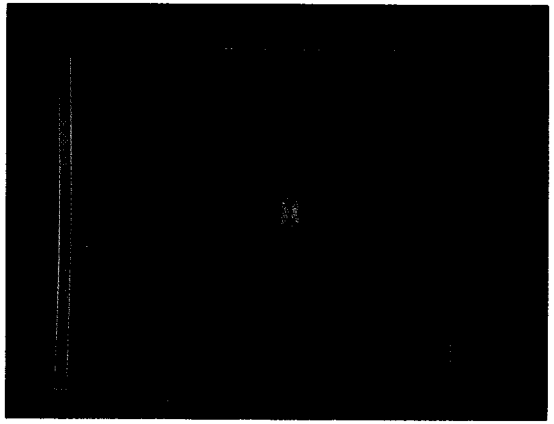


Fig. 1. Tensor field lines tracked through a 2-D domain with a 'source-sink' force pair acting near the centre. Colour coding is of limited use here due to the asymptotic nature of the field near the centre.

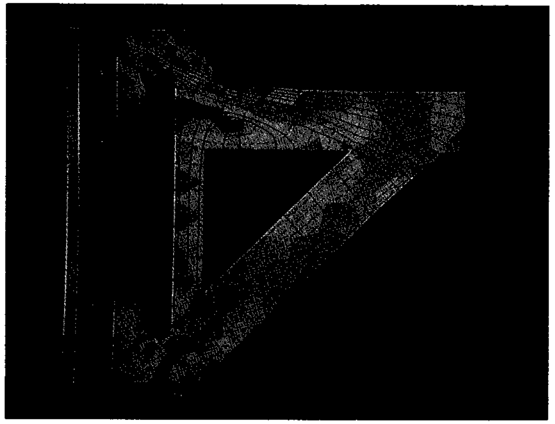


Fig. 2. Tensor field lines tracked through a 2-D wall-mounted bracket with an applied load acting downwards on the top right corner. The background is shaded according to the maximum shear stress.

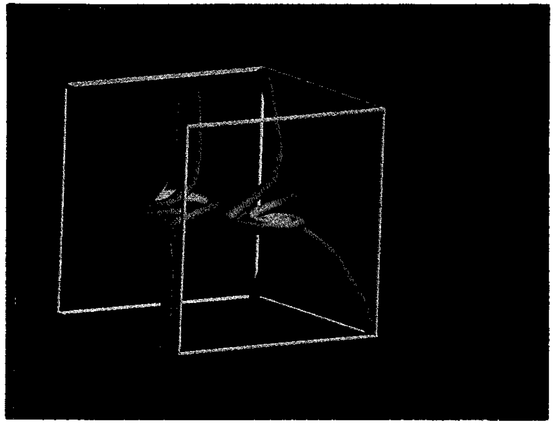


Fig. 3. 3-D space tubes, representing the orientations of molecules in the analysis of a liquid crystal defect.

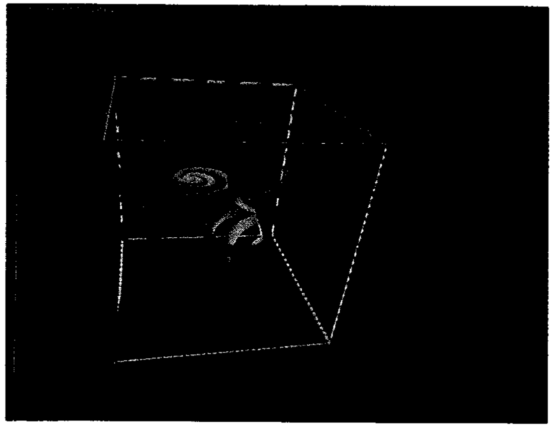


Fig. 4. Another view of the model in figure 3. Lights are used to highlight features of vector direction. Vector magnitude is unimportant in this model.

