The Safari Interface for Visualizing Time-dependent Volume Data Using Iso-surfaces and Contour Spectra

Lutz Kettner [a,*,1], Jarek Rossignac [b,2], Jack Snoeyink [c,3]

[a] *Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany*

[b] *College of Computing and GVU Center, Georgia Institute of Technology, 801 Atlantic Drive, Atlanta, GA, USA 30332–0280*

[c] *Department of Computer Science, UNC Chapel Hill, CB 3175 Sitterson Hall, Chapel Hill, NC, USA, 27599–3175*

Abstract

We describe a geometric basis for the visualization of time-varying volume data of one or several variables as they occur in scientific and engineering applications. We demonstrate a prototype interface for gridded data, extending the contour spectrum interface of Bajaj, Pascucci, and Schikore to higher dimensions and to topological properties that are not decomposable. And we explore the data structure

Key words: Scientific visualization, computational topology, simplicial mesh
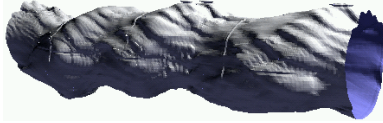
Fig. 1. Iso-surface from JETSTREAM data

1 Introduction

Many areas of science and engineering capture large data sets from a four-dimensional (4D) space-time domain. These data sets represent the evolution through time of quantities such as temperature, pressure, and flow velocity that are measured or computed on a dense grid of sample points in three-dimensional (3D) space for a series of time steps. The visual exploration of these data sets is a primary means for scientific discovery. Unfortunately, the data complexity, i.e., the number of samples and time-steps, far exceeds what can be fully downloaded and viewed in interactive sessions.

For example, a typical full-scale simulation of atmospheric turbulent flow or of the combustion in an engine must be conducted at a high-performance computing center (HPCC) and produces hundreds of gigabytes of data. Currently, researchers request individual slices, iso-surfaces, or simple animations of the evolution of a color-coded cross-section through time or space, which must be computed on the HPCC and downloaded for interactive inspection on the client [36]. The process is time consuming, and reduces the scientist's ability to discover phenomena that occur outside of the requested surfaces or animations.

Because each cross-section or iso-surface is defined by two parameters (say, pressure p and time t), the user cannot request a single linear animation that will visualize the entire 4D data set. The alternative of simultaneously viewing all cross-sections or iso-surfaces via cumulative projections [26] produces blurred and often unsatisfactory images unless a selected set of iso-surfaces is rendered with sufficiently high opacity to make them stand out.

An interactive environment, where the operator controls both time and pressure, could instead permit one to follow significant features (such as the pressure wave due to an explosion) as they evolve through time and space. Furthermore, it could eliminate the need to process, render, and inspect portions of the data that are insignificant. We believe that the challenge posed by the interactive exploration of large, time-dependent data sets is to provide the operator with visualization tools, and the context to decide "when and where" to look.

Bajaj, Pascucci and Schikore [2] proposed *contour spectra* to give a global overview of properties of all iso-surfaces. In their paper, they developed an
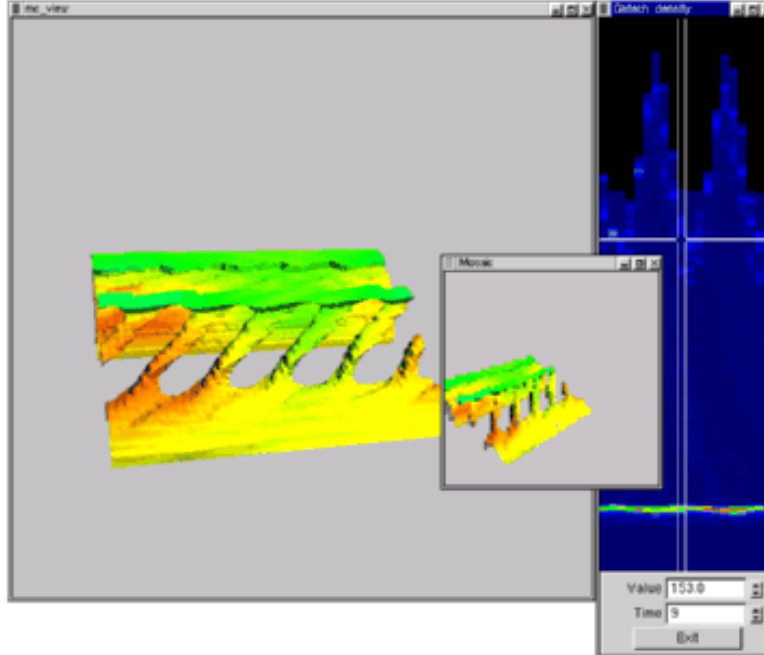
Fig. 2. The prototype user interface visualizes a density iso-surface, pop-up preview, and control plane for the Combustion data set. The iso-surface is colored by a second variable, absolute momentum, and the control plane by the number of connected components.

interface to 3D volume data that guides the selection of interesting iso-surfaces by presenting the user with a graph of properties of iso-surfaces for various threshhold values (particularly decomposable properties, such as surface area or enclosed volume) They demonstrated the feasibility for non-decomposable properties, such as the contour tree, and for 4D data. We extend their work by tackling both simultaneously.

We focus on interactive exploration of iso-surfaces, see Figures 1 and 2 for examples. We call our contour spectrum display a *control plane*, which guides the user in a safari through the most promising (p, t) parameter pairs. Each point in the contour spectrum corresponds one-to-one to an iso-surface. Specific properties of the iso-surface can be illustrated in the control plane using color, symbols and other methods. Furthermore, the user can leave annotations, markers, and traces in the control plane documenting an interactive session and its discoveries.

In addition, we use a thumb-nail *preview window* that follows the mouse pointer interactively in the control plane and that shows a small preview image of the iso-surface at that point. The fast and small preview image provides a quick overview while selecting the point under the mouse pointer loads the iso-surface into the interactive 3D viewer for detailed inspection.

We report on our experience with this prototype Safari interface, illustrated

in Figure 2. After a comparison with previous work in Section 2, we give an overview of our interface in Section 3, then give more detail on the control plane and its color-coding with the number of connected components in Section 4, and the preview window in Section 5. We discuss example data sets and results for the color-coding of the control plane in Section 6. We close in Section 7 with observations on using simplicial meshes as a geometric basis for a client/server visualization system that extracts individual iso-surfaces and summary overviews from time-varying volume data.

2 Previous Work

Our Safari prototype is designed to demonstrate possible user/client interaction for a web-based, client/server visualization system for time-varying volume data. There have been a number of web-based systems aiming towards interactive visualization of static volume data. Bethel [6] uses the terms render-local (render on the client from raw data) and render-remote (render on the server and transfer an image) to characterize most systems, but points out that increases in bandwidth allow client and server to share the rendering task.

Various systems [5,21,20,22,37,35,51,55] span the range from render-local to render-remote. Bailey and Michaels' VizWiz [37] is a Java-based example of render-local. Trapp and Pagendarm [51] presented a render-remote, web-based, flow-visualization service in which an HTML client would request VRML output from a server. Ma and Camp [35] assembled an integrated, parallel, render-remote system where the server sent pipelined, compressed images. They studied different compression schemes and display clients, so their system approaches render-both. Engel and co-authors [21,22] combine local and remote visualization in their application for volume visualization in a medical environment. Bethel's Visipult system [5] runs an IBR-assisted parallel renderer on a server that computes a scene graph for the client to display. Another system of Engel, Gross, and Ertl [20] renders iso-surfaces progressively: their client has sliders to request values for the iso-surface threshold and level of resolution. The server constructs level-of-detail structures in VRML or Java3D, which the client can display once they are received.

