Volume Tracking Using Higher Dimensional Isosurfacing

Guangfeng Ji* Han-Wei Shen† Rephael Wenger‡
Department of Computer and Information Science
The Ohio State University

Abstract

Tracking and visualizing local features from a time-varying volumetric data allows the user to focus on selected regions of interest, both in space and time, which can lead to a better understanding of the underlying dynamics. In this paper, we present an efficient algorithm to track time-varying isosurfaces and interval volumes using isosurfacing in higher dimensions. Instead of extracting the data features such as isosurfaces or interval volumes separately from multiple time steps and computing the spatial correspondence between those features, our algorithm extracts the correspondence directly from the higher dimensional geometry and thus can more efficiently follow the user selected local features in time. In addition, by analyzing the resulting higher dimensional geometry, it becomes easier to detect important topological events and the corresponding critical time steps for the selected features. With our algorithm, the user can interact with the underlying time-varying data more easily. The computation cost for performing time-varying volume tracking is also minimized.

CR Categories: I.3.6 [Computing Methodologies]: COMPUTER GRAPHICS—Methodology and Techniques

Keywords: tracking, isosurface, interval volume, higher dimensional isosurfacing

1 Introduction

One major factor that contributes to the rapid growth of data size in the recent years is the increasingly widespread ability to perform very large scale time-varying simulations. To understand the complex dynamic phenomena from a time-varying data set, the visualization tool must compute, animate, and track salient features at an interactive speed. For scalar fields, isosurfaces and interval volumes are commonly used for characterizing features that can be described by the underlying data values. An isosurface represents points of a constant value, which can be used to reveal the geometric structure of objects represented by the data. In practice, the measurement devices or numerical simulations may involve certain degree of errors in the sampled or simulated data sets. Displaying interval volumes can tolerate the errors and provides more meaningful visualization results.

*e-mail:jig@cis.ohio-state.edu
†e-mail:hwshen@cis.ohio-state.edu
‡e-mail:wenger@cis.ohio-state.edu

IEEE Visualization 2003,
October 19-24, 2003, Seattle, Washington, USA
0-7803-8120-3/03/$17.00 ©2003 IEEE

When analyzing time-varying data sets, a straightforward animation of isosurfaces or interval volumes may not be always effective, since each image can contain a large number of small components, each of which may experience complex evolutions over time. Computing the complete isosurfaces or interval volumes in a long time sequence is not only time-consuming, but also confusing when visualizing all the evolving components simultaneously. To study the temporal characteristics of time-varying data, having the ability to isolate local features of interest and track their evolution in time can be more visually effective and computationally efficient. In this paper, we present an algorithm to track the evolution of isosurfaces and interval volumes from time-varying data using higher dimensional isosurfacing. Our algorithm can efficiently track the user-selected local features, defined as connected isosurface or interval volume components, by performing local propagation and interactive slicing of isosurfaces or interval volumes in R^4. By using higher dimensional isosurfacing, we can easily establish the feature correspondence in adjacent time steps. The resulting higher dimensional geometry also allows us to detect critical points, which can be used to identify the topological changes such as amalgamation, bifurcation, creation and dissipation experienced by the time-varying features. The critical time steps indicating when those events occur can also be found.

The organization of the paper is as follows. We first review the related work, and then describe the algorithm to track time-varying isosurfaces and interval volumes in section 3 and 4 respectively. In both sections, we will present our algorithm to extract overlapping components, the approach to identify critical time steps and the corresponding evolutionary events, and how we verify the correspondence between features. Test results are presented in section 5 and conclusion and future work are presented in section 6.

2 Related Work

Researchers have proposed various techniques to track time-varying features. Banks and Singer [Bank and Singer 1995] used a predictor-corrector method to reconstruct and track vortex tubes from turbulent time-dependent flows. Samtaney *et al.*[Samtaney et al. 1994] tracked 3D features using object centroids and second order moments. Reinders *et al.*[Reinders et al. 2001] calculated a set of attributes, such as center point position, volume, mass, best fitting ellipsoid for all the features in all the frames and used these data to track features through a predication/verification scheme. There is also a rich literature in computer vision on motion tracking [Ballard 1982; Aggarwal and Nandhakumar 1988; Shi and Tomasi 1994; Carlbom et al. 1992]. The main difference between tracking 2D objects from videos and tracking features from simulation data is that features or regions of interest in scientific visualization applications are often manifested as 3D objects which tend to evolve and interact, while those 2D objects in computer vision interact less frequently.

Previous work most related to this paper is the volume tracking algorithms proposed by Silver and Wang [Silver 1995; Silver and Wang 1996; Silver and Wang 1997; Silver and Wang 1998]. Their algorithms are based on the observations that when the underlying temporal resolution is high enough, corresponding features in adja-

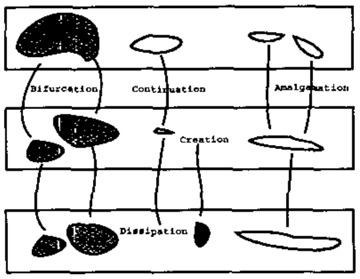

Figure 1: The evolutionary events that a feature may experience.

