Progressive Tracking of Isosurfaces in Time-Varying Scalar Fields

Chandrajit Bajaj† Ariel Shamir* Bong-Soo Sohn†

† Department of Computer Sciences & TICAM
University of Texas at Austin

* The interdisciplinary center
Herzelia

Abstract

Scientific simulations and measurements often involve time dependent processes and produce time dependent data sets. Isosurface extraction is an important tool for visualizing three or two-dimensional time varying scalar fields defined by such data. Nevertheless, the size of the data and the dynamic nature impose difficulty in devising efficient and effective time dependent isosurface extraction techniques. In this paper, we describe a progressive algorithm for time dependent isosurface extraction. The algorithm maintains efficiency in time and space by exploiting coherency in both temporal and spatial dimensions of the data, as well as in the function values domain. It creates the isosurface of consecutive time steps progressively from previous time steps allowing time critical utilization. In addition, it can track evolving isosurface components and identify topology change events such as merge, split, vanish and create. This information is used to define several visualization techniques such as tracking of individual components, which help gain better understanding of the dynamic structure of the data.

1 Introduction

Isosurface extraction is an important visualization technique. By displaying the regions of constant value in scalar fields one can attain insight into the structure of the data. In many senses isosurface extraction can be viewed as a form of feature extraction from raw data, which helps discover significant characteristics inside the data. This is ever more true when time dependent data is concerned. Many scientific simulation and measurements are time dependent in nature and produce data sets of time-varying fields. Isosurface extraction and their tracking over time can greatly assist in understanding the dynamic behavior of the data. Nevertheless, often such time dependent data sets are very large and difficult to handle using conventional methods devised for static data. Moreover, any algorithm working on time-dependent data should be able to exploit the temporal nature of the data on the one hand, and to emphasize it on the other. Toward this end we present an efficient time-progressive isosurface extraction algorithm that enables isosurface components topology tracking.

We will assume that the input data is a two or three-dimensional mesh with scalar values on the nodes. The mesh has some predefined connectivity between the nodes, which imposes a spatial decomposition into cells. Some interpolation function is used to define a scalar field inside each cell, which is usually at least C^0 across cell boundaries. In such conditions, the process of isosurface extraction can be divided into two main stages:

Finding isosurface cells This stage involves the search for all cells in the mesh that contain the function iso-value, indicating that the isosurface must pass through them. We will call these cells *isosurface cells*.

Constructing the surface This stage involves the construction of an actual surface representation for the isosurface by extracting surface mesh elements from the isosurface cells (most commonly triangles).

In this paper we concentrate on the first stage in the extraction and rely on known techniques to build a triangular surface from the isosurface cells [9, 11, 15]. The classical algorithms for finding the isosurface cells involve marching through the whole mesh [9], hence maintaining a complexity of the order of the size of the mesh. More recent algorithms move the search to the function value domain, rearranging the data according to function space coherency [8, 14, 3, 5]. By building a suitable search structure these algorithms achieve complexity of the order of the size of the surface (the number of isosurface cells). Nevertheless, most of those techniques sacrifice spatial coherency (which is inherent in most data sets where C^0 continuity is assumed) by trading it with range space coherency. In addition, they rely on data-structures which sometimes are too heavy, particularly for time dependent data.

A hybrid approach is based on dividing the search for isosurface cells into *range-space* and *geometric* phases, taking advantage of coherence in both domains [2]. A preprocessing stage determines a subset S of all the cells which are maintained as candidate *seed cells*. For an arbitrary input iso-value, it is guaranteed that every connected component of the isosurface will intersect at least one cell in S (see Figure 1). A second preprocessing step constructs a range query structure for the cells in S. In the surface extraction phase, a surface propagation algorithm constructs the surface component (using e.g. breadth first search) from each selected cell in the seed set. Thus, the search for isosurface cells takes advantage of spatial coherence by using surface propagation, and range-space coherence through the utilization of a range-space search structure for seed cells.

Time dependent data presents an additional temporal dimension where high coherency is present. Techniques for time-dependent iso-surface extraction which use range space search either create very large structures, which become critical in out-of-core situations [13], or sacrificed temporal coherency in order to maintain disk access efficiency [18]. The key observation in temporal coherency utilization for isosurface extraction is that the isosurface at time t (I_t) is very close spatially to the isosurface at time $t+1$(I_{t+1}). This means that I_t can be used as an initial guess for I_{t+1}, and refined progressively using spatial marching methods to reach I_{t+1} (see Figure 3).

Isosurface extraction optimization is essential for efficient visualization. Nevertheless, other considerations are important as well for improved data analysis. Finding and visualizing the structural correlation between isosurfaces at successive times can improve the perception of the data dynamics. An isosurface often has many different connected components which gradually deform their shape over time. New components can be created, old components may move, merge, split, or vanish as time progresses. Identifying and tracking these changes may sometimes be crucial for understanding the data. Nevertheless, correlation between isosurface compo-

nents and topology tracking remains challenging when the data set is considered mainly as a collection of separate static time steps.

Our proposed algorithm begins by constructing the isosurface from a pre-generated seed set, identifying each individual component of the isosurface. Once the components are constructed, we trace their evolution over time by progressively marching the isosurface. To extract newly created isosurface components which cannot be traced from the isosurface of the previous time step, we use the pre-generated seed set again. This algorithm is fast enough for interactive visualization of time-dependent isosurfaces, can be used progressively when time limit constraints are imposed, and allows topology tracking over time with no demand for complex correspondence matching tests. The algorithm presents the following main contributions:

1. **Efficiency by means of exploiting coherency $\times$ 3:** Our algorithm extracts time dependent isosurface with minimal time complexity, and at low storage cost. This is achieved through the exploitation of all three coherencies: temporal, spatial and functional. Function value coherency is used by the search structure on the seed sets; temporal coherency is used by the time progressive update of isosurface components; spatial coherency is used by surface marching and propagation. All these guarantee near optimal time complexity (an order of the surface size). In terms of storage cost, the overhead is low since the search structures are used only for seed sets of all time steps.