The render-local/render-remote debate puts the focus on the image that is to be displayed. When the data is drawn from a four-dimensional domain, the system must focus equally or more on giving the user some guidance for which images to display. We try to give the user a global view of the data by using a simple preview window and by using contour trees, which have been used to find seed sets for iso-surfaces.

Marching Cubes [33], with the *asymptotic decider* [39] to resolve ambiguities, has become a standard tool for iso-surface extraction from grids in 3D. Many have noted [47,28] that a simple variant, often called Marching Tetrahedra, generalizes immediately to higher dimensions and is free of topological ambiguities. Recently, simple extensions of marching cubes to higher dimensions have also been given [7]. There have been several recent works on data structures to support efficient extraction of iso-surfaces from volume data [1,13,15,53,31,54]. Some of these have been extended to time-varying volumes [42,44,45,49].

Bajaj, Pascucci and Schikore [2] developed *contour spectra* to efficiently compute decomposable properties of iso-surfaces in static volume data. Decomposable properties, which include properties such as volume and surface area, can be computed by aggregating the values computed on each grid cell. This aggregation can be extended to time-varying data: if a spline function $area_\nu(P, T)$ is used to represent surface area in grid cell ν as a function of P and T, then the spline function for the total surface area can be obtained by summing the splines for each grid cell.

There are important topological properties that are not decomposable, such as the number of connected components. We can still compute these efficiently using contour trees. Contour trees were introduced by Boyell and Ruston [10] as a summary of the evolution of contours on a 2D map, and used by Freeman and Morse to find terrain profiles in a contour map [23]. The contour tree is related to Morse theory in mathematics [4,38,46], which studies the changes in topology of level sets as the parameter changes. Contour trees have been used for image processing and geographic information systems, but principally in 2D applications. It appears that van Kreveld et al. were the first to identify the applicability to iso-surfaces [53]; they advocated using the contour tree to generate a seed point on each component, from which the entire component could be traced; see also [16,3,45].

The contour spectrum, displayed as a control plane, can be compared with other ideas that give users a global picture of large or multidimensional data sets, such as fish-eye views [24,29], or image graphs that arrange and connect different views of a 3D volume set [34]. The preview window has similarities to the magic lens interface technique [8,29,48].

3   System Overview

We assume that simulation data is represented by a vector-valued function $f\colon D \to \mathbb{R}^k$, where the domain $D \in \mathbb{R}^4$ is parameterized by position x, y, and z, and time t. The k values in the range can consist of observed values, such as pressure, temperature, axial velocities, etc., or derived values, such

as vorticity or approximated derivatives. We will use pressure, denoted p, as the example throughout this paper. We will also use the convention that the lower case letters x, y, z, t, and p denote free variables or axes in space and the upper case letters X, Y, Z, T, and P denote chosen values. Thus, $z = Z$ indicates a hyperplane perpendicular to the z axis, going through the point on this axis that is Z from the origin.

The display partitions the data by dimension into a *control plane*, which represents the space of all iso-surfaces, and a *viewing volume*, which displays a chosen iso-surface. We begin with a real-valued function $f(x, y, z, t)$ defining the pressure over our domain $(x, y, z, t) \in \mathbb{R}^4$. The graph of the function f is a surface in a 5-dimensional space:

$$\{(x, y, z, t, p) \in \mathbb{R}^5 \mid p = f(x, y, z, t)\}.$$

We obtain an *iso-surface* by restriction along two different dimensions, pressure $p = P$ and time $t = T$:

$$iso(P, T) := \{(x, y, z) \in \mathbb{R}^3 \mid P = f(x, y, z, T)\}.$$

The iso-surface is a 2D surface embedded in three-space; several examples are shown in Table 1.

The definition of an iso-surface partitions the five data dimensions into the three dimensions, (x, y, z), for the viewing volume and the remaining two dimensions, (p, t), for the control plane. The user can select viewing parameters by selecting a position on the control plane

We have created a prototype implementation for the visualization to study the usability of our approach to data navigation. Snapshots of the prototype in Figure 2 and color plate Figures 9 and 10 show the three-dimensional viewing volume on the left, the two-dimensional control plane on the right, and a small preview window hovering above the control plane.

Each point (P, T) in the control plane corresponds to the particular iso-surface $iso(P, T)$, see Figure 3 for an illustrative example. The user can select the iso-surface by a mouse click in the control plane (direct interaction), by cursor keys, or by numerical entry in dialog boxes (conventional indirect interaction).

To find interesting iso-surfaces for investigation, the control plane can be annotated with navigational aids. We give examples in the next section. In addition, we present the user with a small preview window that hovers close the current mouse pointer whenever the mouse pointer is in the control plane. We explain details of the preview window in Section 5.
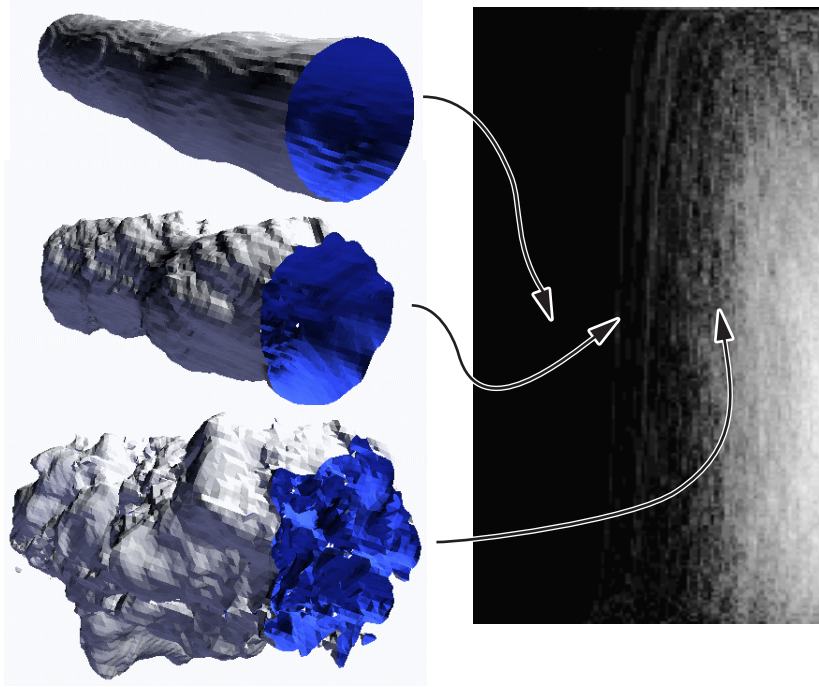
Fig. 3. Control plane (right), showing numbers of connected components in gray, and three corresponding iso-surfaces (left) for JETSTREAM data

The prototype also features selected conventional visualization techniques, e.g., it can show cross sections, several iso-surfaces at once, and it can color one iso-surface based on the values of a second volume data set. All images in this paper are produced with our program.

The prototype is based on original and subsampled volume data, i.e., regular 3D grid samples for several time steps. A cache management keeps as many time steps in main memory as possible. Marching cubes or marching tetrahedra are used to extract iso-surfaces [33] and an octree per time step speeds up the search for voxels that contribute to the iso-surface [54]. OpenGL display lists can optionally be used to speed up rendering once an iso-surface is selected.

Our test machine was a Linux Laptop with a 400 MHz Mobile Pentium II, a 12GB hard disk, and 256 MB main memory. Graphics was rendered using Mesa without hardware graphics accelerator.