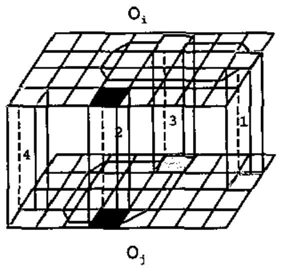

Figure 2: The four different types of 3-cells when two isocontours in R^2 intersect.

cent time steps usually overlap. Given that, features manifested as interval volumes are first identified and stored in appropriate data structures. Then, correspondences between features in consecutive time steps are identified using a two-stage process including an overlap and a best matching test. In the overlap test, spatially overlapped features from consecutive time steps are identified and the number of intersecting nodes is also computed. Octree and linked list data structures are used to accelerate this process. The best matching test involves inspecting the ratio of difference versus maximum volumes among all combinations of overlapped features, and identifying how the feature changes its topological structures. Based on how the topological structure of a local feature evolves over time, one of the following events can occur: (see Figure 1)

- Continuation: an object continues to the next time step, with possible shape deformation and change of position, orientation, etc.
- Creation: a new object starts to appear.
- Dissipation: an object disappears.
- Bifurcation: an object splits into several objects.
- Amalgamation: several objects merge into a single one.

Inspired by Silver and Wang's work [Silver 1995; Silver and Wang 1996; Silver and Wang 1997; Silver and Wang 1998], in this paper we present a volume tracking algorithm based on higher dimensional isosurfacing. Our goal is to allow interactive tracking of *local* features. A local feature is defined in this paper as a connected component that belongs to an isosurface or interval volume. When a scientist is presented with a rendering of an isosurface or an interval volume at a particular time step, he or she can select a connected component of interest, and then follows its evolution over time. Instead of computing the whole isosurfaces or interval volumes in the subsequent time steps, our algorithm generates on the fly only the overlapping components to reduce the computation cost. In the following, we describe our algorithm in detail.

3 Isosurface Tracking

Our algorithm is based on higher dimensional isosurfacing. Computing isosurfaces or interval volumes in R^4 allows us to easily track overlapped local features. With the resulting geometry in R^4, we can also detect the topological events such as amalgamation, bifurcation, creation and dissipation. In the following, we first describe the algorithm to track time-varying isosurfaces.

3.1 Extracting Overlapping Time-varying Isosurfaces

To track isosurfaces in a time-varying field, instead of first computing a complete set of isosurfaces in the time sequence and then identify the component that overlaps with the user selected local feature, better efficiency can be achieved if we only compute the overlapped components from adjacent time steps. To do so, we will extract isosurfaces in R^4. Previously, researchers have proposed algorithms to extract isosurfaces in R^4 from time-varying data [Bhaniramka et al. 2000; Weigle and Banks 1998; Weigle and Banks 1996]. Smoother surface animations can be produced since the isosurface in R^4 can be sliced at higher temporal resolutions. In our algorithm, we use isosurfacing in R^4 to detect the overlapping isosurfaces based on the observation that if two isosurface components from the consecutive time steps overlap with each other in the spatial domain, they will belong to the same connected isosurface component in R^4. This can be shown as follows.

Suppose an isosurface component O_i in R^3 at time t overlaps with another isosurface component O_j in R^3 at time $t+1$. Each isosurface component goes through a list of 3-cells(cells in R^3). We can build 4-cells(cells in R^4) out of the 3-cells at time t and $t+1$. Each 4-cell is made up of two 3-cells that have the same spatial location but one from t and the other from $t+1$. There are two cases on how these two isosurface components in R^3 overlap.

- 1. O_i and O_j have surface intersection.
- 2. O_i and O_j have no surface intersection, but one is completely inside of the other.

For the first case, each 4-cell in the time-varying field can be classified into one of the following four categories: (For 2D case, see Figure 2)

- 1. The cell passes through O_i at t but not O_j at $t+1$.
- 2. The cell passes through O_j at $t+1$ but not O_i at t.
- 3. The cell passes through O_i at t and O_j at $t+1$.
- 4. The cell passes through neither O_i at t nor O_j at $t+1$.

Note that the different categories are mutually exclusive and all four categories comprise the entire 4-cells in the time-varying field. Type 1, 2 and 3 4-cells are isosurface cells in R^4, since they contains isosurface in R^3, while type 4 4-cells are not isosurface cells in R^4. Any type 3 4-cell can propagate to any type 1, 2 and 3 4-cell; any type 1 4-cell can propagate to any type 1 and 3 4-cell; and any type 2 4-cell can propagate to any type 2 and 3 4-cell. Type 1 and type

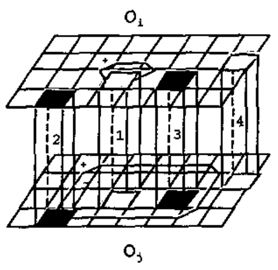

Figure 3: The four different types of 3-cells when one isosurface is contained within the other.

2 4-cells can propagate to each other only if there exists a type 3 4-cell. Therefore, if O_i at t has surface intersection with O_j at $t+1$ in space, that is, there exists a type 3 4-cell, then there will be a connected component in R^4 that passes through O_i at t and O_j at $t+1$. In other words, O_i and O_j belong to the same connected component in R^4.

In the second case, O_i has no surface intersection with O_j, but the volumes inside the surfaces overlap. Without loss of generality, we assume O_i lies within O_j. The 4-cells connecting them can be classified into the following four categories: (For 2D case, see Figure 3)

- 1. The cell passes through O_i at t but not O_j at $t+1$.
- 2. The cell passes through O_j at $t+1$ but not O_i at t.
- 3. The two corresponding 3-cells lie outside of O_i but inside of O_j.
- 4. The two corresponding 3-cells lie either inside O_i or outside O_j.