2. **Progressive temporal extraction solution:** Using marching of isosurface components provides the ability to use temporal approximations of the real isosurface. The isosurface of time I_{t+1} is not extracted from scratch, but rather refined from the isosurface of I_t using two stages: temporal propagation of old components, and seed set generation of new components. These two stages rely on simple atomic operations that may easily be interrupted. In time critical situations, partially refined surfaces may be rendered as approximations. Moreover, skipping of time steps may also be employed easily, since holding the seed set for each time step allows full synchronization at any point in time (nevertheless, such synchronization can loose the topology tracking information).

3. **Effective topology tracking:** Topology changes such as creation, disappearing, merging and splitting of surface components may easily be tracked for dynamic structure analysis and component interaction. Moreover, when the isosurface has many evolving components including possibly some noise, the user can find and follow only a subset of important isosurface components. This can reduce distraction; provide informative statistics on the interaction among the components, and increase efficiency.

The rest of this paper is organized as follows. In Section 2, we present related work on isosurface extraction and volume feature tracking. In Section 3, we give definitions and explain some concepts used later in the text. Section 4 gives an overview of the algorithm for time dependent isosurface extraction, and the temporal propagation part is detailed in Section 5. Implementation details are given in Section 6, and some results in Section 7. We conclude and discuss future extensions in Section 8.

2 Related Work

Isosurface Extraction in Static Data Sets

A large amount of research has been devoted in the past for fast isosurface extraction from 3D static volume data. Marching Cubes algorithm [9] searches for isosurface cells by visiting all cells in the volume. An improvement is gained by using octrees [21] to skip cells not contributing to isosurface. Near optimal and optimal algorithms [8, 14, 3, 5] mostly use search structures on the function value domain. Contour propagation is used for efficient isosurface extraction [3, 7, 6]. Given an initial cell that contains the isosurface, the whole surface can be traced by contour propagation. This property is utilized (e.g. in the seed sets) to significantly reduce the space and time for searching isosurface cells. In addition, it allows identifying connected component of the isosurface by regarding each connected surface as a separate object.

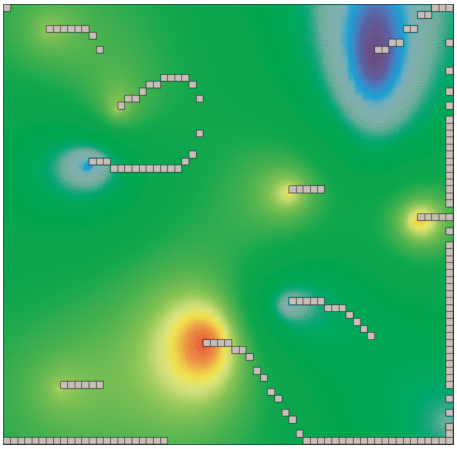

Figure 1: A two-dimensional scalar field (color indicates value), and a group of cells forming a seed set. For any iso-value, every connected component of the isosurface will intersect at least one cell in the set.

Although those algorithms accelerate searching time, a significant amount of memory is required to construct and maintain associating data structures. The hybrid approach presented by the seed set algorithm tends to keep the search data structures small. Several algorithms for seed set generation and spatial contour propagation, also called mesh propagation [7, 6], are described in [3]. The spatial contour propagation traces and constructs an isosurface component from a single cell by iterating breadth-first traversal through the face-adjacencies and triangulating until the whole connected surface is constructed. Other marching methods and level sets were also defined and used for many applications such as image enhancement and classification in [10, 12]. There, the underlying field changes over time according to some partial differential equation, pushing the interface in the direction of the field gradient.

Isosurface Extraction in Time Dependent Data Sets

Weigle et al. [20] considers time dependent data in 4-dimensional space. They first extract a 3-dimensional solid mesh on which the function value is the same as the iso-value. Then, a 2-dimensional isosurface is extracted from the solid mesh when time constraint t is specified. Although smoothly changing time-varying isosurface can be constructed, this technique is highly demanding in terms of space and time. Shen [13] proposed Temporal Hierarchical Index (THI) Tree as the extension of ISSUE algorithm [14]. Cells are adaptively coalesced based on temporal variation reducing the amount of space needed for the search structure. However, since data access pattern is not predictable, the entire data of each time step needs to be loaded to main memory. In order to minimize unnecessary I/O access, Sutton et.al. [18] proposed the Temporal Branch-on-Need tree. Neglecting temporal coherency, they consider each time step independently constructing a branch-on-need

octree. Each node contains min and max values of its sub-volume. When given (time step, iso-value) as a query, only data blocks corresponding to the octree nodes which intersect isosurface in a given time step are loaded from disk.

Feature & Component Tracking

Silver [17, 16] defines a feature in a volume as a region of interests which consist of cells satisfying predefined criteria. After features extraction, they perform correspondence matching test of features based on the degree of overlap to track the movement of features over time. Feature events are classified as continuation, creation, dissipation, bifurcation and amalgamation. Then, individual features can be isolated and quantified to give more information on the dataset. Although they provide enhanced visualization possibilities for time varying datasets, the correspondence-matching test is complex and not optimized for isosurfaces.

The contour tree is a graph that captures the global changes in contour topology of a static scalar field. The contour tree has been used mainly for finding minimal seed set [19, 4] or capturing the global structural information in contour topology of static scalar fields [1]. Tracking contour trees can be used to trace component topology accurately in a time sequence. Nevertheless, our algorithm tracks the contour topology while also extracting the isosurface itself.