We can examine iso-surfaces in the 3D viewer at moderate convenient frame rates of 1 to 5 frames per second for data sets of size of $64^3 \times 100$. Contemporary workstations are more powerful and include hardware graphics accelerators, however there is still a considerable gap to the data set size we are aiming for to visualize, such as $256^3 \times 100$ or larger. So our prototype serves also as a benchmark between this conventional approach and future work aimed to bridge the gap.

4   Control Plane

There is a one-to-one correspondence between a point in the 2D control plane and an iso-surface. This straightforward observation, illustrated in Figure 3, is the basis for our use of the control plane to provide guidance for the user's exploration of the data.

The control plane can be used to display summary information about the corresponding iso-surface. In our prototype we can do this in two ways: first, by loading an image for display on the control plane and second by loading a set of precomputed iconic views of iso-surfaces for display in a preview window that follows the mouse cursor over the control plane.

The control planes in this paper display the number of connected components in the corresponding iso-surface, using a simple color ramp from black (zero) through blue, green, yellow, and red (maximum). Changes in the number of connected components can be related to the behavior of the phenomenon under study. For example, in our jet dynamics simulation data sets described in the next section, an increase in the number of connected components indicates that a flow has developed instability and is shedding vortices at some chosen threshold levels. Changes can also indicate the behavior of the modeling; we give more detail in an example of Section 6 that studies oil displacing water in soil. As water is displaced the grid-based modeling tends to produce small fluctuations in the values near the interfaces. Due to the linear interpolation, these fluctuations reveal themselves for threshold values near the critical value as connected components that surround a single data point. Almost by coincidence, this causes the control plane to reveal the change in interface value over time.

The number of connected components is a particularly good example of the type of data that can be displayed on a control plane, because we can determine the number without computing all iso-surfaces. We spend the remainder of this section describing how we do so using a *contour tree*, which records the evolution of components of the iso-surface for a fixed time as threshold value v changes. The edges of a contour tree correspond to components, and vertices to the threshold values v at which these components appear, merge, split, or disappear. (The components of the iso-surface may also change genus, which affects their topology but not their number; such events are not recorded in the contour tree.)

We mentioned earlier that the contour tree is related to Morse theory in mathematics, which studies the changes in the topology of level sets as the threshold value changes. Points at which the topology of the level sets change are called *critical points*. Morse theory requires that the critical points be isolated—that

8

they occur at distinct points and values—which can be assured by (conceptual) perturbation of the data values so that they are distinct. We are interested only in the critical points that change the number of connected components.

Van Kreveld et al. [53] gave an algorithm to compute a contour tree for a 3D mesh with N points. Their algorithm swept through all iso-surfaces and used $O(N^2)$ time. Tarasov and Vyalyi [50] reduced the time to $O(N \log N)$ by doing three sweeps: two to classify critical points and one to build the tree. Their approach also involved increasing the mesh size by a factor of 30–300, which rendered it impractical. Carr et al. [12] simplified the algorithm so that the first steps produced partial trees without sweeping iso-surfaces, and the third simply merged these partial trees.

Once contour trees are computed, it is a simple matter to traverse the tree edges and gather the number of connected components. With an existing implementation for the contour tree, it takes minutes to compute numbers of connected components for all iso-surfaces in our smaller example data sets and a few hours for the largest.

Other guiding images could be computed for the control plane to illustrate other properties of corresponding iso-surfaces. Examples include topological properties, such as the number of tunnels; statistical properties, such as distribution characteristics of the values of other data sets; or visual properties, by displaying the preview images as *small multiples* [52] as in the color plate Figure 11. Of course, the efficient algorithms of Bajaj, Pascucci and Schikore [2] for contour spectra apply to compute decomposable properties such as surface area for or volume enclosed by a time-varying iso-surface. It is interesting to ask what other non-decomposable properties, in addition to the number of components, can be computed without computing iso-surfaces.

To display an additional computed value, the control plane may also be presented as a colored terrain, where at each point (P, T) the elevation of the terrain represents one characteristic of $iso(P, T)$ and the color on the terrain represents another. Finally, as mentioned in the introduction, the user can trace a path on the control plane to indicate a sequence of desired views, and can leave annotations, trails, and markers to record his or her journey of exploration and point out important views.

5   Preview Window

We can provide a small preview window that is shown continuously at the current mouse pointer position in the control plane, see Figure 2. The window follows the mouse pointer instantaneously and gives a preview of the iso-

surface that corresponds to the current mouse pointer position. This preview facilitates a quick interactive overview, while selecting the current location in the control plane switches to the corresponding detailed view in the 3D viewer for iso-surfaces for further detailed inspection. The preview window has similarities to the magic lens interface technique [8,48].

In the current prototype, the preview is realized as a low resolution image that has been pre-computed for a fixed viewpoint. The preview images are stored as a mosaic of small images in a large image. For example, for the COMBUSTION data set the preview images have 66×66 pixels, and we store them for 20 time steps and 256 different density values, a total of 1320×16896 pixels for the mosaic. Image 11 in the color plate shows a part of that mosaic.

We can pre-compute the preview images for our data sets in the range of several minutes to a few hours, depending on the data set size. The next section gives the details of several data sets that drove our research.

6   Data Sets and Results

JETSTREAM   TURBULENTJET   CONVECTION   OIL&WATER   COMBUSTION

Name	dim x	dim y	dim z	dim t	# values	voxel	total size
JETSTREAM	256	256	256	100	1	byte	1680 MB
TURBULENTJET	104	129	129	150	1	float	1040 MB
CONVECTION	128	128	64	30	1	byte	120 MB
OIL&WATER	56	50	50	16	2	double	36 MB
COMBUSTION	110	64	81	20	7	float	320 MB

Table 1
Example volume data sets.

We illustrate our prototype on five example data sets whose characteristics are summarized in Table 1. In addition, we use subsampled versions of these data sets. Figure 8 in the color plate shows, for each data set, the images in the control plane illustrating the number of connected components in a false color encoding: black for zero or one component, dark blue for a few components, going over green and yellow to red for the maximal number of connected components. Time extends to the right and the iso-surface value
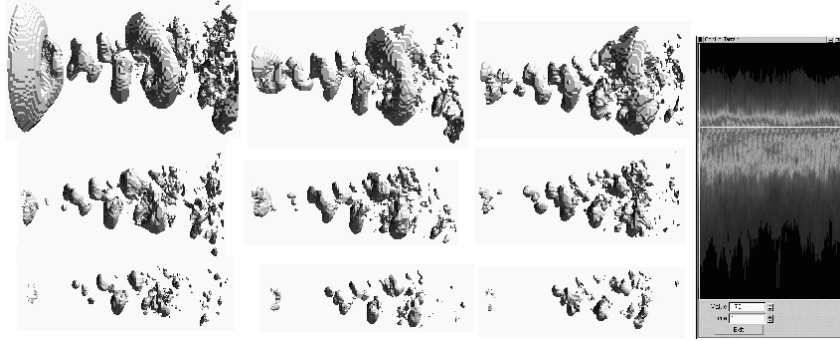
Fig. 4. Nine iso-surfaces and the control plane from a pulsed flow data set TURBU-
LENTJET.

increases from bottom to top. The time axis is stretched for the smaller data
sets. We discuss phenomena visible in these images.

