Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2004)
O. Deussen, C. Hansen, D.A. Keim, D. Saupe (Editors)

Efficient Isosurface Tracking Using Precomputed Correspondence Table

Guangfeng Ji [†] and Han-Wei Shen[‡]

Department of Computer and Information Science
The Ohio State University
Columbus, Ohio, USA

Abstract

Feature tracking is a useful method for visualizing and analyzing time-varying scalar fields. It allows scientists to focus on regions of interest and track their evolution and interaction over time. To allow the user to freely explore the data set, features must be tracked in an efficient manner. In this paper, we present an efficient time-varying isosurface tracking algorithm. Unlike the previous algorithms which compute the corresponding isosurface components in the adjacent time steps by performing expensive computation at run time, our algorithm can rapidly identify corresponding isosurfaces by performing simple table lookup operations. This table, called the correspondence lookup table, can be computed at a preprocessing stage. The idea behind our approach is that the correspondence relationship can only change at critical isovalues in R^3 or R^4 and remains unchanged between adjacent pairs of critical isovalues. With our algorithm, isosurfaces can be tracked in an efficient manner with minimal overhead.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Methodology and Techniques]: Interaction Techniques

1. Introduction

Scientists are now able to perform large scale time-varying simulations to model phenomena that are complex and highly unsteady. For example, meteorologists often perform simulations to study how storms form and evolve. In [Koe01], scientists studied the autoignition phenomena by tracking time-dependent features defined as high intermediate concentrations. To analyze data generated from those simulations, visualization has become an essential tool. Besides the basic goal of presenting an intuitive view of the data, an important aim of visualization is to highlight salient data features and offer unique insight into the underlying problem. For time-varying data, an effective visualization tool should also compute and track features over time in an accurate and efficient manner.

Displaying isosurfaces is a common way to characterize features in a scalar field. By displaying points of a constant threshold value specified by the user, isosurfaces reveal the geometric structure of the objects represented by the data set. For time-varying data, visualization of time-varying isosurfaces offers insights into how the data evolves over time. However, one major challenge to compute dynamic isosurfaces is the enormous size of the data set. A time-varying data set may contain hundreds, or even thousands of time steps, and each time step can have 512^3 or 1024^3 grid points. It is thus prohibitively expensive to extract and track isosurfaces from huge data sets.

Many data sets from scientific simulations contain multiple components, and scientists often want to track their evolution individually. To provide a scientist insight into how the features evolve, an effective time-varying isosurface tracking algorithm with minimal computation complexity is essential.

In this paper, we present an algorithm that facilitates effi-

[†] email: jig@cis.ohio-state.edu
[‡] email: hwshen@cis.ohio-state.edu

cient time-varying isosurface tracking. The primary observation is that the correspondence relationship between isosurface components of two consecutive time steps can change only at critical isovalues in R^3 or R^4 where the number of isosurface components in that space undergoes a change. The correspondence between the isosurface components will remain the same when the isovalue changes between two adjacent critical isovalues. Based on this property, the correspondence between the isosurface components for any possible isovalue can be precomputed and stored into a table. When an isosurface component of an arbitrary isovalue and time step is selected by the user, its corresponding component(s) in the next time step can be rapidly identified by looking them up in the correspondence table. Compared with the existing algorithms [JSW03, SW96, SW98] which compute corresponding isosurfaces in consecutive time steps at run time, our isosurface tracking method offers much better efficiency with minimal overhead.

The organization of the paper is as follows. We first review the previous work in section 2, and then present our algorithm to construct the correspondence lookup table in section 3. In the same section, we also provide a rigorous analysis and justification of our approach and describe an optimized algorithm to compute the correspondence lookup table with better efficiency and smaller storage overhead. Our isosurface tracking algorithm using correspondence lookup table is also described in section 3. Test results are presented in section 4 and the paper is concluded with the future direction of this research.