3 Definitions and Concepts

A time dependent scalar field will be defined as $\mathcal{F} : \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}$. In discrete form we will have a sampling of this field as a finite sequence of scalar fields $\mathcal{F}_0, \mathcal{F}_1, \ldots \mathcal{F}_m$, each of which is a simple scalar field $\mathcal{F}_i : \mathbb{R}^n \rightarrow \mathbb{R}$, and all of them are defined on the same domain. When dealing with meshes and imposed scalar fields, this means we assume the geometry and connectivity of the mesh does not change over time, only the function values defined on the mesh nodes change over time. Similar to the continuity assumption in space inside mesh cells, we assume the changes over time are continuous. Hence, one can view time as an additional discrete sampling of a mesh with one higher dimension (time).

Given an iso-value scalar $\theta \in \mathbb{R}$ and a scalar field $F : \mathbb{R}^n \rightarrow \mathbb{R}$ an isosurface $I(\theta)$ is defined as $I(\theta) = \{\mathbf{x} \in \mathbb{R}^n | F(\mathbf{x}) - \theta = 0\}$. A time dependent isosurface of a time-dependent scalar field $\mathcal{F}$ will be a sequence of isosurfaces $I_0(\theta), I_1(\theta), \ldots, I_m(\theta)$, where $I_i(\theta) = \{\mathbf{x} \in \mathbb{R}^n | F_i(\mathbf{x}) - \theta = 0\}$. The basic query we support in time-dependent isosurface extraction is: "given a time interval $[i, j]$ and an iso-value θ, return $I_{i,j}(\theta) = \{I_i(\theta), I_{i+1}(\theta), \ldots, I_j(\theta)\}$". In the following discussion we will drop the parameter θ and assume we are dealing with the same θ value when we refer to an isosurface unless explicitly stated otherwise. Also, in order to distinguish between the input mesh and the isosurface mesh we use the term node for mesh 0-dimension elements and vertices for isosurface 0-dimension elements.

Each isosurface I_t can be broken into several separate connected components $I_t = \{C_1^t, C_2^t, ..., C_{m_t}^t\}$, each of which can be considered as a separate object. We will use the term *component* (denoted by C_i) as a general term for a connected component of an isosurface. The challenge in topology tracking of components over time is to find a correspondence ψ_t between the components of consecutive isosurfaces I_t and I_{t+1}, $\psi_t : I_t \rightarrow I_{t+1}$, for all t in the time sequence. This correspondence can map many components at time t to one component at time t+1, it can map one component at time t to many at time t+1, it can map a component at time t to no component at t+1, and leave components at time t+1without any components mapped to them. However, using such a correspondence, a classification of the modifications a component can undergo through time can be provided as follows:

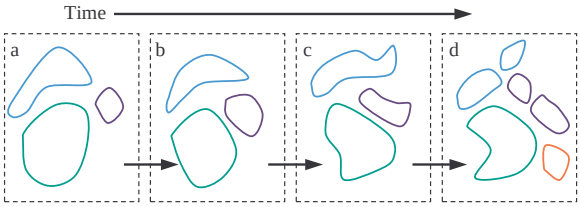

Figure 2: Overview of the time dependent isosurface extraction algorithm: (a) the first isosurface is extracted using the seed set algorithm, (b) and (c) temporal contour propagation is used to track components over time, along with (d) new components identification and creation using the seed set.

create A component $C_k^{t+1} \in I_{t+1}$ is created at time $t+1$if $\psi_t^{-1}(C_k^{t+1}) = \emptyset$.

disappear A component $C_k^t \in I_t$ disappears at time $t+1$if $\psi_t(C_k^t) = \emptyset$.

merge A component $C_k^t \in I_t$ is merged at time t+1if $\psi_t(C_k^t) \neq \emptyset$ and there exists at least one other component $C_j^t \in I_t, C_j^t \neq C_k^t$ such that $\psi_t(C_k^t) = \psi_t(C_j^t)$.

split A component $C_k^t \in I_t$ is split at time t+1if $|\psi_t(C_k^t)| > 1$, i.e. if there exist $\{C_{k_1}^{t+1}, \ldots, C_{k_m}^{t+1}\} \subset I_{t+1}$ such that $\psi_t^{-1}(C_{k_i}^{t+1}) = C_k^t$ for $1 \leq i \leq m$ and $m > 1$.

continue A component $C_k^t \in I_t$ continues at time t+1if there exists one and only one component $C_k^{t+1} \in I_{t+1}$ such that $\psi_t(C_k^t) = C_k^{t+1}$.

Depending on the direction we view time (from t to t+1or vice verse), merge and split and similarly create and disappear are opposite descriptions of the same phenomenon. This means each component of the isosurface at time t can either be a merged or a continued component from time $t-1$, or it must be a newly created component. Hence, we separate the components of I_t to mapped components I_t^M, and new components I_t^N. The mapped components contain all merged and continued components $I_t^M = \{C^t \in I_t | \exists C^{t-1} \in I_{t-1}, \psi_t(C^{t-1}) = C^t\}$, while the new components are all other components $I_t^N = I_t \setminus I_t^M$.

Our primary goal is to efficiently produce time dependent isosurfaces $I_{i,j} = \{I_i, I_{i+1}, \ldots, I_j\}$ and a sequence of correspondence maps $\psi_i, \psi_{i+1}, \ldots, \psi_j$ such that the topology change of all components can be tracked and visualized over time.