Similar to the above analysis, these four categories of 4-cells are mutually exclusive and all 4 categories comprise the entire 4-cells in the field. It can be seen that type 1 and 2 4-cells are isosurface cells in R^4, since they contain isosurfaces in R^3. For each type 3 4-cell, the signs of its corners at t are different to those at t+1 and thus the isosurface in R^4 will cut through the 4-cell, which implies type 3 4-cells are also isosurface cells in R^4. Type 4 4-cells have the same sign for every corner and thus are not isosurface cells in R^4. Type 3 4-cells can propagate to type 1 and 2 4-cells while type 1 and 2 4-cells can propagate to each other by going through some type 3 4-cells. An isosurface component in R^4 that passes through O_i at t and O_j at $t+1$ lies exactly within all 4-cells of type 1, 2 and 3.

It is worth mentioning that here we assume that the scalar values inside the isosurfaces in R^3 at the consecutive time steps are consistent relative to the isovalue, i.e., all of them are either smaller or greater than the isovalue. It may happen that the scalar fields get reversed, that is, at one time step, the scalar values inside the isosurface are smaller than the isovalue but at the other time step they are greater than the isovalue. This happens when the scalar field changes too quickly, or when two concentric isosurface components are tracked. In both cases, the isosurface in R^4 will not pass through the region between these two isosurfaces in R^3, and it will have two disjoint components. Specifically, when the scalar field changes too quickly, the isosurface component at time t will disappear and then another isosurface component at time $t+1$ starts to appear. In the case that two concentric isosurface components are tracked, the fact that these two components in R^3 do not belong to the same isosurface component in R^4 guarantees that the inner component at time t will be tracked to the inner component at time $t+1$ and the outer component at time t to the outer one at time $t+1$.

Based on our analysis, given a user selected isosurface component O_i in R^3 at time t, we can rapidly compute the overlapping isosurface component(s) at time $t+1$ by first extracting the connected isosurface component in R^4 that intersects with O_i, and then slicing it to get the isosurface component(s) at time $t+1$. The isosurface in R^4 can be computed by propagation from any 4-cell that intersects with O_i. When the underlying mesh is a Cartesian grid, the 4-cell is a hypercube. If the mesh is a tetrahedral grid, then the cell is a 4-simplex. Previously, researchers have used cell propagations to compute isosurfaces in R^3 [Bajaj et al. 1996], we can perform cell propagation in R^4 in a similar manner. Note when propagating 4-cells, for a large time varying data set, it may be impractical to get the full isosurface component in R^4, since it is very likely that the component will span many time steps and it will incur serious memory overhead to generate it. In our implementation, we only propagate to generate the necessary part of isosurface components which lie between time t and time $t+1$.

During the propagation, the 4-cells that contain the isosurface need to be triangulated. Previously, Weigle and Banks [Weigle and Banks 1996] proposed a recursive contour meshing algorithm, which extracts isosurfaces in R^4 by first breaking the 4-cell, or hypercube into 4-simplicies, and then looping through the faces of the simplices to construct the isosurfaces. Bhaniramka *et al.*[Bhaniramka et al. 2000] devised an algorithm that can generate triangulation tables for arbitrary N dimensional cells. Both algorithms can be used to compute isosurfaces in R^4. Since the algorithm by Bhaniramka *et al.*produces a smaller number of tetrahedra, we choose their method in our algorithm and create the triangulation table for 4-cells at a preprocessing stage. Once the table is generated, isosurfaces in R^4 can be generated at run time by a method similar to the Marching Cubes algorithm [Lorensen and Cline 1987]. The resulting isosurface is a collection of tetrahedra embedded in R^4. To extract the overlapping isosurface components in the spatial domain, we can just slice the connected isosurface component in R^4 at the desired time step.

3.2 Detecting Topological Event and Critical Time Step

By resorting to isosurfaces in R^4, the correspondence between isosurface components in R^3 in consecutive time steps can be established. In addition to tracking the evolution of features visually by slicing the component in R^4 along the time axis and displaying the resulting objects, it is possible to detect the critical time steps automatically when the features of interest undergo a change in their topological configuration. This can be done by analyzing the generated geometry in R^4. In the following, we describe the detection of the critical time steps as well as the identification of the topological events for time-varying isosurfaces.

An isosurface component in R^4 is a tetrahedral mesh embedded in 4-space. Each vertex of the tetrahedral mesh has coordinates (x,y,z,t). To compute the isosurface component in R^3 at a particular time step τ, we can slice the tetrahedra in R^4 based on the t coordinates at the tetrahedra vertices. This is equivalent to treating the mesh in R^4 as a normal mesh in R^3, with the time values as the scalar values defined over the tetrahedron vertices. Hereafter we call this *T-mesh*. The isosurface of $t = \tau$ from the T-mesh then corresponds to the isosurface component in R^3 at time τ, which can be easily extracted using the marching tetrahedra algorithm.

Based on the above idea, tracking an isosurface component in time is equivalent to computing the isosurfaces using different threshold values τ from the T-mesh. The critical points of this mesh, i.e., local minima, local maxima, and saddle points with time values as the scalar values, indicate when and where the topology of the isosurface will change if the isosurface is animated across those time steps. Previously, critical points have been studied in many applications [Kreveld et al. 1997; Tarasov and Vyalyi 1998; Carr et al. 2003; Pascucci and Cole-McLaughlin 2002; Gerstner and Pajarola 2000]. When analyzing the topology of T-mesh, a local minima corresponds to the *creation* of an isosurface component in R^3 while a local maxima corresponds to the *dissipation* of a component in R^3. Saddle points may correspond to either the *amalgamation* or the *bifurcation* of components in R^3. *Continuation* occurs at regular vertices of the mesh. The time values associated with the critical points indicate when those topological events happen. Here we call them *critical time steps*.