2. Previous Work

Researchers have proposed various techniques to track time-varying features. Arnaud *et al.*[ADM92] tracked 2D cloud patterns and used area overlap to determine correspondence. Banks and Singer [BS95] used a predictor-corrector method to reconstruct and track vortex tubes from turbulent time-dependent flows. Samtaney *et al.*[SSZC94] tracked features using their centroids, masses, volumes and circulations (in 2D). Reinders *et al.*[RPS01] calculated a set of attributes, such as center point position, volume, mass, and best fitting ellipsoid for every feature in every frame and used these data to track features through a predication/verification scheme. Silver and Wang [Sil95, SW96, SW97, SW98] had an observation that when the temporal resolution of the underlying data is high enough, corresponding features in adjacent time steps usually overlap. Given that, they first extracted features manifested as interval volumes and stored them in appropriate data structures. Then, correspondences between features in consecutive time steps were 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

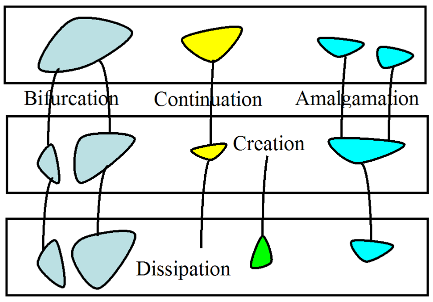

Figure 1: *The evolutionary events that a feature may experience.*

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 with possible shape deformation and change of position, orientation, etc.
- Creation: a new object appears.
- Dissipation: an object disappears.
- Bifurcation: an object splits into several objects.
- Amalgamation: several objects merge into a single one.

Chen *et al.*[CSJ03] extended the work by Silver and Wang [Sil95, SW96, SW97, SW98] to track features in distributed AMR (Adaptive Mesh Refinement) datasets within a distributed computing environment. The resulting feature tree allows a viewer to watch how a multi-level isosurface changes over time, space and across different resolutions.

It is also worth mentioning that there is a rich literature in computer vision on motion tracking [Bal82, AN88, ST94, CCH92]. 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.

In this paper, we track isosurfaces with great efficiency by using a precomputed correspondence lookup table. It is based on the fact the correspondence relationship between isosurfaces of two consecutive time steps can change only at critical isovalues in R^3 or R^4 and remains unchanged between any two adjacent critical isovalues. Here critical isovalues are the isovalues at which the isosurface will change its number of components. Therefore, the correspondence relationship between isosurfaces from any two consecutive time steps will only change finite amount of times and thus the correspondence lookup table will contain finite entries and can be precomputed. With this correspondence table, isosurface tracking can be achieved by simple table lookup operations. These operations virtually take no time and thus our algorithm has minimal overhead. In the next section, we will first explain the reason why the correspondence relationship changes in this manner, and then introduce our method to build the correspondence lookup table and the way to utilize it in the isosurface tracking process.

3. Isosurface Tracking

3.1. Tracking by using higher dimensional isosurfacing

If a data set has been sampled well temporally, according to Silver and Wang's observation, corresponding isosurface features between two adjacent time steps overlap. A straightforward way to find out how an isosurface component evolves is first extracting all the isosurface components in the next time step and then performing an overlap test with each of the components. This could be very time consuming, since it is very likely that there exist a large number of components in the next time step and only a very small number of components overlap with the tracked isosurface. In our previous work [JSW03], we tracked isosurfaces efficiently by using higher dimensional isosurfacing. The key observation that leads to efficient tracking of isosurfaces in R^3 is that the isosurface component and the component(s) it overlap(s) with in the next time step belong to the same isosurface component in R^4. This isosurface component in R^4 can be extracted by propagating any 4-cell that intersects with the tracked isosurface component within these two time steps. From this property, in order to track where an isosurface component in R^3 evolves, our algorithm [JSW03] first extracts the isosurface component in R^4 that contains the tracked isosurface component in R^3. We then slice the generated isosurface component in R^4 at the next time step to get the corresponding isosurface component(s) in R^3 which correspond(s) to the tracked isosurface component in R^3.