4 Extraction Algorithm

The key idea in our algorithm is to perform a *temporal contour (or surface) propagation* $\mathcal{TP} : I_t \rightarrow \{I_{t+1} \cup \emptyset\}$ on each component of the isosurface and track through time its evolution and interaction with other components. This propagation (see Section 5) will define the components correspondence between consecutive time steps, and allow us to track the topology events of components according to our classification. Hence, we define $(C_k^t, C_j^{t+1}) \in \psi_t$ if and only if $\mathcal{TP}(C_k^t) \subset C_j^{t+1}$ and if $\mathcal{TP}(C_k^t) = \emptyset$ then $\psi_t(C_k^t) = \emptyset$ (C_k^t disappears). However, there are still components C_k^{t+1} at time step t+1where $\psi_t^{-1}(C_k^{t+1}) = \emptyset$. These components are created at time step t+1and temporal propagation fails to find them. In turn, they are found and created using usual function domain search methods along with spatial propagation.

Figure 2 depicts the two basic types of contour propagation we use in the extraction algorithm: spatial contour propagation from seed cell and temporal contour propagation. At the starting time step i, all components are identified and constructed using the seed set (Figure 2(a)). Later, temporal contour propagations trace the movements of components and progressively approximate succeeding components by iterative local update in the surface neighborhood (Figure 2(b,c,d)). Seed sets along with spatial contour propagation locate and create new contours (red component in Figure 2(d)). It is important to note that the seed set propagation algorithm as well as temporal propagation will sometimes identify two different connected components of the surface as one component if they pass through the same isosurface cell. This situation means that the two components are in fact very close and probably are in the process of merging or splitting over time. Nevertheless this situation can be corrected by post processing of the components.

As a preprocessing step we construct for each time step i, $0 \leq i \leq m$, the seed set S_i of the field $\mathcal{F}_i$ and build the associated interval tree IT_i for indexing the S_i. Given the run-time query searching for isovalue θ in a time range $[i, j]$, the extraction algorithm is specified as follows:

1. Extract isosurface $I_i(\theta)$ for time step i using the seed set S_i. More specifically, After identifying seed cells which contain θ by traversing the interval tree IT_i, perform spatial contour propagation for each identified seed cell. This will construct all isosurface components of time step i: $I_i = \{C_1^i, , C_2^i, ..., C_n^i\}$.

2. For each time step t from i to $j-1$ perform the following:

   (a) After I_t is constructed, perform temporal contour propagation on each component to construct the corresponding components I_{t+1}^M of next time step t+1.

   (b) Use the seed set S_{t+1} to find all newly created components I_{t+1}^N. For each seed cell whose value range contains θ, check if that seed cell intersects some $C_k^{t+1} \in I_{t+1}^M$. If not, perform spatial contour tracking from the seed cell to construct the newly created isosurface component $C_j^{t+1} \in I_{t+1}^N$.

This approach exploits the high temporal coherency in time dependent data, resulting in near optimal performance in average case. It tracks the topology of isosurface components through time, allowing to trace, classify and quantify each component for enhanced visualization and query processing. The approach is applicable to any structured and unstructured grid of cells on which a scalar field is defined. In addition, it demands no complex correspondence matching test between components. Lastly, since the isosurface I_{t+1} is constructed progressively from I_t, time critical environments can display and use temporal isosurface-approximations using this algorithm.

5 Temporal Propagation

In this section, we describe the temporal surface propagation $\mathcal{TP}$ which is the basis of our isosurface tracking algorithm over time. As shown in Figure 2, $\mathcal{TP}$ is a method for tracing the deformation of individual isosurface component while the underlying field values change over time. Given the isosurface component C_k^t at time step t, the propagation algorithm traces the path of the surface movements incrementally accounting also for changes in the connectivity of C_k^t. This process gradually deforms C_k^t to the corresponding isosurface components C_j^{t+1} at time step t+1, which are temporally continuous with C_k^t.

```
for each isosurface cell of time t
   enqueue and mark every node in the cell
while the queue is not empty {
   dequeue a node n
   apply function value change to n
   if sign_t(n) ≠ sign_{t+1}(n) {
      update the local isosurface connectivity change
         (local means around n)
      check and update components merges
   } else if val_t(n) ≠ val_{t+1}(n)
      update the local surface geometry change
   loop on neighboring cells to n
      if the cell is a new isosurface cell
         enqueue and mark all its unmarked nodes
}
check for disappearing or split components
track topology changes of each component
```

Figure 4: Outline of the temporal contour propagation algorithm.

Given a scalar field defined on a mesh and an iso-value θ, the sign of a node in the mesh is "+" if the function value on the node is above θ, otherwise it is "-". If the function value is exactly θ, we use perturbation on the function value. Similar to previous isosurface extraction algorithms, we can identify all vertices of the isosurface as edges of the input mesh where the two end nodes have opposite signs. These edges are intersected by the isosurface creating the isosurface vertices. The position of which are found by interpolation on the edges between the nodes. Connecting the vertices of the isosurface inside the mesh cells will create the isosurface elements (e.g. triangles).

Let N_t be the union of all nodes of all isosurface cells at time t. At time t+1if the sign of a node $n \in N_t$ changes, the local connectivity of the isosurface related to n changes and should be updated. Additionally, some components might merge and others may be split by this update and this should be tracked. If the sign does not change but the value of n changes, we must update only the geometry of the isosurface by moving the position of isosurface vertices along the edges. If we apply these value changes gradually to all nodes in the isosurface cells, the isosurface of time t will gradually deform towards the isosurface of time t+1(see Figure 3). After the local updates, new isosurface cells can be formed around n. Any node in such cell which is not in N_t is added to N_t. This process continues until no more nodes are left at N_t (for example using a queue and iterating until the queue is empty). Figure 4 presents an outline of the $\mathcal{TP}$ algorithm.

5.1 Topology Tracking

Using temporal propagation we can identify four out of the five possible topological events of components. A merge occurs if during contour propagation a new isosurface cell is reached, which is already a cell of another component. Therefore this check is performed after updating local connectivity. In contrast, the split and disappearance check for all components is performed after all propagation is done. Continuing event occurs simply when no other event took place. The only event which is not identified using $\mathcal{TP}$ is the component creation event, for which the seed set is needed.