To detect the critical time steps and learn how the isosurface component changes its topology, we can find the critical points from the T-mesh. Since the tetrahedral mesh in R^4 contains only linear elements, all critical points are located at the vertices of the mesh. For each vertex v_i, an efficient way to identify whether it is a critical point is as follows [Gerstner and Pajarola 2000]:

- Find the set S which consists of all the adjacent vertices of v_i.
- Mark each vertex in S. If the scalar value at the vertex is greater than the value of v_i, mark it as positive. Otherwise mark it as negative.
- For each edge connecting vertices of S, if two end vertices have the same sign, keep the edge. Otherwise delete the edge.
- Count the number of the remaining connected components of the adjacent vertices.
- If the component number is 1, then v_i is a local minima or maxima. If the component number is greater than 2, v_i is a saddle point that corresponds to topological events.

Saddle points with component number greater than 2 occur at vertices where topological bifurcation or amalgamation happens. If only one component out of all components has time values less than the time at v_i, the saddle point corresponds to bifurcation. And if only one component has time values greater than the time at v_i, then the saddle point corresponds to amalgamation.

3.3 Verification

Correspondences between isosurface components from adjacent time steps can be easily found by using isosurfacing in higher dimensions. The idea behind is that if components overlap with each other, they must belong to the same isosurface component in higher dimension. One issue needs to be addressed is that this higher dimensional isosurfacing approach does not discriminate the degree of overlap between components. However, sometimes users might want to set a threshold so that only components that have a significant overlap are concluded as having correspondence. In this section, we describe how we take this user-defined overlap threshold into account.

Two isosurface components in consecutive time steps correspond to each other if they can be generated by slicing the same isosurface component in R^4. However, in some cases when the overlap between the two components is too small, the user may not want to consider them as corresponding objects. That is, component O_i and O_j correspond to each other if and only if:

- 1. They are generated by slicing the same isosurface component in R^4.
- 2. $V(\text{overlap}(O_i, O_j)) / \text{Min}(V(O_i), V(O_j)) > \mathit{Threshold}$

where *Threshold* is a user-supplied parameter, V() returns the volume of the component, and V(overlap(O_i, O_j)) gives the volume of the overlap between two components.

We measure the volume of an isosurface component in R^3 based on the number of 3D cells lying inside the surface. The inside/outside decision can be precomputed based on the sign of the voxel compared to isovalue, except when embedded isosurface exists, in which case the decision should be reversed. The issue of which sign corresponds to which side can be solved based on the following heuristic:

- The volume of two random picked isosurface components in R^3 are very likely to be different, but the volume outside of them will be the same.

For any cell inside the isosurface, its eight vertices should have an identical sign, i.e., they are either all greater, or all smaller than the isovalue. Furthermore, the signs are consistent for all the inside cells. Therefore, the volume of an isosurface component can be computed by first finding an inside cell, and then using the cell as a seed to flood the entire volume inside the isosurface. Cells on the boundary of surface also contribute to the volume and should be processed properly. A good approximation of their contribution to the whole volume would be the ratio of corners that lie inside the surface to the total eight corners.

The volume of the overlapped region for two isosurface components in R^3 at two consecutive time steps is the number of 3D cells that lie in both components. This can be easily calculated by counting the number of cells that belong to both components. This volume is used to test whether the degree of overlap between the isosurface components satisfies the user's threshold. If the threshold is satisfied, the correspondence holds; otherwise there will be no correspondence.

4 Interval Volume Tracking

Features are often manifested as points within a range of data values, which can be represented as an interval volume [Guo 1995]. An interval volume is a generalization of isosurface and represents a three-dimensional subvolume for which the associated scalar values lie within a user-specified closed interval. When there are errors caused by measurements or numerical simulations, or the target object contains structural ambiguity, displaying interval volumes can provide more meaningful visualization results [Fujishiro et al. 1996].

To track interval volumes in a time-varying data set, we can generalize the idea of isosurface tracking described above. That is, given a user selected interval volume component V_i at time t, we can extract an interval volume in R^4 that intersects with V_i, and then slice the interval volume in R^4 along the time dimension to get the interval volume component(s) at time $t+1$ that overlap with V_i. We can also detect topological events and critical time steps by analyzing the criticality of the resulting geometry and verify whether the overlap is significant.

4.1 Extracting Overlapping Time-Varying Interval Volumes

To extract interval volumes that overlap with a user selected component V_i at time t, we will first extract an interval volume in R^4 that intersects with V_i, which will be then sliced along the time dimension to get the interval volume component(s) at time $t+1$ that overlap with V_i. Previously, researchers have proposed algorithms to extract interval volumes in R^3. Fujishiro *et al.*[Fujishiro et al.

1996] extended the Marching Cubes algorithm and used a solid fitting algorithm to tetrahedralize the interval volume. The algorithm proposed by Nielson *et al.*[Nielson and Sung 1997] performed the tetrahedralization by decomposing each volume cell to five tetrahedra and then using an efficient lookup table to compute the interval volume within each tetrahedron. Bhaniramka *et al.*[Bhaniramka et al. 2000] extracted the interval volume by creating two scalar fields from the original data and then constructed an isosurface in R^4. The projection of the isosurface in R^4 to the spatial domain is the interval volume. All the above algorithms are for computing interval volumes in 3-space. In the following, we present an algorithm to compute interval volumes in 4-space.