Our first data set JETSTREAM[4], in Figures 1 and 3, contains the result of a
flow simulation studying Helmholtz instability of a jet stream introduced into
a stable medium. The data provides concentration values of a passive tracer
material initially placed in the jet-stream that spreads over time. (Lines that
can be seen on the iso-surfaces are in the data set, probably as artifacts of the
computational methods that generated the data.) The image in the control
plane illustrates where turbulence on the originally smooth, connected surface
becomes unstable and creates more and more small disconnected parts of the
iso-surface.

The second data set TURBULENTJET, shown in Figure 4, contains the re-
sult of another turbulent jet simulation. This is a pulsed jet in steady state,
whereas JETSTREAM was introduced into a stable medium. Thus, the control
planes show quite different characteristics for these two jets. The image for
TURBULENTJET is more regular, indicates the pulses, and shows the range of
iso-values that are of interest for inspection.

The third data set, CONVECTION[5], shown in Figure 9 in the color plate,
contains the result of a numerical simulation of convection flow in earth's
crust. Two iso-surfaces for different temperatures show hotter magma in red
and cooler magma in blue; one can watch the hot magma rising and cold
magma falling over time. The two identical control planes show the different
iso-value settings. The control plane image shows the interesting range of iso-
values, similar to one for the TURBULENTJET data set.

The fourth data set OIL&WATER, shown in Figure 10 in the color plate,

---

[4] `http://www-unix.mcs.anl.gov/~hudson/htmlbase/jet.html`; also available
from `http://www.avtc.org/`
[5] `http://wwwvis.informatik.uni-stuttgart.de/~engel/`, also see
`http://earth.uni-muenster.de/geodynamik/index.html`

contains the result of a pollution study of soil. In the simulation an oil spill displaces water from the soil. The data provides concentration values for both, oil and water. The soil contains gravel that is not available as separate data but it is visible as stable empty pockets in the 4D volumes of Figure 10. This figure shows an actual screen shot of the prototype used for all images, with one viewing window and two control planes. Control planes for water and oil are synchronized in time, independent in density. The left control plane shows density values, decreasing over time, at which water has iso-surfaces with many connected components. In the viewing volume, these components will mostly be noise, attributable to discretizing the data to fixed precision. Nevertheless, the control plane still displays the useful information that water is being forced out.

The fifth data set, COMBUSTION [6], shown in Figure 2, contains the result of a combustion simulation. The data provides seven variables; density, energy, flame location, turbulence kinetic energy, and three components for a momentum vector. All images using this data set show the density variable, colored by the absolute value of the momentum. The simulation aims for lean combustion where a fuel scarce flame optimizes the use of fuel and reduces exhaust. In this simulation of a gas turbine the fuel enters the chamber already mixed with air. The flame is not stable in this simulation but oscillates. This happens also in practice and can even extinguish the flame. The oscillation and interesting areas with several components can be seen in the image in the control plane. The density iso-surface actually exhibits the interesting topological feature of tunnels. They occur right before we have several connected components. The number of tunnels would be another interesting color-coding for the control plane.

7   A Geometric Basis for Visualizing Time-varying Volume Data

We built this prototype to determine whether splitting the data dimensions into a control plane and a 3D viewer would lead to an intuitive user interface for navigating time-varying volume data sets. This interface can serve as a display client in a client/server architecture of a visualization system, where the visualization server stores the entire data set, computes an initial iso-surface and overview images for the control plane and preview windows, then receives updated (P, T) values from the client and produces the corresponding updates to the client's iso-surface. In this section, we briefly describe geometric foundations for this architecture based on simplicial meshes. With Ajith Mascarenhas of UNC Chapel Hill, we are currently evaluating tradeoffs and extensions of these foundations.

---

[6] `http://oriole.ae.gatech.edu/ccl/`

We begin with a number of standard definitions so that we can formally introduce the pentamesh and the interpolation function $p = f(x, y, z, t)$.

A set of $k+1$ points, $\{V_0, V_1, \ldots, V_k\} \subset \mathbb{R}^m$, is said to be *affinely independent* if the vectors $V_1 - V_0$, $V_2 - V_0$, $\ldots$, $V_k - V_0$ are linearly independent. The *affine hull* is defined by linear combinations with real weights α_i that sum to unity:

$$\mathit{aff}(V_0, V_1, \ldots, V_k) = \Big\{ \sum_{0 \le i \le k} \alpha_i V_i \;\Big|\; \sum_{0 \le i \le k} \alpha_i = 1 \Big\}.$$

The *convex hull*, $CH(V_0, V_1, \ldots, V_k)$, is defined similarly, with the added restrictions that $\alpha_i \ge 0$ for all $0 \le i \le k$.

A k-simplex is the convex hull of an affinely independent set of $k+1$ points. Since 2-simplices are triangles and 3-simplices are tetrahedra, the 4-simplices have been called *pentatopes* in recreational mathematics, and we use this term in this paper. Note that 1-simplices are edges and 0-simplices are points (called *vertices*). By convention, the empty set $\emptyset$ is considered a (-1)-simplex.

A k-simplex with vertices $V_0, V_1, \ldots, V_k$ defines a coordinate system on the points in its convex hull (and affine hull). Each point $Q \in \mathit{aff}(V_0, \ldots, V_k)$ can be expressed as $Q = \sum_{0 \le i \le k} \alpha_i V_i$ with $\sum_{0 \le i \le k} \alpha_i = 1$. The weights $(\alpha_0, \alpha_1, \ldots, \alpha_k)$ are called the *barycentric coordinates* of Q. One can check that either this expression is unique or the vertices of the k-simplex are not affinely independent.

Returning to geometry, any k-simplex has $(k+1)$ boundary $(k-1)$-simplices. For example, a pentatope is bounded by 5 tetrahedra, which correspond to the five ways of choosing 4 points from the pentatope vertices. In general, a k-simplex has $\binom{k+1}{j+1}$ boundary j-simplices. A pentatope, in addition to its 5 tetrahedra, has 10 triangles, 10 edges, 5 vertices, and 1 empty set.

A *simplicial complex* is any collection of simplices that is closed under the operations of intersection and taking boundaries. This is a compact way of saying that the only non-empty intersections allowed between k-simplices are along common j-simplices for $j < k$. One implication is that in k-dimensions, a $(k-1)$-simplex bounds at most two k-simplices. For example, in 3D a triangle can be shared by two tetrahedra, but there is no room to squeeze in a third. Similarly in 4D, a tetrahedron is shared by at most two pentatopes.

For a given set of points, a *simplicial mesh* is a simplicial complex whose vertices (0-simplices) are exactly the given set of points and whose union is the convex hull of these points. The Delaunay triangulation for a set of points in general position is an example of a simplicial mesh [9,17,40]. It is defined
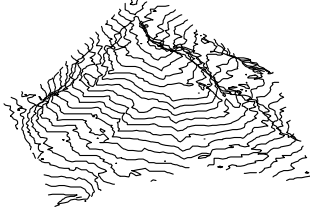
Fig. 5. Contours

by an empty sphere property: include a simplex for each subset of points that defines some sphere that is empty of other points.

Recall that our function $p = f(x, y, z, t)$ is defined on sample points in $\mathbb{R}^4$. We define a pentatope mesh, or *pentamesh*, as a 4D simplicial complex whose vertices are these sample points with their pressures. Thus, we can consider our pentamesh as a 4D surface in $\mathbb{R}^5$.