6 Implementation details

Time dependent scalar fields often imply very large data sizes. In such cases representation and storage layout become an important issue. This becomes more crucial when the data is too large to fit in memory and out-of-core accesses must be used.

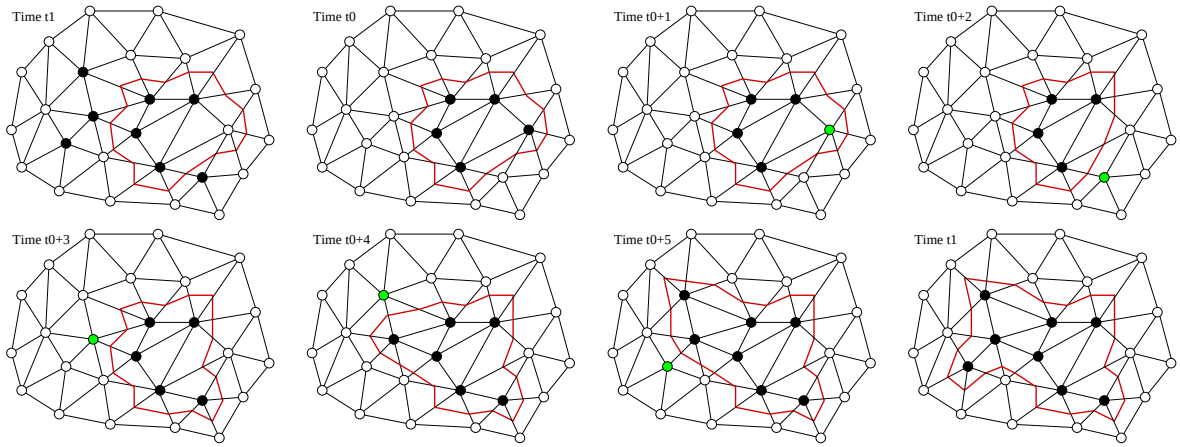

Figure 3: A 2D example of temporal contour propagation process for one isosurface component: in the top left the function values of nodes change at time $t1$ imposing some changes of node signs (black nodes depict $+$ sign and white nodes $-$ sign). This makes the isosurface of time $t0$ invalid. The $t0$ isosurface is gradually propagated (top row to the right and bottom row from left to right) by applying the value change of nodes leaving the queue (green nodes) and updating the contour locally.

6.1 Mesh Representation

Our temporal isosurface extraction algorithm can be applied to both structured and unstructured meshes. We assume the mesh connectivity does not change over time. In structured meshes the indices and adjacency information can be automatically determined on the fly. The format of a 3D structured data set can therefore be seen as an array of function values for each time step. Unstructured meshes require some extra storage for node, edge, face and cell adjacency information (not all of them are in fact used). However, in time dependent data this overhead is not so significant since it is not time varying, and needs to be stored only once.

The real question is how to store the time dependent scalar function values for each node. The two simplest choices are to store the whole mesh values for each time step together (favoring spatial coherency) or to store the whole sequence of values for each node together (favoring temporal coherency). Previous work [18] seem to suggest that especially in out-of-core situations, the former is more efficient. In our algorithm it is also preferable due to the heavy use of marching methods on the mesh (both for $\mathcal{TP}$ and for seed set contour creation), which benefit from spatial coherency.

More complex techniques or data arrangements for out-of-core situations can be combined with our basic approach. For instance, instead of loading the whole mesh for time step t+1, the $\mathcal{TP}$ algorithm only needs node function values in a small spatial range around I_t. Assuming the function has some Lifshitz bound, we can pre-fetch only cells in a certain distance from I_t isosurface cells, and march inside them. Similar to [18], one can collect these cell access requests and fetch all blocks containing the required cells together from disk. Note however that unlike [18, 13], the cells collected are not necessarily isosurface cells for I_{t+1}, but only cells that might be encountered during contour propagation.

In addition to the mesh structure and function values, we need to store the seed set and an interval tree for each time step. However in practice the number of seed cells does not exceed one or two percent of total number of cells. Therefore, the total size of this information storage is negligible.

6.2 Surface Representation

As presented earlier we represent an isosurface by a collection of components: $I_t = \{C_1^t, ..., C_m^t\}$. Each component holds a list of isosurface cells so that dynamic insertion and deletion during temporal contour tracking can be efficiently performed. Merging two or more components is performed by merging all their cell lists. In addition, this data structure can record the history of each component for visualizing the interaction among components (Figure 5).

Usually a surface triangle is composed of three vertices positions or indices. We represent each triangle vertex as a mesh edge index and an iso-value. This has several advantages. First, it relieves the need to actually calculate intersections many times. Second, it simplifies geometry update, for instance, when only the function values of the two end nodes of an edge change, but the signs do not change, no real update is needed at all. Third, it connects the isosurface representation to the mesh and supports tracing of cells and edges intersecting with the isosurface using the isosurface itself. The isosurface triangles can be extracted from the cells using known methods [9, 11, 15].

6.3 Complexity

Several seed set generation algorithms given in [2], have a tradeoff between the size of the set $|S| = n_s$ and the generation time ($O(n_c)$ where n_c is the number of cells in the mesh). In practice, the seed set size is one or more orders of magnitude smaller than n_c and the set generation algorithm is performed as a preprocessing stage. Let the number of unique minimum and maximum function values in $|S|$ be n_u ($n_u \leq 2n_s$). The storage cost of the interval tree for S is $O(n_s + n_u)$ and a search for the seeds of a specific iso-value takes $O(k + \log(n_u))$ time where k is the size of output. Using contour propagation from seed cells takes $O(n_i)$, where n_i is the number of isosurface cells. Temporal propagation may in fact take $O(n_c)$ depending on how far I_{t+1} is from I_t, nevertheless assuming the function is Lifshitz, we can bound the range of propagation and visit $O(n_i)$ cells.