4.1.1 Computing Interval Volumes in R^4

Our method is based on five dimensional isosurfacing. Given a four dimensional scalar field f(x,y,z,t), the interval volume $I[a,b]$ consists of all the points that satisfy $a \leq f(x,y,z,t) \leq b$. To compute the interval volume, we first artificially create a five dimensional scalar field $g(x,y,z,t,w)$ with the w dimension equal to two. We let $g(x,y,z,t,0) = f(x,y,z,t) - a$ and $g(x,y,z,t,1) = f(x,y,z,t) - b$. We also assume the five dimensional scalar field varies linearly along the w dimension, that is,

$$g(x,y,z,t,w) = (1-w)g(x,y,z,t,0) + wg(x,y,z,t,1)$$

where w is between 0 and 1. The interval volume $a \leq f(x,y,z,t) \leq b$ can be extracted by first computing the zero isosurface from the 5D scalar field g, and then projecting the resulting isosurface along the w axis to 4-space. The correctness of this algorithm is shown as follows.

The zero isosurface of the five dimensional field g contains the 5D points (x,y,z,t,w) that satisfy $g(x,y,z,t,w) = 0$. Hereafter we call those points zero isosurface points. For the zero isosurface points that have w=0, if we project them orthographically along the w direction, i.e., project from (x,y,z,t,0) to (x,y,z,t), we know that the projected point (x,y,z,t) will satisfy $f(x,y,z,t) = a$ since $g(x,y,z,t,0) = 0 = f(x,y,z,t) - a$. Similarly, for the zero isosurface points that have w = 1, i.e., (x,y,z,t,1), we have $g(x,y,z,t,1) = 0$. Since $g(x,y,z,t,1) = f(x,y,z,t) - b$, if we project those points along the w axis, we know the projected point (x,y,z,t) will satisfy $f(x,y,z,t) = b$. Finally, for those zero isosurface points that have $w \in (0,1)$, since $g(x,y,z,t,w) = (1-w)g(x,y,z,t,0) + wg(x,y,z,t,1)$, if we project those points to 4-space, it can be seen that $a < f(x,y,z,t) < b$ will be satisfied. Therefore, if we project the zero isosurface points (x,y,z,t,w) to four dimensions along the w axis, we will get all the points (x,y,z,t) that satisfy $a \leq f(x,y,z,t) \leq b$, i.e., the interval volume $I[a,b]$ in R^4.

Given a time-varying field, remember that the goal of computing the interval volume in R^4 is to enable the tracking of an interval volume in R^3 from time step t to time step $t+1$. Therefore, the construction of the five dimensional scalar field g(x,y,z,t,w) need not be performed globally. Only the volume cells in R^5 that will be encountered during propagation need to be constructed. Hence, the space and computation overhead is manageable. To compute the zero isosurface in R^5, we can use the triangulation table generated with the algorithm by Bhaniramka *et al.*[Bhaniramka et al. 2000]. Constructing isosurfaces in R^5 requires a five dimensional triangulation table. In the next section, we discuss some practical issues related to the generation of the table.

4.1.2 Triangulation Table for isosurfacing in R^5

Similar to the Marching Cubes lookup table, the 5D triangulation table provides the edge intersection and connectivity information to guide the triangulation process. Each entry in the table contains the geometry configuration consisting of four-simplices to represent the isosurface in the 5D hypercube. A 5D hypercube is composed of two 4D hypercubes defined in xyzt spatio-temporal dimensions, which are connected by 16 edges lying in the w direction. We used the algorithm proposed by Bhaniramka *et al.*[Bhaniramka et al. 2000] to generate the table. We also augment the table to include the propagation directions based on the case's geometry configuration. Since a 5D hypercube contains 32 vertices, the size of the table can be quite large. Therefore, care should be taken when using the table at run time.

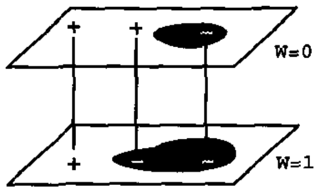

Figure 4: Only 3 sign combinations are possible when extracting interval volumes in R^2.

An N dimensional hypercube contains 2^N vertices. Without performing case reduction as done in the Marching Cubes algorithm, the number of cases for triangulation table will be 2^{2^N}. For instance, in the 3D triangulation table there are $2^8 = 256$ cases. For a 5-cell, the number of vertices will be 32. Therefore, a 5D isosurface lookup table will contain 2^{32}=4G entries. The 256 cases in the 3D marching cubes can be reduced to 15 after applying complementary and rotational symmetry. However, in four dimensions, further efforts must be taken to ensure a consistent triangulation on the surfaces of adjacent 4-cells. This problem will become even more complex when the dimension grows to five, which makes it very difficult to apply complementary and rotational symmetry.

When the 5D triangulation table is used to compute interval volumes in R^4, not all the four billion cases are possible. In the process of computing the interval volume $a \leq f(x,y,z,t) \leq b$, the scalar field at $w = 0$ is generated by subtracting the 4D field by a, and the scalar field at $w = 1$ is generated by subtracting the 4D field by b. Since $b > a$, for each of the 16 edges in the 5D hypercube lying in the w dimension, the scalar value at $w = 0$ is always greater than the scalar value at $w = 1$. This implies that the edges of the 5D hypercube lying in the w direction can not have a minus-plus sign combination, otherwise the scalar value at $w = 0$ will be smaller than the scalar value at $w = 1$. Hence, there are only 3 possible sign combinations, rather than 4, for each edge of the 5D hypercube in the w direction. For this reason, the 5D triangulation table used for interval volume generation can only contain $3^{16} \approx$ 43M, rather than 4G case. The case in R^2 is illustrated in Figure 4.

Although the 5D triangulation table is reduced from 4G cases to 43M, its size is still too large to be processed in core. However, given a data set, we find out that some cases occur more frequently, and some other cases never happen. The reduction from 4G cases to 43M and the uneven possibility of the occurrence of each case make it feasible and effective to store the 5D triangulation table into a hash table which is small enough to fit into main memory. With a properly designed hash function, the hash table will cache almost all 5D triangulation cases that can happen for a data set. In this way, the extraction of 4D interval volumes can be achieved in an efficient manner.