We define f by linear interpolation over each pentatope in the pentamesh. Suppose that we represent the five vertices of a pentatope π using homogeneous coordinates: $V_i = (1, X_i, Y_i, Z_i, T_i, P_i)$, for $0 \leq i \leq 4$. Then, for any point Q in 4D, $Q = (1, X, Y, Z, T, p)$, we may find the barycentric coordinates $\alpha_0, \ldots, \alpha_4$ and interpolate a pressure p by solving for these six variables in the system of six linear equations $Q = \sum_{0 \leq i \leq 4} \alpha_i V_i$. If all of the barycentric coordinates are non-negative, then Q is in pentatope π, and we accept the computed p value as the interpolated pressure at Q.

Notice that if some of the barycentric coordinate values are zero, then Q is on the boundary of possibly many simplices in the mesh. However, each of these simplices will interpolate the same values, because each combines the same vertices with the same set of non-zero weights.

7.2   From Pentatopes to Iso-surfaces

There are some simple, elegant connections from the topology of a pentamesh to an iso-surface for chosen values of pressure and time, $iso(P, T)$. Because the pentamesh can be considered as a four dimensional surface in five dimensions, and we slice away two dimensions to form the iso-surface, an analogy with topographic maps of terrain may be helpful.

Terrain data frequently comes in the form of elevation samples, z values, at points in an x, y plane. To form a terrain model, one interpolates elevations. On large maps, this is frequently done by computing a triangulation in the x, y plane and raising the vertices to their elevations; this extends the height mapping to a function $z = f(x, y)$ by linear interpolation over each 2-simplex (triangle). For topographic maps such as the one in Figure 5, one computes
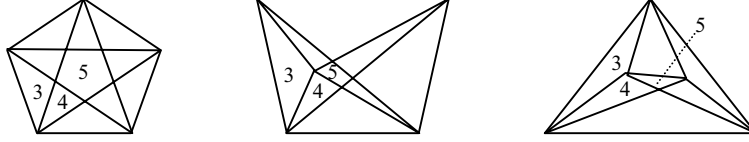
Fig. 6. Three projections of pentatopes onto the p, t plane; some regions are labelled by the number of triangles the project onto them.

contours, or iso-lines, by slicing this terrain model by horizontal planes, e.g. $z = H$.

The triangles and edges that contribute to contours at height $z = H$ are those whose projection onto the z axis contain the point H. A 2D triangle intersects the plane $z = H$ in a line segment, and an edge in general position intersects the plane $z = H$ in a point. These join together to form 1d curves in the $z = H$ plane; in fact, if one can find one triangle or edge on a contour curve, then one can trace that contour by walking around the terrain model at height H.

When the chosen height H varies, all contours move, but as long as H remains in the same intervals in the projection, the same triangles contribute to the contours and the contour topology does not change. When H passes the value of a sample point, then the set of triangles and edges contributing to the contour changes, which may also cause contours appear or disappear, join or split.

To determine iso-surfaces from the pentamesh, we have some of the same operations of projection, intersection, tracing, and updating. We begin by considering how a single pentatope contributes to an iso-surface $iso(P, T)$, based on its projection on the p, t plane. Three possible projections are shown in Figure 6.

Lemma 1 *A pentatope π contributes a convex polygon to the iso-surface $iso(P, T)$ if and only if the point (P, T) is in the projection of the pentatope π in the p, t plane. The convex polygon can be a triangle, quadrilateral, or pentagon corresponding to the number of triangles of π that contain the point (P, T), which is 3, 4, or 5 respectively.*

PROOF. Since the contribution $\rho = \pi \cap iso(P, T)$ is the intersection of two hyperplanes, $p = P$ and $t = T$, with a general convex 4-simplex in 5D, the result ρ is clearly a convex, 2-dimensional polygon in 3D. Since the pentatope (4-simplex) is bounded by 3D tetrahedra and 2D triangles, their intersections should give the 1D edges and 0D vertices that bound ρ. What remains is to determine the exact form of ρ.

Using homogeneous coordinates as in Section 7.1, consider a triangle with

vertices V_0, V_1, and V_2 and a point $Q = (1, x, y, z, T, P)$. We can solve the six equations of $Q = \sum_{0 \leq i \leq 2} \alpha_i V_i$ for the six variables x, y, z, α_0, α_1, α_2 as long as the three vertices V_0, V_1, and V_2 are not collinear when projected to the p, t plane. And this we may assume by simulating simplicity, if necessary [19].

Notice that $(\alpha_0, \alpha_1, \alpha_2)$ are the barycentric coordinates of the projection $Q|_{pt} = (P, T)$ in the p, t plane, and that (x, y, z) is the corresponding location of a point $Q|_{xyz}$ in the viewing volume. If, and only if, the α_is are all non-negative, then the computed point Q projects to the point (P, T) in the triangle, and is a vertex of $iso(P, T)$.

Therefore, there is a vertex of $iso(P, T)$ for each triangle of pentatope π that is stabbed by (P, T) in the projection. As Figure 6 suggests, and the reader can check, this contributes a triangle, quadrilateral, or pentagon ρ to $iso(P, T)$.

The topology of the iso-surface $iso(P, T)$ is determined in a straightforward manner from the topology of the pentamesh.

Lemma 2 *The local topology of the pentamesh determines the order of polygons and edges around vertices in the iso-surface $iso(P, T)$.*

PROOF. Every tetrahedron in a pentamesh bounds two pentatopes, except for those on the mesh boundary, which bound only one. Within a pentatope, each triangle is contained in two of the tetrahedra bounding π.

If we fix a triangle τ and look at the pentatopes and tetrahedra containing τ, we find either a cycle of pentatopes alternating with their bounding tetrahedra, or a sequence from one boundary tetrahedron to another, through an alternating sequence of pentatopes and bounding tetrahedra.

In the iso-surface $iso(P, T)$, therefore, the vertex V corresponding to τ has a cycle or sequence of 1D edges and 2D polygons that are incident upon V. These aspects of the local topology of the pentamesh are inherited in the iso-surface $iso(P, T)$.

With this lemma, and a data structure that gives access to the pentatopes and tetrahedra bounded by triangles containing a point (P, T), it is not difficult to trace out a connected component of the iso-surface $iso(P, T)$; all we need is a place to start. We will come back to this in Section 7.4.

We would like to make one final observation based on the computation in the proof of Lemma 2. The server can send the vertices of $iso(P, T)$ as linear functions of P and T. As the client navigates in the p, t plane, it requires

updates from the server only for those iso-surface vertices that correspond to newly stabbed triangles.

Lemma 3 *If a user changes the (P,T) viewing parameters without leaving the projection of a triangle τ, the position of the corresponding vertex of iso(P,T) moves linearly in P and T.*

PROOF. From the proof of Lemma 2, one can determine that the position (x,y,z) has a linear dependence on P and T that is determined by coordinates of the triangle vertices V_0, V_1, and V_2. One can derive the following matrix expression for the position of the iso-surface vertex:

$$
\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} X_0 & X_1 & X_2 \\ Y_0 & Y_1 & Y_2 \\ Z_0 & Z_1 & Z_2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ P_0 & P_1 & P_2 \\ T_0 & T_1 & T_2 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ P \\ T \end{bmatrix}.
$$

These correspondences allow the client to maintain a polygonal iso-surface while the server uses the pentatope mesh and tracks the viewing parameters p and t.

7.3   A Corner Table Data Structure for the Pentamesh

The data structure chosen to represent a simplicial complex has a significant impact on the efficiency of its manipulation. Some structures store too little: a simple list of pentatopes would require a search for adjacencies. Other structures store too much: the entire facial lattice [18] or the elegant generalizations of the quadedge [25] to higher dimensions, such as Leinhardt's n-G maps [30] and Brisson's cell-tuples [11], require many pointers.