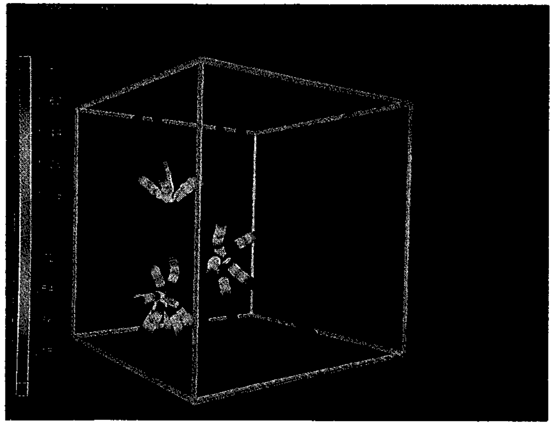


Fig. 5. Gradient streamlines of a molecular electric field, interactively located to highlight the three of six singularities of particular interest, colour coded according to gradient magnitude.

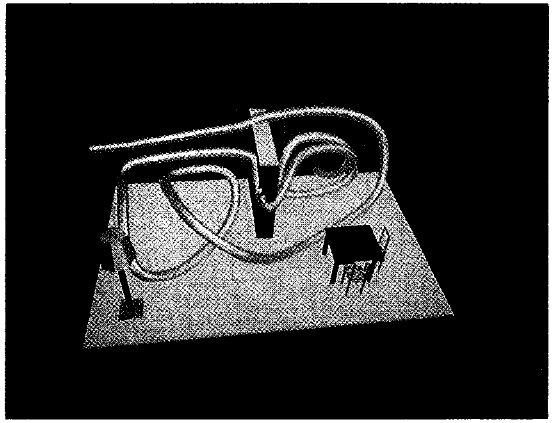


Fig. 6. An airflow 'streamtube' through a room with a baffle between the inlet near the ceiling of the left end, and the outlet in the rear wall near the floor.

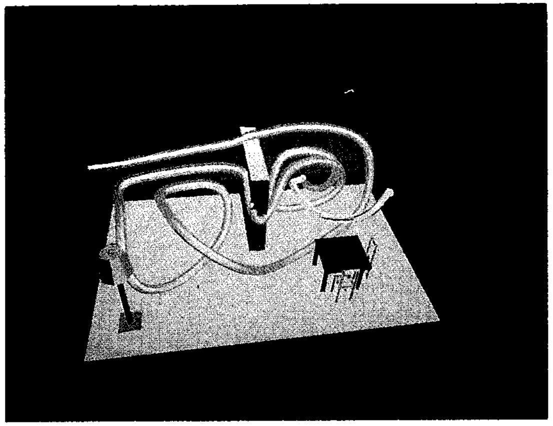


Fig. 7. Another view of the model in figure 6, with an additional tube defining the centreline of a vortex pair behind the baffle.

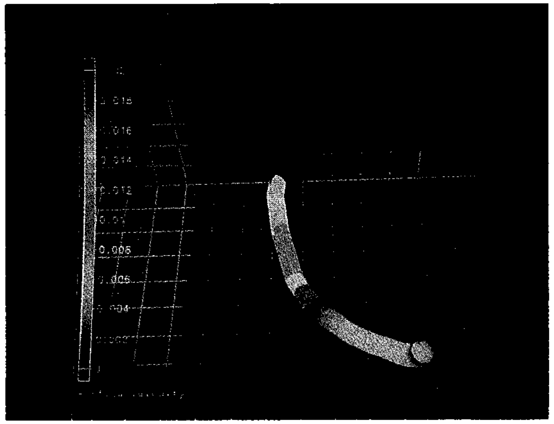


Fig. 8. Another view of the curve defining the vortex pair behind the baffle, colour coded according to velocity magnitude.