7 Results

We have tested several time dependent data sets querying different iso-values and using different visualization techniques. Preprocessing for generating seed set is performed on each time step. Table 1 provides information on our test data sets. Results were computed on a Silicon Graphics ONYX2 InfiniteReality2 system with 24 R12000 processors and 25GB main memory.

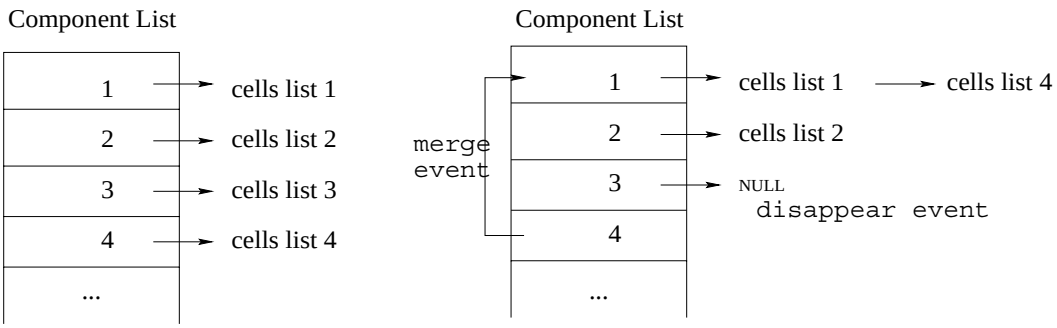

Figure 5: Left: the data structure for isosurface representation as an array of components. Each component holds a list of mesh cells where the iso-surface lays. The surface triangles can easily be extracted from the cells. Right: over time component 3 disappears and component 4 is merged with component 1. Using this data structure such topological events can easily be tracked.

Data	Resolution	Time steps	Avg. #seeds
Vorticity magnitude	$128 \times 128 \times 128$	30	1.05%
Ocean speed	$512 \times 256 \times 30$	112	1.11%
Gas dynamics	$256 \times 256 \times 256$	144	0.90%

Table 1: Information on time-varying test data sets.

Table 2 gives some timing results and the relationship with the number of extracted triangles for different iso-values. The timings are dependent on the iso-value because the number of triangles in the isosurface varies with the iso-value. These timings include finding isosurface cells, topology tracking and surface extraction (which includes computing vertex positions and normals and triangulation within the cells). As can be seen from figure 6, our isosurface extraction algorithm scales approximately linearly with respect to the number of triangles extracted. This means the search for isosurface cells and the topology tracking do not impose any overhead on the triangulation, and implies near optimal complexity (the order of the size of the isosurface).

Component tracking and identification becomes most significant when there are many evolving components in a single isosurface, and it is difficult to distinguish between them (see Figure 7). Moreover, component separation can be used to identify interesting components based on their quantified information such as surface area, volume or gradients. More advanced visualization primitives can be defined to trace and interrogate dynamic data such as tracking isolated components movements with gradual decay of history positions (Figure 8). More efficient visualizations may be created by rendering only sub groups of more interesting components (in terms of number of triangles, a single or a few components may take only a small percent of a whole isosurface), and mixing components from different iso-values may be carried out more easily (Figure 9). Moreover, in time-critical situations the surface propagation may be interrupted and approximations of the real isosurface may still be rendered (Figure 10).

8 Conclusion and Future Work

We have presented a fast and progressive algorithm for extracting time dependent isosurfaces and tracking their components over time. Observed average time complexity is linear in the number of isosurface triangles. The algorithm tracks the evolution of each

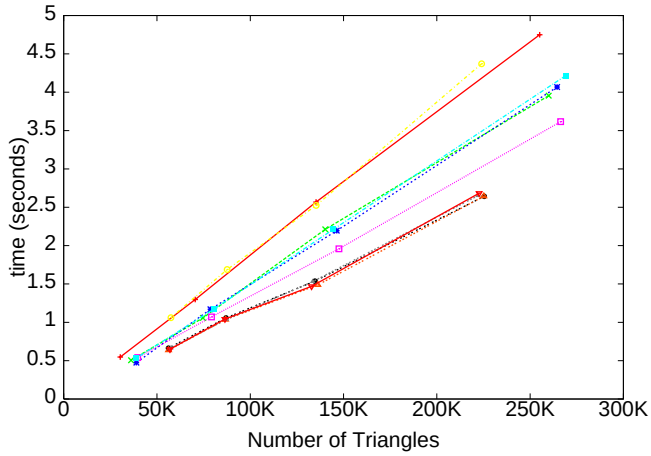

Figure 6: Number of surface triangle vs. extraction time in different data sets. Each line represents plots for different iso-values in one time step. All tests show that extraction time scales linearly with the number of triangles and implies near optimal solution.

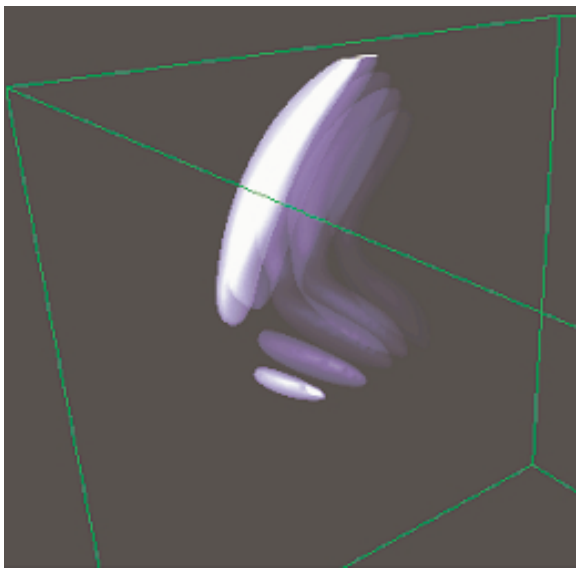

Figure 8: Isolated component tracking over time using gradual decay of history positions.