Since our primary operation is to "slice" the pentamesh, generating an iso-surface by fixing parameters P and T, we do not need the entire facial lattice, but will be satisfied if we can access adjacencies between pentatopes, their bounding tetrahedron, and their triangles. On the other hand, it would be beneficial if operations used to walk the pentamesh, such as retrieving Lemma 2's cycle of tetrahedra and pentatopes around a triangle, were particularly efficient.

We therefore use a corner-based data structure that is a refinement of the simplex-based structure of Paoluzzi et al. [41]. A *corner* is the use of a vertex in a pentatope; each pentatope has five corners. Each corner i stores a pair of indices; one to $v(i)$, the vertex used at that corner, and one to $c(i)$, the corner

	cor	vert	opp
	id i	$v(i)$	$c(i)$
	0	A	9
	1	B	-
π	2	C	14
	3	D	-
	4	E	-

	cor	vert	opp
	id i	$v(i)$	$c(i)$
	5	B	-
	6	C	10
ρ	7	D	-
	8	E	-
	9	F	0

	cor	vert	opp
	id i	$v(i)$	$c(i)$
	10	A	6
	11	B	-
σ	12	D	-
	13	E	-
	14	F	2

Fig. 7. Corner structure

of the neighboring pentatope that is opposite $v(i)$. A similar data structure has been advocated by Rossignac and colleagues for their implementation of the Edgebreaker algorithm for compressing 3D meshes [43].

We list corners ordered first by their pentatopes so that knowing the *corner index*, i, gives easy access to the *pentatope index*, $\lfloor i/5 \rfloor$. We use the vertex index $v(i)$ as a secondary key, so that within a pentatope the vertex indices increase. Let us also define the *corner number* $\#(i) = (i \bmod 5)$ so that $i = 5\lfloor i/5 \rfloor + \#(i)$. By pointing to corners, rather than to pentatopes, and having a canonical ordering of the corners in each pentatope, we can navigate the structure without having to test which face we entered from and where we are going.

Even this pentamesh data structure is large; in our pentameshes a vertex is used as corner about 150 times on average.

Consider a small example as we look at some of the operations that this structure supports. On six vertices, A–F, form three pentatopes $\pi = ABCDE$, $\rho = BCDEF$, and $\sigma = ABDEF$ with a cycle around $\triangle BDE$. The corners are listed in Table 7.

Find pentatope vertices: From corner index i, obtain pentatope index $j = \lfloor i/5 \rfloor$. The vertices are $v(5j)$, $v(5j+1)$, ..., $v(5j+4)$.

Walk through bounding tetrahedron: Corner index i also indicates the tetrahedron obtained by removing vertex $v(i)$ from the pentatope $\lfloor i/5 \rfloor$. The index $c(i)$ indicates the same tetrahedron obtained in the adjacent pentatope. Notice that walking back is idempotent: $c(c(i)) = i$.

Find pentatope orientation: Our use of a canonical ordering for vertices implies that not all of our simplices are oriented according to a right-hand rule. To obtain the orientation for a (non-degenerate or properly perturbed) pentatope with vertex sequence $V_0, V_1, \ldots, V_4$, we may compute the sign of

the 5×5 determinant whose rows are the homogeneous coordinates of these vertices (sans pressure values). Once we know our orientation, however, we can determine the orientation of a neighbor $c(i)$ by reversing orientation if the difference $(\#(i) - \#(c(i)))$ is even. (Proof: The orientation reverses when we replace $v(i)$ by $v(c(i))$ at row $\#(i)$ in the matrix, then we may need to do row interchanges to move row $\#(i)$ to row $\#(c(i))$.) Thus, maintaining orientation has a small overhead.

Cycle about triangle: Because of the canonical ordering of corners, when we cycle around a fixed triangle, we can move from pentatope to pentatope, and always know which three vertices in the pentatope list are the triangle corners. Row interchanges moving i to $c(i)$ will simply move some corners up or down. Specifically, pentatope $\lfloor i/5 \rfloor$ corner number j goes to pentatope $\lfloor i/5 \rfloor$ corner number $j - 1$, if $\#(i) < j \le \#(c(i))$, or to $j + 1$, if $\#(c(i)) \le j < \#(i)$, or else remains at j.

In the example of Table 7, we can cycle around $\triangle BDE$. We let the pentatope indices $\pi = 0$, $\rho = 1$, and $\sigma = 2$, so that we can write corner indices as a combination of a pentatope index and a corner number, 0–4. We start in pentatope π with vertices $ABCDE$. The triangle vertices are 1,3,4, so 0 and 2 indicate the neighbors; we follow $c(5\pi + 0) = 5\rho + 4$. To move corner number 0 to 4 by row interchanges, the other vertices each move up one, so corner number 2 changes to 1 to indicate the next neighbor, $c(5\rho + 1) = 5\sigma + 0$. Interchanging 1 and 0 does not renumber corner 4, which indicates the next neighbor $c(5\sigma + 4) = 5\pi + 2$. Row interchanges that move 4 to 2 do not renumber corner 0, so we have returned to $5\pi + 0$.

If it is important to store all pentatopes with positive orientation, then we can modify the structure of each pentatope as follows. After listing the five corners in order of vertex index, interchange the first two rows, if necessary, to make the orientation positive. Operations can compare these two vertex indices to determine if they have been reversed before proceeding.

7.4 Arrangements on the Control Plane

The control plane represents every iso-surface $iso(P, T)$ as a point, and so it is the natural space in which to consider tracking how the user views the data. We define two natural arrangements that can be built on the control plane: the projection of the 1-skeleton of the pentatope mesh, and the projection of silhouette edges.

The *1-skeleton* of a simplicial complex is the set of all 1-dimensional simplices, or edges. Thus, the inked parts of Figure 6 are the projection of 1-skeletons. The cells of this arrangement correspond to points with the same set of pro-

jecting pentatopes, triangles, and tetrahedra, which determine the iso-surface, according to Lemma 1.

The *silhouette edges* are the edges of the pentamesh for which all incident triangles lie to one side. When the point (P, T) moves across a silhouette edge, a new connected component appears or disappears from $iso(P, T)$. Rather than building the arrangement of the entire 1-skeleton, we could build the arrangement of the silhouette edges only. In this arrangement, it is easy to color the control plane by the number of connected components.

These graphs can be linked to the generation of the iso-surface from the pentamesh. The idea of marching cubes and marching tetrahedra is to visit all cube or simplex elements and decide which contribute to the iso-surface. In 3D, researchers have used several data structures to identify the contributing cubes or simplices, such as octrees in the viewing volume [54] or interval trees in the projection [14,16,32]. The projection of the 1-skeleton generalizes the interval tree ideas.

The alternative to visiting all elements is to identify a *seed* on each connected component of an iso-surface, and trace the iso-surface from that seed [27,53,12]. The projection of the silhouette edges identifies the connected components, and which seeds are needed. The operations on the pentamesh data structure allow us to trace a connected component of the iso-surface $iso(P, T)$ by traversing only those pentatopes, tetrahedra, and triangles that intersect the component. We believe that this alternative is the most promising way to generate iso-surfaces.

7.5 Storing and Updating the Client's Iso-surface

Three questions indicate several options for representing the iso-surface at the client. How much topological structure should the client keep? Should the client or server compute the iso-surface from pentatopes? Should it do so by scanning all relevant pentatopes or by tracing connected components of the iso-surface? The first two questions combine to ask whether the client should maintain a surface representation of the iso-surface, either as a list of polygons, as triangle strips, or as a planar subdivision, or whether the client should maintain the pentatopes that contribute to the iso-surface, again as a list, as a spanning forest, or as a subset of our pentamesh data structure.