4.2 Detecting Topological Event and Critical Time Step

We can detect topological events and critical time steps for interval volume components by analyzing the resulting geometry. When extracting interval volumes in R^4, isosurfaces in R^5 are generated. Each vertex of the isosurface contains (x, y, z, t, w) coordinates. After projecting the isosurfaces to R^4, we get a mesh consisting of four-simplices that represents an interval volume embedded in R^4. The interval volume at a particular time step $t = \tau$ can be obtained by slicing the four-simplex mesh and using the time value of each vertex as the scalar value. This is similar to the method used to compute isosurfaces in R^3 from the T-mesh described in section 3.2, except that this T-mesh consists of 4-simplices rather than tetrahedra. To detect critical points in the interval volume, we can use the same method as the one in section 3.2 to extract the local minima, local maxima, and saddle points to identify the evolutionary events of the interval volumes in R^3.

4.3 Verification

A verification approach similar to that of isosurface component can be applied to tracking of interval volumes. The only difference would be the volume calculation. The volume enclosed by an isosurface is the whole region inside the isosurface; while the volume of an interval volume is the region lying between the top isosurface and the bottom one. An interval volume in R^3 can be generated by projecting an isosurface in R^4 along the w direction. Notice that this isosurface in R^4 goes through a 4-cell if and only if the interval volume in R^3 goes through the projection of the 4-cell along the w direction, which is a 3-cell. So the number of 3-cells passed by an interval volume in R^3 is equal to the number of 4-cells passed by the corresponding isosurface in R^4. Hence, the volume of the interval volume in R^3 can be easily calculated by counting the number of 4-cells on the isosurface in R^4. After all the necessary volumes are calculated, we can test whether the degree of overlap between the interval volume components satisfies the user's threshold. If it is satisfied, the correspondence holds; otherwise there will be no correspondence.

5 Results and Discussion

We have tested our tracking algorithm using a 128*128*128 vorticity magnitude data set with 100 time steps. All results were computed on a Pentinum IV 1.4GHZ PC with 768 Mbytes memory. In this section, we demonstrate the results of our algorithm for tracking both isosurfaces and interval volumes.

Figure 5 shows an example of tracking an isosurface that consists of multiple components. The time-varying isosurface is extracted with an isovalue 6.0. The resulting isosurfaces in R^4 are used to guide the coloring of the connected components, which allows us to distinguish different local features when following their evolutions over time. The color of each component is inherited from its parent in the previous time step. When a component is created, a new color is generated. When two or more components merge into a component, the current component would get the color of the previous dominant component. The dominance could be determined by properties such as volume, mass or local extremal value. In our result, we use volume as the criterion to determine the dominance, i.e., the current component would follow the color of the previous largest component when amalgamation happens.

Figure 6 shows the tracking of a local isosurface component. Snapshots from six time steps are shown. The isovalue of these components is 6.55. When one or a few components are selected, only the components from the subsequent time steps that overlap with the selected features are extracted using the four-dimensional

Time step(t)	9	10	17	18	24	27
Tracking time	0.07	0.08	0.071	0.06	0.04	0.01
4D isocontouring time	0.02	0.02	0.02	0.01	0.01	0.00
Num of tetrahedra	20873	21566	16980	15502	5195	99
Num of triangles at t	4340	4452	3936	3564	1256	32
Num of triangles at t+1	4452	4524	3564	3272	628	32

Table 1: The time to track the isosurface component illustrated in Figure 6(in seconds).

Critical time step	Critical point	Evolutionary Event
9.52427	Saddle	Bifurcation
17.0855	Local max	Dissipation
27.1854	Local max	Dissipation

Table 2: The critical time steps and evolutionary events of the example in Figure 6.

isosurfacing algorithm. In Figure 6, the component bifurcates into two at t=10. These two components continue to t=17. The smaller component disappears at t=18 while the larger one continues to t=27 and disappears at t=28. The tracking time and the 4D isosurfacing time are shown in Table 1. The tracking time includes extracting the connected isosurface components in R^4 across the adjacent time steps, calculating the volumes of the components and their overlaps, and slicing the isosurfaces in R^4 back to three dimensions. Several factors influence the performance of our tracking algorithm. Among them, the sizes of the generated geometries in both R^3 and R^4 from the adjacent time steps dominate. This shows that the complexity of our tracking algorithm is dependent on the feature size rather than the size of the data. The sizes of these geometries are also show in Table 1. Table 2 shows the critical time steps and the corresponding topological events for the isosurface components tracked in Figure 6.

Our algorithm can also track all interval volume components as well as individual interval volume component effectively. Figure 7 shows the tracking of two interval volume components. Snapshots from five time steps are shown. The interval volume was generated with isovalues between 5.3 and 7.0. We rendered the interval volume by drawing the faces of every tetrahedron. These two components first merge at t=29 and then split at t=31. All the timing and the sizes of geometries are shown in Table 3. Critical time steps and the evolutionary events are listed in Table 4.

Besides visualization, it is useful to know other properties such as the volume and mass of the features and how they change over time. This information can be used by scientists to gain further insight into the features. In our algorithm, the volumes of the components are computed in the verification process. Other properties can be computed similarly. As an example, the variations of the volume as a function of time for the case of isosurface component tracking in Figure 6 is illustrated in Figure 8.