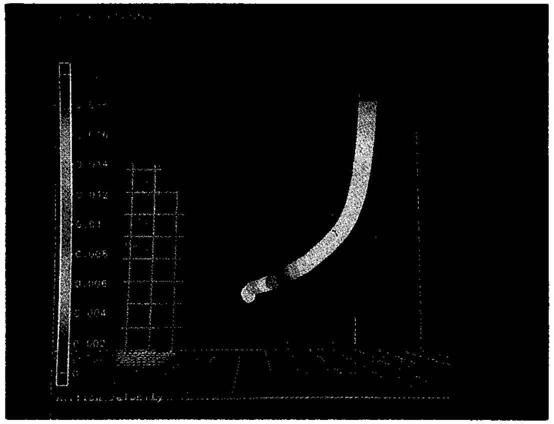


Fig. 9. Another view of the model in figure 8 is given here to illustrate its 3-D form. There are 2 point singularities – one at the roof, and one half-way down.

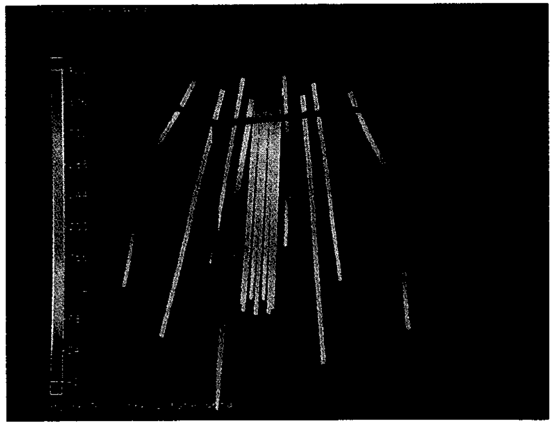


Fig. 10. Tensor field lines associated with the major eigenvalues throughout a point loaded cube, colour coded according to the major (largest) eigenvalue.

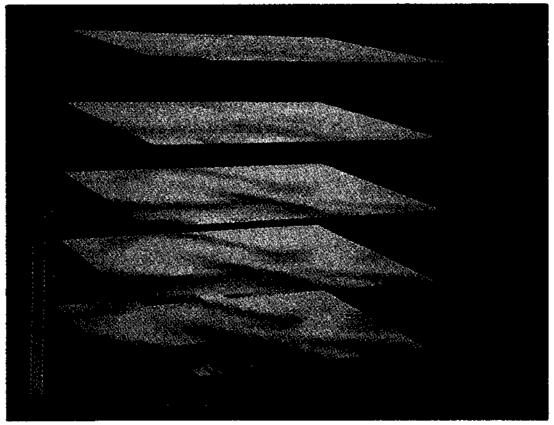


Fig. 11. Continuous 'colour' maps on sections through an atmospheric humidity field. The dark areas indicate high humidity.

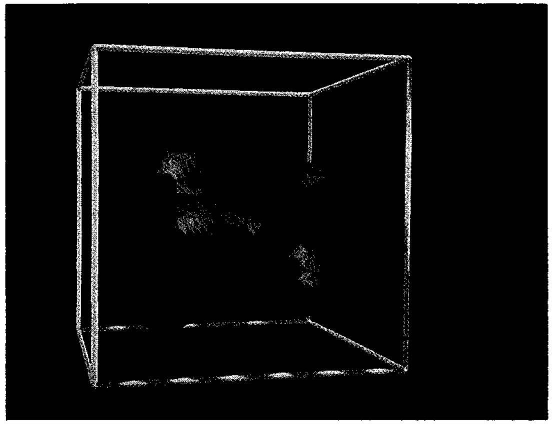


Fig. 12. A simple example of the use of isotimic surfaces with the field that is shown in Fig. 5 using gradient streamlines. Interactive 'fly-through' section planes also complement the use of isotimic surfaces in a natural way.