The trade-offs are between structure size, transmission bandwidth, and computational demand on the client, both for initial construction and for updating as the client makes incremental changes to the viewing parameters P, T. The data structure must support drawing and update operations. For drawing, a list of triangles is sufficient, and triangle strips are ideal. The updates are

edge contractions and expansions, and triangle contractions and expansions; for these operations the extra pointers of a planar subdivision help, although they do add to the operation costs. There is ample opportunity for further research, especially in extending the various surface and mesh compression schemes and using them to reduce communication bandwidth.

Acknowledgments

References

[1] C. Bajaj, V. Pascucci, and D. Schikore. Fast isocontouring for improved interactivity. *IEEE Trans on Vis. and Comp. Graph.*, 2(1):39–46, 1996.

[2] C. Bajaj, V. Pascucci, and D. Schikore. The contour spectrum. In *Proc. 1997 IEEE Vis*, pages 167–173, October 1997.

[3] C. Bajaj, V. Pascucci, and D Schikore. Seed sets and search structures for optimal iso-contour extraction. Technical Report 99–35, Univ. Texas, 1999. `http://www.ticam.utexas.edu/CCV/papers/fhighlights.html`.

[4] T. F. Banchoff. Critical points and curvature for embedded polyhedra. *J. Diff. Geom.*, 1:245–256, 1967.

[5] W. Bethel. Visapult: A prototype remote and distributed visualization application and framework. In *Sketches*. ACM SIGGRAPH, July 2000. `http://vis.lbl.gov/projects/visapult/visapult-s2000.pdf`.

[6] Wes Bethel. Visualization dot com. *IEEE Comp. Graph. Appl.*, 20(3), May/June 2000.

[7] Praveen Bhaniramka, Rephael Wenger, and Roger Crawfis. Isosurfacing in higher dimensions. In *IEEE Visualization '00 Proceedings*, pages 267–273, 2000.

[8] Eric Bier, Maureen Stone, Ken Pier, William Buxton, and Tony DeRose. Toolglass and magic lenses: The see-through interface. In *Proc. ACM SIGGRAPH Conference*, pages 73–80, 1993.

[9] Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic Geometry*. Cambridge University Press, UK, 1998. Translated by Hervé Brönnimann.

[10] R. L. Boyell and H. Ruston. hybrid tehniques for real-time radar simulation. In *Proc. IEEE Fall Join Comp. Conf.*, pages 445–458, 1963.

[11] E. Brisson. Representing geometric structures in d dimensions: Topology and order. *Discrete Comput. Geom.*, 9:387–426, 1993.

[12] Hamish Carr, Jack Snoeyink, and Ulrike Axen. Computing contour trees in all dimensions. In *Proc 11th SODA*, pages 918–926, 2000. Accepted to *Comp. Geom. Theory Appl.*

[13] Y.-J. Chiang, C. Silva, and W. Schroeder. Interactive out-of-core isosurface extraction. In *Proc. IEEE Visualization '98*, pages 167–174, 1998.

[14] Y.-J. Chiang and C. T. Silva. I/O optimal isosurface extraction. In *Proc. IEEE Visualization*, pages 293–300, 1997.

[15] P. Cignoni, P. Marino, C. Montani, E. Puppo, and R. Scopigno. Speeding up isosurface extraction using interval trees. *IEEE Trans Visualization Computer Graphics*, 3(2):158–170, 1997.

[16] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Optimal isosurface extraction from irregular volume data. In *Proc. Volume Visualization '96*, pages 31–38, 1996.

[17] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[18] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, Heidelberg, West Germany, 1987.

[19] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graph.*, 9(1):66–104, 1990.

[20] K. Engel, R. Grosso, and T. Ertl. Progressive isosurfaces on the web. Late Breaking Hot Topics, Visualization 98, 1998. http://wwwvis.informatik.uni-stuttgart.de/~engel/Publications/vis98_ps%.gz.

[21] K. Engel, P. Hastreiter, B. Tomandl, K. Eberhardt, and T. Ertl. Combining local and remote visualization techniques for interactive volume rendering in medical applications. In *Procceedings of IEEE Visualization '00*, pages 449–452. IEEE, 2000.

[22] Klaus Engel, Rüdiger Westermann, and Thomas Ertl. Isosurface extraction techniques for web-based volume visualization. In *IEEE Visualization '99 Proceedings*, pages 139–146, 1999.

[23] H. Freeman and S. Morse. On searching a contour map for a given terrian elevation profile. *Journal of the Franklin Institute*, 284(1):1–25, 1967.

[24] G. W. Furness. Generalized fisheye views. In *Proc. SIGCHI'86*, pages 16–23, 1986.

[25] Leonidas J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, 4(2):74–123, April 1985.

[26] V. Interrante. Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution. In *Proc. SIGGRAPH 97*, pages 109–116, 1997.

[27] T. Itoh and K. Koyamada. Automatic isosurface propagation using an extrema graph and sorted boundary cell lists. *IEEE Trans. Visualizat. Comput. Graph.*, 1:319–327, 1995.

[28] Daniel B. Karron, James Cox, and Bhubaneswar Mishra. New findings from the spiderweb algorithm: Toward a digital morse theory. In Richard A. Robb, editor, *Visualization in Biomedical Computing - '94*, number 2359 in SPIE Proceedings, pages 643–657, 1994.

[29] T. Alan Keahey. *Nonlinear magnification*. PhD thesis, Indiana University, Computer Science, December 1997. `http://www.cs.indiana.edu/~tkeahey/research/papers/thesis.html`.

[30] P. Leinhardt. Subdivision of n-dimensional spaces and n-dimensional generalized maps. In *Proc. 5th Annu. ACM Sympos. Comput. Geom.*, pages 228–236, 1989.

[31] Y. Livnat, H. Shen, and C. Johnson. A near optimal isosurface extraction algorithm for structured and unstructured grids. *IEEE Trans on Vis. and Comp. Graphics*, 2(1):73–84, 1996.

[32] Y. Livnat, H.-W. Shen, and C. R. Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Trans. Visualizat. Comput. Graph.*, 2:73–84, 1996.

[33] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Comput. Graph.*, 21(4):163–170, 1987.

[34] Kwan-Liu Ma. Image graphs – a novel approach to visual data exploration. In *IEEE Visualization '99 Proceedings*, pages 81–88, 1999.

[35] Kwan-Liu Ma and David Camp. High performance visualization of time-varying volume data over a wide-area network. In *Supercomputing '00*, November 2000.

[36] S. Menon and M. Rizk. Large-eddy simulations of three-dimensional impinging jets. *Intl. J. Comp. Fluid Dynam.*, 7(3):275–290, 1996.

[37] C. Michaels and M. Bailey. Vizwiz: a Java applet for interactive 3D scientific visualization on the web. In *Proc. IEEE Visualization '97*, pages 261–267, 1997. `http://www.sdsc.edu/vizwiz/`.

[38] John W. Milnor. *Morse Theory*. Princeton University Press, Princeton, NJ, 1963.

[39] G. Nielson and B. Hamann. The asymptotic decider: Resolving the ambiguity in marching cubes. In *IEEE Visualization '91 Proceedings*, pages 83–91, October 1991.

[40] Atsuyuki Okabe, Barry Boots, and Kokichi Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, Chichester, UK, 1992.