3.2. Generation of the correspondence lookup table

By using higher dimensional isosurfacing, we greatly improved the efficiency of tracking local isosurface components. However, there is still a significant amount of overhead involved. The major part of the overhead is due to isosurfacing in higher dimensions (R^4). If the isosurface contains multiple components and only few components are tracked, the overhead is small. However, in some cases where the tracked isosurface components contain a large part of the whole isosurface, the overhead of extracting the isosurface in R^4 can make the method even slower than the straightforward method. In this section, we will introduce a method to build a correspondence lookup table which contains the correspondence information between isosurfaces of two consecutive time steps across the whole value range. With this correspondence table, isosurface tracking can be achieved by simple table lookup operations which take almost no time. Therefore, our isosurface tracking algorithm has minimal overhead.

To facilitate our discussion, we define that two isosurface components correspond to each other if and only if they belong to the same isosurface component in R^4. Now let's examine the correspondence between the isosurfaces of these two time steps across the whole value range, i.e., from the minimum data value to the maximum value of both time steps. We will see that the isosurfaces at both time steps change their shapes continuously, with possible topology changes at critical isovalues. We should also observe that the correspondence relationship between these two isosurfaces changes in the following manner: the correspondence starts at some configuration at the minimum value and remains the same for some value interval. Then at some value the correspondence changes and remain unchanged for another interval. This procedure repeats until we reach the maximum value, where both isosurfaces vanish.

A 2D example is illustrated in figure 2. At the minimum value of both time steps, there is no isosurface component at either time step. So the correspondence relationship is empty. Just above the minimum value, a red component appears at time step t_0. The correspondence relationship changes to $\{C_{red}$->empty$\}$. As the isovalue increases, the red component changes its shape at t_0 and no new components appear at t_1 and thus the correspondence relationship remains unchanged. As the isovalue continues to increase, a green component appears at t_1. Since the red component overlaps with the green one therefore they belong to the same isosurface component in R^4, the correspondence relationship changes to $\{C_{red}$->$C_{green}\}$. Again, the red component and the green one change their shapes as the isovalue increases, but the correspondence relationship remains the same. But when the isovalue reaches an isovalue v_0, the red component splits into a yellow component and a blue one. The yellow component has no overlap with the green one but the blue one does. The correspondence relationship changes to $\{C_{yellow}$->empty, C_{blue}->$C_{green}\}$. This correspondence relationship will remain unchanged until the green component splits into a grey component and a black one, and both overlap with the blue component. Now the correspondence relationship changes to $\{C_{yellow}$->empty, C_{blue}->C_{grey}, C_{blue}->$C_{black}\}$. This procedure will repeat until these two isosurfaces vanish at the maximum value.

A natural question is which values cause the correspondence relationship to change. We will assume that every field value at the grid points of the data set is unique for each pair of adjacent time steps. This could be easily satisfied by a small perturbation to equivalent field values at the grid points for any real data set. We will also assume that the underlying data field is composed of simplicial meshes. We know that at some isovalue, the isosurface

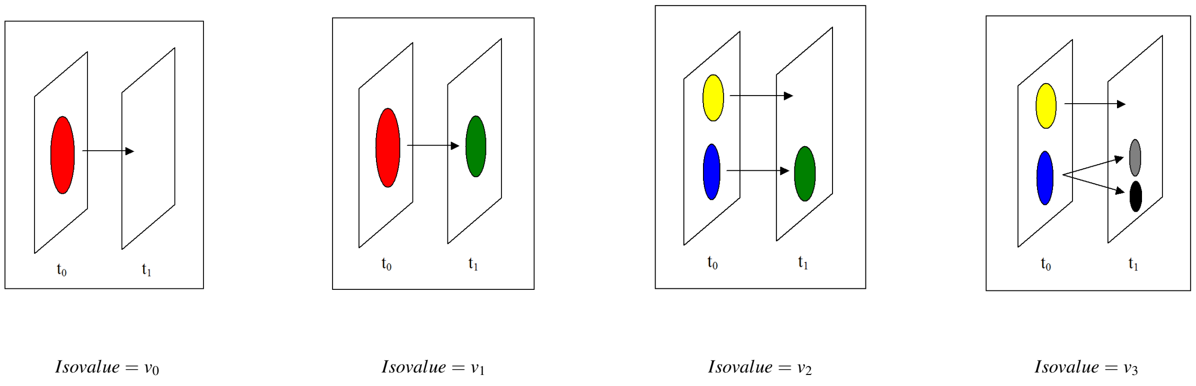

Figure 2: *Four snapshots to illustrate how the correspondence relationship between isosurfaces from two consecutive time steps changes when the isovalue increases from v_0 to v_3 ($v_0 < v_1 < v_2 < v_3$).*