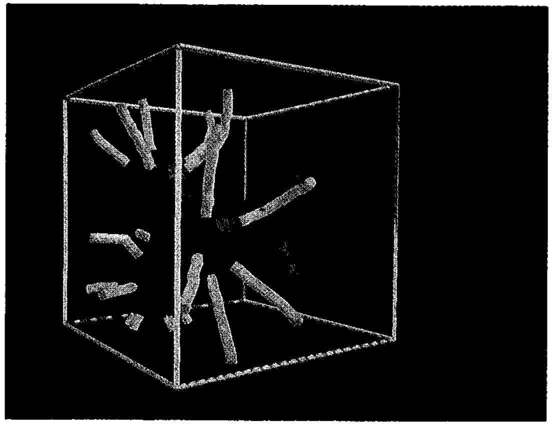


Fig. 13. The surfaces and tubes of Figs. 5 and 12 can be used together to simultaneously provide visual clues as to function values and the direction and magnitude of gradients.

Fig. 14. An isotimic surface of the pressure field corresponding to the velocity field in Figures 6 to 9. This surface highlights the low pressure zone associated with the outlet vortices.

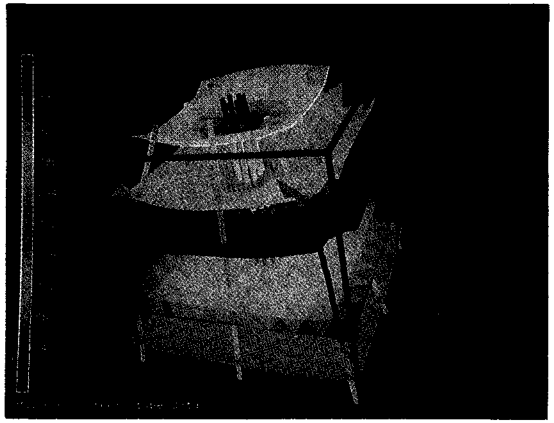


Fig. 15. 3-D tensor field surfaces associated with the middle and minor eigenvalues of the field depicted in figure 10. The surfaces are colour coded (using a lower value to colour mapping than for the tubes) according to the minor (smallest) eigenvalue.

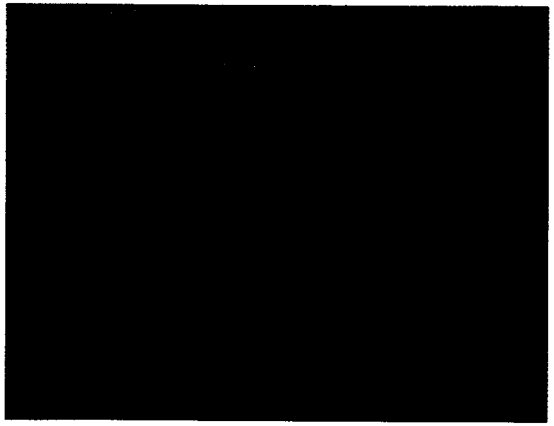


Fig. 16. An example of the use of 'bubble-trace' animation to display direction and magnitude using time as a visualization dimension.

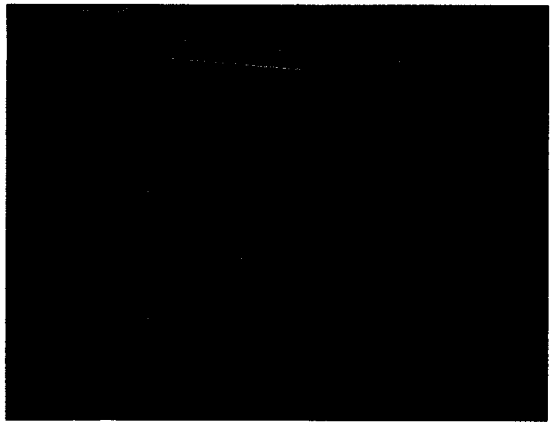


Fig. 17. Another particle trace, this time using a less simplistic animation object to aid in the visualization of the torsion of the curve.