[41] A. Paoluzzi, F. Bernardini, C. Cattani, and V. Ferrucci. Dimension-independent modeling with simplical complexes. *ACM Trans. Graph.*, 12(1):56–102, January 1993.

[42] J. Rossignac. Considerations on the interactive rendering of four-dimensional volumes. In *Proc. of the Chapel Hill Workshop on Volume Visualization*, pages 67–76, 1989.

[43] J. Rossignac, A. Safanova, and A. Szymczak. 3d compression made simple: Edgebreaker on a corner table. In *Proc. Shape Modeling International Conference*, pages 278–283, Genoa, Italy, May 2001.

[44] H.-W. Shen. Isosurface extraction in time-varying fields using a temporal hierarchical index tree. In *Proceedings of IEEE Visualization*, pages 159–166, October 1998.

[45] Han-Wei Shen, Ling-Jan Chiang, and Kwan-Liu Ma. A fast volume rendering algorithm for time-varying fields using a time-space partitioning (TSP) tree. In *Proc. IEEE Visualization '99 Conference*, pages 371–378, 1999.

[46] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosien. Surface coding based on morse theory. *IEEE Comput. Graph. Appl.*, 11:66–78, September 1991.

[47] P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. In *San Diego Workshop on Volume Visualization*, volume 24 of *Comput. Gr*, pages 63–70, December 1990.

[48] Maureen C. Stone, Ken Fishkin, and Eric A. Bier. The movable filter as a user interface tool. In *Proc. CHI'94*, pages 306–312, 1994.

[49] Philip Sutton and Charles Hansen. Isosurface extraction in time-varying fields using a temporal branch-on-need tree (T-BON). In *Proc. IEEE Visualization '99 Conference*, pages 147–153, 1999.

[50] S. Tarasov and M. Vyalyi. Construction of contour trees in 3d in $O(n \log n)$ steps. In *14th Annual ACM Symposium on Computational Geometry*, Minneapolis, MN, June 7-10 1998.

[51] J. Trapp and H-G. Pagendarm. A prototype for a WWW-based visualization service. In *Proc. Eurographics '97*, pages 23–30, 1997.

[52] E. R. Tufte. *The Visual Display of Quantative Information*. Graphics Press, Cheshire, Connecticut, 1983.

[53] M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 212–220, 1997.

[54] J. Wilhelms and A. Van Gelder. Octrees for faster isosurface generation. *ACM Trans. Graph.*, 11:201–227, 1992.

[55] C. Wittenbrink, K. Kim, J. Story, A. Pang, K. Hollerbach, and N. Max. Permweb: remote parallel and distributed volume visualization. In *SPIE Proceedings on Visual Data Exploration and Analysis IV*, pages 100–110, San Jose, CA, Feb. 1997.

oil   water   density momentum energy   kinetic   flame

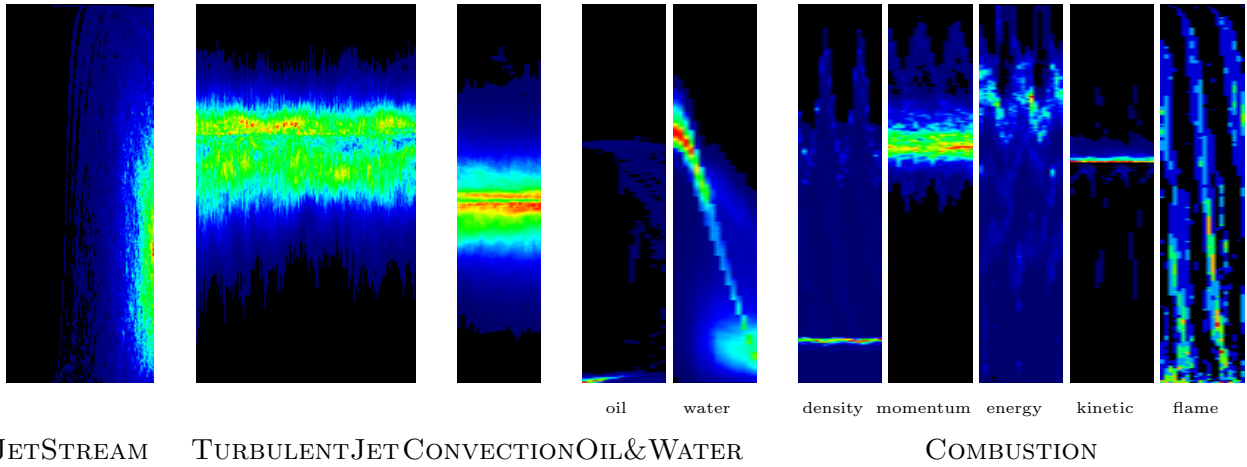JETSTREAM   TURBULENTJET CONVECTIONOIL&WATER   COMBUSTION

Fig. 8. Images in the control plane illustrating the number of connected components for our data sets
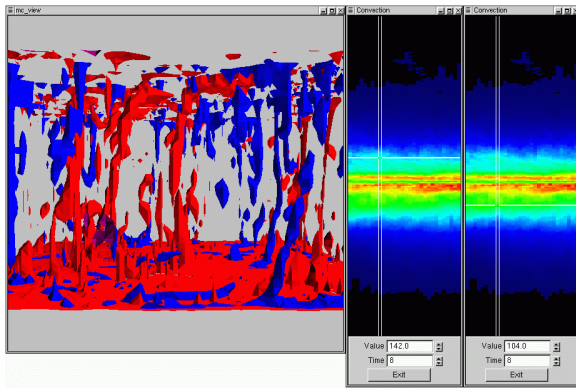

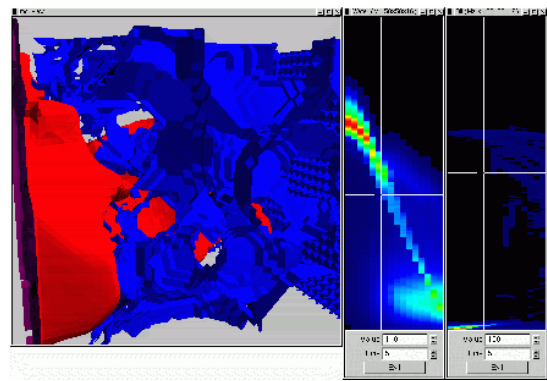Fig. 9. CONVECTION: Two iso-surfaces for same data set show hot magma rising and cold falling

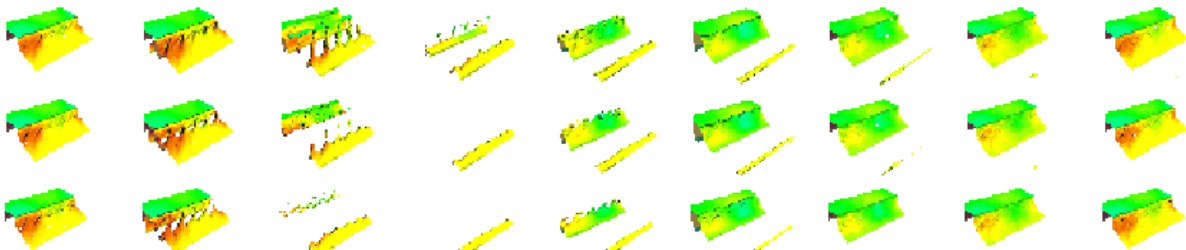Fig. 10. OIL&WATER: iso-surfaces and control planes for water and oil densities

Fig. 11. Part of the mosaic image for the density iso-surface preview of the COMBUSTION data set