The overlapping threshold plays an important role in identifying the correspondence between features from adjacent time steps. An appropriate value should depend on the drifting speed of the features as well as the data sampling rate. If the threshold is set too high, many features would be identified as creation/dissipation instead of continuation/amalgamation/bifurcation since no correspondence could be established. If the threshold is too low, features that overlap little would also be identified as continuation. In our

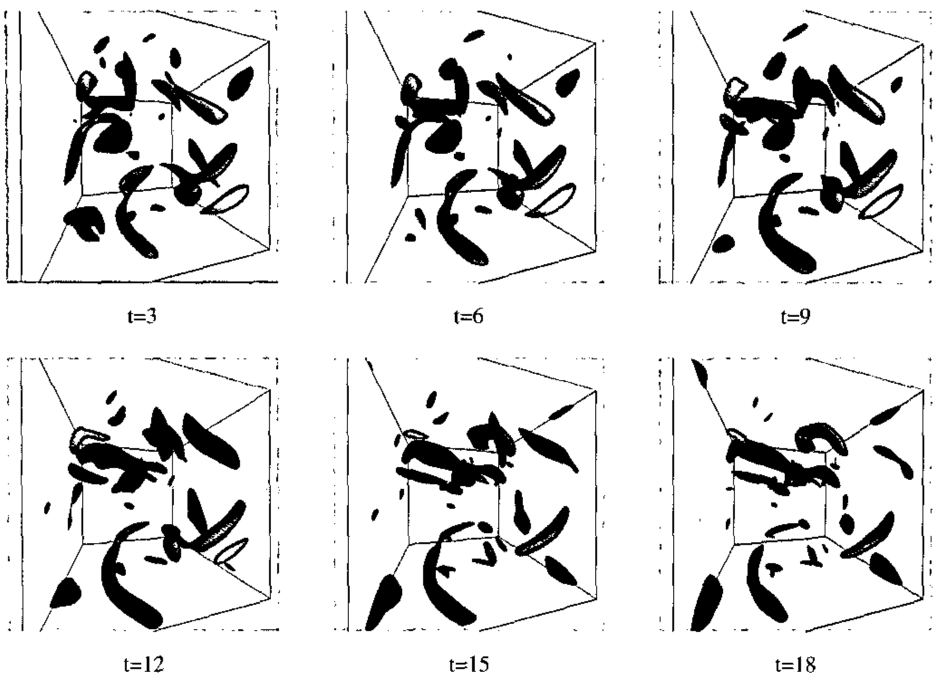

Figure 5: 6 out 100 time steps are shown. All isosurface components are tracked and the evolutionary information determines the coloring of the components.

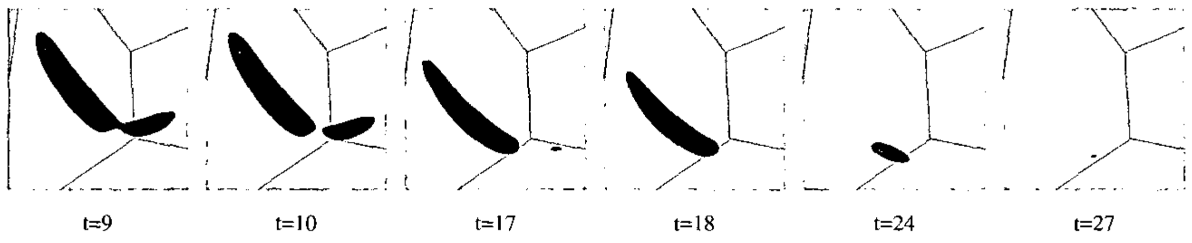

Figure 6: An isosurface component is tracked. Bifurcation happens at t=10 and dissipation at t=18 and 28.

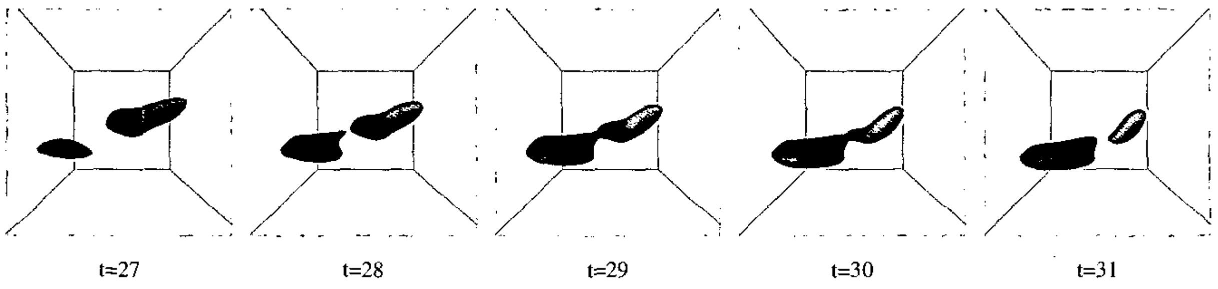

Figure 7: Two interval volume component are tracked. These two components merge at t=29 and then split at t=31.

test, we set the overlap threshold to 35% which leads to good event classification.

6 Conclusion and Future Work

In this paper we present a novel volume tracking technique based on higher dimensional isosurfacing. Local features defined as connected isosurface or interval volume components are tracked in an efficient manner. This is achieved by first extracting the connected

Time step(t)	27	28	29	30	31
Tracking time	0.20	0.26	0.271	0.29	0.29
5D isocontouring time	0.09	0.13	0.14	0.15	0.15
Num of 4-simplices	102160	142393	160689	174276	172887
Num of tetrahedra at t	25059	25070	29507	34132	35702
Num of tetrahedra at t+1	25070	29507	34132	35702	34577

Table 3: The time to track interval volume components illustrated in Figure 7(in seconds).