in R^3 would change its number of components, i.e., a component may merge with another component, or it may split into several components. A new component may appear, or a component disappears. We name these events critical events and these isovalues critical isovalues in R^3 [WSHH02, GP00, KOB*97, TV98, PCM02, CSA03, CS03]. This also happens to isosurface in R^4, that is, when the isosurface in R^4 changes its number of components at some isovalues, we call these isovalues critical isovalues in R^4 (Please note that within this paper, the isosurface in R^4 is only extracted within two consecutive time steps). A key property that leads to our efficient isosurface tracking is that the correspondence change can only occur at the critical isovalues in R^3 or R^4. That is to say, the correspondence change can only occur when the number of components of the isosurface in R^3 or R^4 changes. To show this is true, we show the following observation.

Observation: Assume an isosurface component in R^3 is generated by slicing an isosurface component in R^4. When the isovalue changes and no critical event happens in R^3 or R^4, the isosurface component in R^4 changes into another new isosurface component in R^4. Likewise the isosurface component in R^3 changes into a new isosurface component in R^3. The new isosurface component in R^3 can still be generated by slicing the new isosurface component in R^4.

Let's say an isosurface component C_i at t_0 and another isosurface component C_j at t_1 correspond to each other, i.e., they belong to the same isosurface component C_k in R^4. If no critical event happens in R^3 or R^4 when the isovalue changes, based on our observation, C_i, C_j and C_k all change but after the change of isovalue, C_i and C_j can still be generated by slicing C_k, i.e., they belong to the same isosurface component in R^4. Therefore, they still have correspondence. We can see that when isovalue changes and if no critical event occurs in R^3 or R^4, two previously corresponding components will still correspond to each other. Therefore, the correspondence relationship between isosurface components of two adjacent time steps can only change at critical isovalues in R^3 or R^4. There appears to be several counter cases to this argument. We will disprove some of the counter cases in the following.

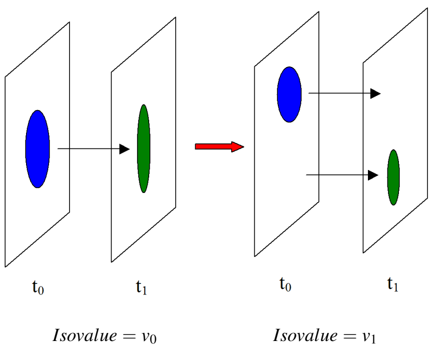

Figure 3: *Two previously corresponding components do not correspond to each other any more and the number of components in R^3 remains the same when the isovalue changes.*

Case 1: In figure 3 (Figure 3, 4 and 5 are 2D examples), the isosurfaces at both time steps contain only one component. Initially they correspond to each other. However, when the isovalue changes, all of a sudden they have no correspondence any more. The number of components in R^3 remains the same and we will show that the number of components in R^4 must change, i.e., this case can not happen without change of number of isosurface components in R^4.

These two components in figure 3 originally correspond to each other, so they belong to the same isosurface component in R^4. After the isovalue is changed, the component

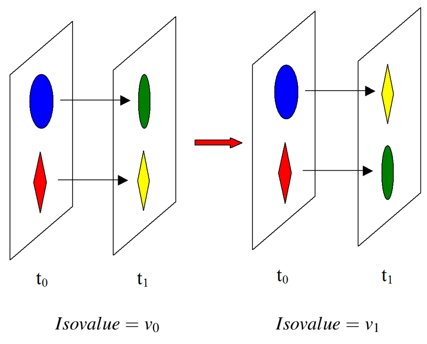

Figure 4: *A plausible case where two pairs of components switch their correspondence without the change of number of isosurface components in R^3 and R^4 when the isovalue changes.*