Vorticity magnitude data (#triangles and seconds)										
Isovalue	timestep 1		timestep 2		timestep 3		timestep 4		timestep 5	
4.0	255156	4.750	259974	3.955	264465	4.068	266357	3.617	269059	4.216
5.0	135284	2.568	140214	2.213	146360	2.195	147558	1.958	144532	2.212
6.0	70480	1.297	74604	1.058	78666	1.170	79214	1.070	80532	1.173
7.0	30330	0.547	36096	0.506	38928	0.474	39678	0.540	38940	0.536
Ocean speed data (#triangles and seconds)										
Isovalue	timestep 1		timestep 2		timestep 3		timestep 4		timestep 5	
0.3	224012	4.371	225442	2.643	225188	2.644	222106	2.652	222682	2.683
0.4	135316	2.524	134650	1.537	136222	1.488	133450	1.531	132826	1.469
0.5	87620	1.688	87004	1.058	86734	1.035	87154	1.037	86416	1.042
0.6	57488	1.060	56166	0.663	56180	0.637	57524	0.666	56954	0.649

Table 2: Isosurface extraction performance and number of triangles from vorticity magnitude and ocean speed data sets for different isovalues.

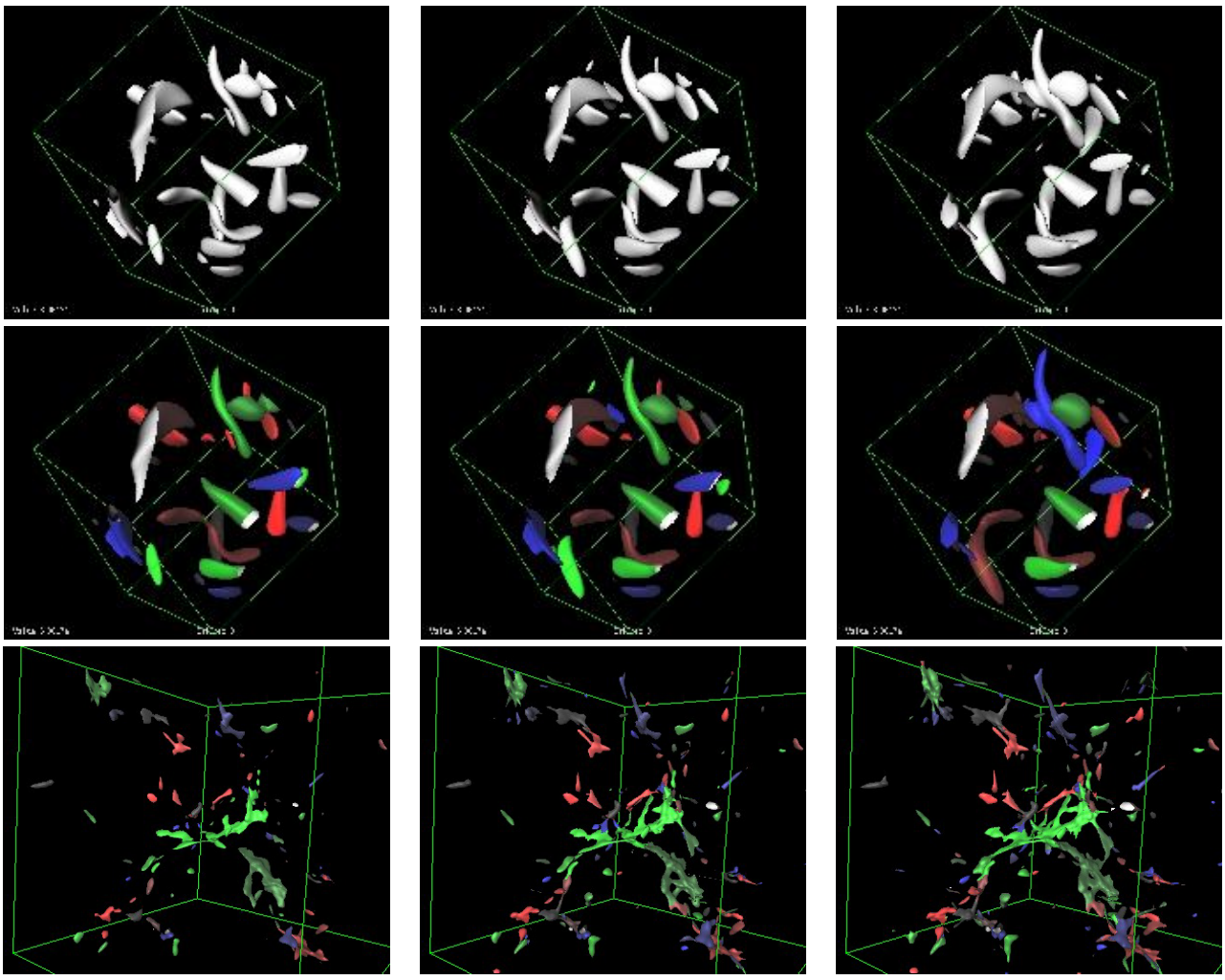

Figure 7: Standard time dependent isosurface visualization (vorticity magnitude top) vs. visualization using isosurface component identification and tracking (vorticity and gas dynamics middle and bottom). Note for example, how the long green surface at the top frame of the middle series changes its color when merging with a blue component.