Time step	Critical point	Evolutionary Event
28.619	Saddle	Amalgamation
30.647	Saddle	Bifurcation

Table 4: The critical time steps and evolutionary events of the example in Figure 7.

components in higher dimension that intersects with the selected local components and slicing them to obtain the overlapping components at the next time step. The computing cost of performing higher dimensional isosurfacing is minimal, since an output sensitive propagation method is used. We can classify the type of evolutionary events experienced by the time-varying features and identify the critical time steps by analyzing the generated geometry in higher dimension.

All current work is based on the assumption that the sampling frequency is high enough so that features that have correspondence will overlap with each other in the spatial domain. However, in practice under-sampled data sets can exist. Our future research will investigate tracking techniques when the data is not well sampled.

7 Acknowledgement

This work is supported in part by NSF grant ACR 0118915, NASA grant NCC-1261, Ameritech Faculty Fellowship, and Ohio State Seed Grant. We wish to thank the anonymous reviewers for their valuable suggestions.

References

AGGARWAL, J., AND NANDHAKUMAR, N. 1988. On the computation of motion from sequences of images – a review. *Proceedings of the IEEE 76*, 8, 917–935.

BAJAJ, C., PASCUCCI, V., AND SCHIKORE, D. 1996. Fast isocontouring for improved interactivity. In *Proceedings of the 1996 IEEE Symposium on Volume Visualization*, 39–46.

BALLARD, D. 1982. *Computer Vision*. Prentice-Hall,Inc, Englewood, New Jersey.

BANK, D., AND SINGER, B. 1995. A predictor-corrector technique for visualizing unsteady flow. *IEEE Transactions on Visualization and Computer Graphics 1*, 2, 151–163.

BHANIRAMKA, P., WENGER, R., AND CRAWFIS, R. 2000. Isosurfacing in higher dimensions. In *Proceedings of Visualization 2000*, 267–274.

CARLBOM, I., CHAKRAVARTY, I., AND HSU, W. 1992. Integrating computer graphics, computer vision, and image processing in scientific applications. *Computer Graphics 26*, 1, 8–17.

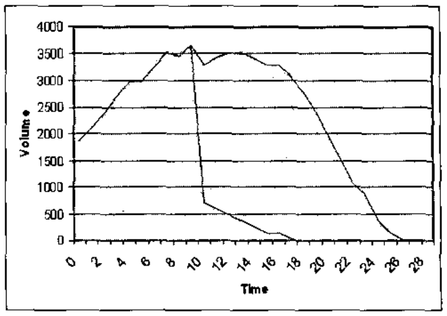

Figure 8: Volume vs time for the example in Figure 6. Note the component bifurcates into two and each of them dissipates.

CARR, H., SNOEYINK, J., AND AXEN, U. 2003. Computing contour trees in all dimensions. *Computational Geometry 24*, 2.

FUJISHIRO, I., MAEDA, Y., SATO, H., AND TAKESHIMA, Y. 1996. Volumetric data exploration using interval volume. *IEEE Transactions on Visualization and Computer Graphics 2*, 2, 144–155.

GERSTNER, T., AND PAJAROLA, R. 2000. Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In *Proceedings of Visualization 2000*, 259–266.

GUO, B. 1995. Interval set: A volume rendering technique generalizing isosurface extraction. In *Proceedings of Visualization 1995*, 3–10.

KREVELD, M., OOSTRUM, R., BAJAJ, C., PASCUCCI, V., AND SCHIKORE, D. 1997. Contour trees and small seed sets for isosurface traversal. In *Proceedings of 13th Annual ACM Symposium on Computational Geometry*, 212–220.

LORENSEN, W., AND CLINE, H. 1987. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics 21*, 4, 163–169.

NIELSON, G., AND SUNG, J. 1997. Interval volume tetrahedralization. In *Proceedings of Visualization 1997*, 221–228.

PASCUCCI, V., AND COLE-MCLAUGHLIN, K. 2002. Efficient computation of the topology of level sets. In *Proceedings of Visualization 2002*, 187–194.

REINDERS, F., POST, F., AND SPOELDER, H. 2001. Visualization of time-dependent data using feature tracking and event detection. *The Visual Computer 17*, 1, 55–71.

SAMTANEY, R., SILVER, D., ZABUSKY, N., AND CAO, J. 1994. Visualizing features and tracking their evolution. *IEEE Computer 27*, 7, 20–27.

SHI, J., AND TOMASI, C. 1994. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition 1994*, 593–600.

SILVER, D., AND WANG, X. 1996. Volume tracking. In *Proceedings of Visualization 1996*, 157–164.

SILVER, D., AND WANG, X. 1997. Tracking and visualizing turbulent 3d features. *IEEE Transactions on Visualization and Computer Graphics 3*, 2, 129–141.

SILVER, D., AND WANG, X. 1998. Tracking scalar features in unstructured datasets. In *Proceedings of Visualization 1998*, 79–86.

SILVER, D. 1995. Object-oriented visualization. *IEEE Computer Graphics and Applications 15*, 3.

TARASOV, S., AND VYALYI, M. 1998. Construction of contour trees in 3d in o(nlog n) steps. In *Proceedings 14th Annual ACM Symposium on Computational Geometry*, 68–75.

WEIGLE, C., AND BANKS, D. 1996. Complex-valued contour meshing. In *Proceedings of Visualization 1996*, 173–180.

WEIGLE, C., AND BANKS, D. 1998. Extracting iso-valued features in 4-dimensional scalar fields. In *Proceedings of the 1998 IEEE Symposium on Volume Visualization*, 103–110.