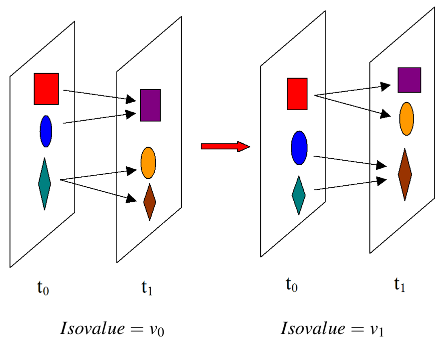

Figure 5: *Another seemingly reasonable case: one component at each time step changes its correspondence when the isovalue changes but without the change of number of isosurface components in R^3 and R^4.*

at t_0 corresponds to an empty component at t_1 and an empty component at t_0 corresponds to the component at t_1. It means that these two components belong to two different isosurface components in R^4. Therefore, there must be a split of the original isosurface component in R^4 which leads to the change of number of components in R^4, i.e., this case can not happen without change of number of isosurface components in R^4.

Case 2: In figure 4, the isosurfaces at both time steps consist of two components. At some isovalue, the circle component at t_0 corresponds to another circle component at t_1 and the diamond component at t_0 corresponds to another diamond component at t_1. However, when the isovalue changes, suddenly the correspondence switches. This seems to be a reasonable case where the correspondence changes but the number of components in R^3 and R^4 remains the same.

Originally the two diamond components belong to the same isosurface component in R^4 and the two circle components belong to another component in R^4. When the isovalue changes, the diamond components changes into new diamond components and the circle components into new circle components. Since the two diamond components belonged to the same component in R^4 before the change of isovalue and assuming no change of number of components in R^3 or R^4 therefore no critical event occurs, according to the observation, the two new diamond components will still belong to the same isosurface component in R^4, i.e., the two diamond components still correspond to each other. So do the two circle components. Therefore, the case in figure 4 cannot happen without change of number of components in R^3 or R^4.

Case 3: In figure 5, the isosurface at both time steps are composed of three components. Two components at t_0 merge into one component at t_1 and the other component at t_0 splits into two at t_1. When the isovalue is changed, the correspondence changes as illustrated in figure 5. Intuitively, this could also happen and the correspondence changes without change of number of components in R^3 or R^4.

Originally the square and circle components at t_0 and the square component at t_1 belong to an isosurface component in R^4. The diamond component at t_0 and the square and circle components at t_1 belong to another component in R^4. After the isovalue changes, the circle component at either time step belongs to a different isosurface component in R^4, which implies at least two critical events happen simultaneously. In contradiction, we have assumed that every field value at the grid points of the data set is unique for each pair of adjacent time steps and the underlying data field is composed of simplicial meshes. Since critical events can only happen at vertices of simplices, we get a conclusion that at any isovalue, critical events can only happen at most in one time step in R^3, or at most between one time interval in R^4. Therefore it is true that at most one critical events can happen among all the time steps. There is a contradiction here. So the case in figure 5 cannot happen.

Based on our analysis above, the change of correspondence relationship can only occur at the critical isovalues in R^3 or R^4. Since there are finite critical isovalues in R^3 and R^4, the correspondence relationship will only change finite amount of times across the whole value range. Thus the correspondence table will have a finite amount of entries and can be precomputed. Between two adjacent critical isovalues, the correspondence relationship will remain the same. Therefore, a representative value within the interval can be used to calculate its correspondence relationship. The correspondence table can then be used in isosurface tracking. When an isosurface component is selected by the user, we can query which component(s) it corresponds to by simply looking them up in the correspondence table using the isovalue and the ID of the selected component as the index. The tracking overhead is minimized because only a simple table lookup is involved.