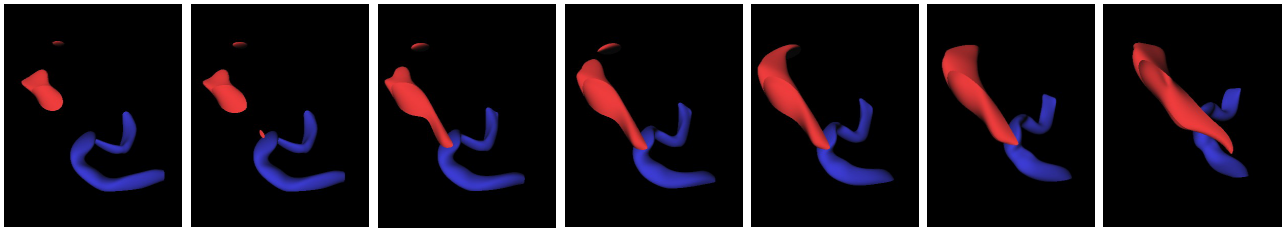

Figure 9: Two isosurface components of different iso-value isolated and tracked over time. Note how the blue component deforms much less than the red component.

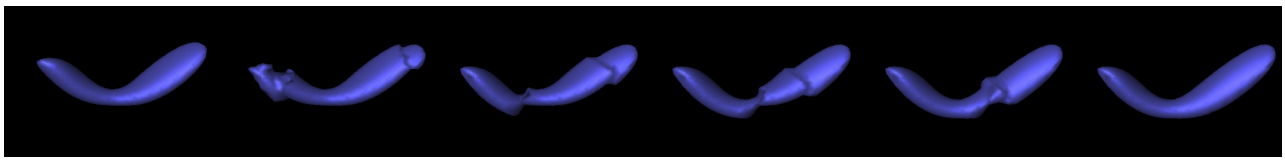

Figure 10: An isosurface component of time t gradually evolves to the component at time t+1. Although it may create serious artifacts, this process can be interrupted at any point and the approximated surface used for rendering.

contour component and finds useful structural information by tracking their topology changes. This is used to create enhanced visualizations of complex time-varying data set, and assist data analysis.

Future directions include the definition of other interrogative visualization techniques which would employ the tracking and topological information gathered by this algorithm. Incorporating this technique in a more strenuous time-critical environment and in a more general scheme to support out-of-core situations. We believe the benefits of exploiting all three coherencies inherent to the data (time, space and function value) by this algorithm would be demonstrated in those situations further.

9 Acknowledgement

We are grateful to Dr. V. Fernandez, S. Y. Chen and Dr. Silver for providing the vorticity magnitude data set. We would also like to thank Prof. Detlef Stammer for providing access to the oceanography simulation data.

References

[1] C. L. Bajaj, V. Pascucci, and D. R. Schikore. The contour spectrum. In *IEEE Visualization Conference*, 1997.

[2] C. L. Bajaj, V. Pascucci, and D. R. Schikore. Seed sets and search structures for optimal isocountour extraction. Technical Report 99-35, TICAM, Univerisity of Austin, Texas, 1999. http://www.ticam.utexas.edu/CCV/papers/cont-tog.pdf.

[3] C. L. Bajaj, V. Pascucci, and D.R. Schikore. Fast isocontouring for improved interactivity. In *Preceedings of the 1996 Symposium for Volume Visualization*, 1996.

[4] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. In *Computational Geometry : Theory and Applications*, 2001.

[5] P. Cignoni, P. Marino, E. Montani, E. Puppo, and R. Scopigno. Speeding up isosurface extraction using interval trees. In *IEEE Transactions on Visualization and Computer Graphics*, 1997.

[6] C.T. Howie and E.H. Black. The mesh propagation algorithm for isosurface construction. In *Computer Graphics Forum*, 1994.

[7] T. Itoh and K. Koyamada. Automatic isosurface propagation using an extrema graph and sorted boundary cell lists. In *IEEE Transactions on Visualization and Computer Graphics*, 1995.

[8] Y. Livnat, H.-W. Shen, and C. Johnson. A near optimal isosurface extraction algorithm using the span space. In *IEEE Transactions on Visualization and Computer Graphics*, 1996.

[9] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM SIGGRAPH '87*, 1987.

[10] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[11] B. K. Natarajan. On generating topologically consistent isosurfaces from uniform samples. *The Visual Computer*, 11(1):52–62, 1994.

[12] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, Cambridge, second edition, 1999.

[13] H.-W. Shen. Isosurface extraction in time-varying fields using a temporal hierarchical index tree. In *IEEE Proceedings of Visualization '98*, 1998.

[14] H.-W. Shen, C. Hansen, Y. Livnat, and C. Johnson. Isosurfacing in span space with utmost efficiency(issue). In *IEEE Proceedings of Visualization '96*, 1996.

[15] H. W. Shen and C. R. Johnson. Sweeping simplices: A fast iso-surface extraction algorithm for unstructured grids. In *Proceedings of IEEE Visualization '95*, pages 143–150, 1995.

[16] D. Silver. Object-oriented visualization. In *IEEE Computer Graphics and Applications*, 1995.

[17] D. Silver and X. Wang. Tracking and visualization turbulent 3d features. In *IEEE Transactions on Visualization and Computer Graphics*, 1997.

[18] P. Sutton and C. Hansen. Isosurface extraction in time-varying fields using a temporal branch-on-need tree(t-bon). In *IEEE Proceedings of Visualization '99*, 1999.

[19] M. van Kreveld, R. van Oostrum, C. L. Bajaj, V. Pascucci, and D. R. Schikore. Contour trees and small seed sets for isosurface traversal. In *the 13th ACM Symposium on Computational Geometry*, pages 212–220, 1997.

[20] C. Weigle and D. Banks. Extracting iso-valued features in 4-dimensional scalar fields. In *IEEE Visualization Conference*, 1995.

[21] J. Wilhelm and A. Van Gelder. Octrees for faster isosurface generation. In *ACM Transactions on Graphics '92*, 1992.