Putting it into another perspective, we know that contour trees [KOB*97, TV98, PCM02, CSA03, CS03] describe the topology of a scalar field. A contour tree is composed of supernodes and superarcs. Supernodes are vertices of the scalar field where the number of isosurface components changes. The values of supernodes are critical isovalues. Superarcs connect two supernodes and within a superarc the associated isosurface component will not change its number of components. The correspondence relationship between two isosurfaces of any two consecutive time steps across the whole value range can be represented by connecting the contour trees of these two scalar fields. The correspondence can only change at either the supernodes of two contour trees in R^3 or the supernodes of the contour tree of the 4D field, which is the composition of these two 3D fields. Actually a good way to store the correspondence relationship between isosurfaces of two consecutive time steps within two adjacent critical isovalues is to store the superarc correspondence, since there is a one-to-one correspondence between superarcs and isosurface components. An advantage of this scheme is that the superarc provides a good way to enumerate the isosurface component and it can also be utilized to generate the seed from which the isosurface component can be propagated.

3.3. Optimization in generation of the correspondence table

Based on the fact that the correspondence can change only at critical isovalues in R^3 or R^4 and remains constant between two consecutive critical isovalues, the correspondence relationship will only change finite amount of times and thus the correspondence table will only have finite entries and can be precomputed. A brute force method to generate the table can work as follows: First find all the critical isovalues in R^3 and R^4 and sort them by field values. Within every two adjacent critical isovalues, the correspondence relationship remains constant. Therefore, a representative value such as the midvalue can be used to retrieve the correspondence relationship. Many methods can be used to find the correspondence between isosurface components of two adjacent time steps at a given isovalue [JSW03, SW96, SW98]. The correspondence relationship from all adjacent critical isovalue intervals forms the correspondence table.

The correspondence table contains all of the correspondence relationships between isosurfaces of two consecutive time steps across the whole value range. Within every two adjacent isovalues, the correspondence relationship remains constant. At either end of the interval, due to the existence of a critical point in R^3 or R^4, the correspondence may change. If a critical event in R^3 happens, it means that a component at either t_0 or t_1 appears, disappears, merges with another component or splits into several components. Other components at this time step retain their topology. Since no critical event occurs to these components, they will correspond to the same components as in the previous critical isovalue intervals in the correspondence table. Therefore, there is no need to recompute their correspondences and only components whose topology changes need to recompute their correspondence. If a critical event in R^4 happens, a component in R^4 changes its topology. Recomputation of correspondence only needs to be performed for those components in R^3 which belong to the affected component in R^4. Therefore, by utilizing the value coherence, the correspondence relationship can be updated with respect to the previous critical isovalue interval in the correspondence table. This difference updating method has two advantages. First, the computation time can be reduced significantly, since no computation is wasted on components whose correspondence remains the same. Second, since two adjacent critical isovalue intervals share most of the correspondence, only the correspondence relationship for components whose correspondence changes needs to be stored. This information is very small compared with the correspondence relationship of all components. By utilizing the value coherence, the computation time and storage overhead of the correspondence table can be significantly reduced.

3.4. Isosurface tracking using correspondence lookup table

The correspondence lookup table records all the correspondence information between isosurfaces of two adjacent time steps across the whole value range. After it is built, it can be used in the isosurface tracking process. We know that each isosurface component corresponds to exactly one superarc in the contour tree. An isosurface component can be generated by propagating a seed cell which is found by using the seed information associated with the superarc. Therefore, when an isosurface component is selected, its corresponding superarc can be found immediately. Recall that the correspondence table stores the isosurface component correspondence information as superarc correspondence. So we can query the correspondence table with the current isovalue and the superarc the selected component belongs to and the result is the superarc(s) the current superarc corresponds to. The corresponding isosurface component(s) can then be generated by propagating seed cell(s) found by using seed information associated with the superarc(s). Compared with our previous work [JSW03] where the corresponding isosurface components are identified by isosurfacing in R^4, our current method only needs simple table lookup operations to find the corresponding isosurface components. Note in both algorithms, isosurfacing in R^3 is needed to extract the corresponding components.

Notice that the corresponding isosurface components generated in this way do not distinguish the degree of overlap. A verification stage [JSW03] can be performed to guarantee that the overlap between the tracked isosurface component and the corresponding isosurface component is significant enough to be considered as correspondence.

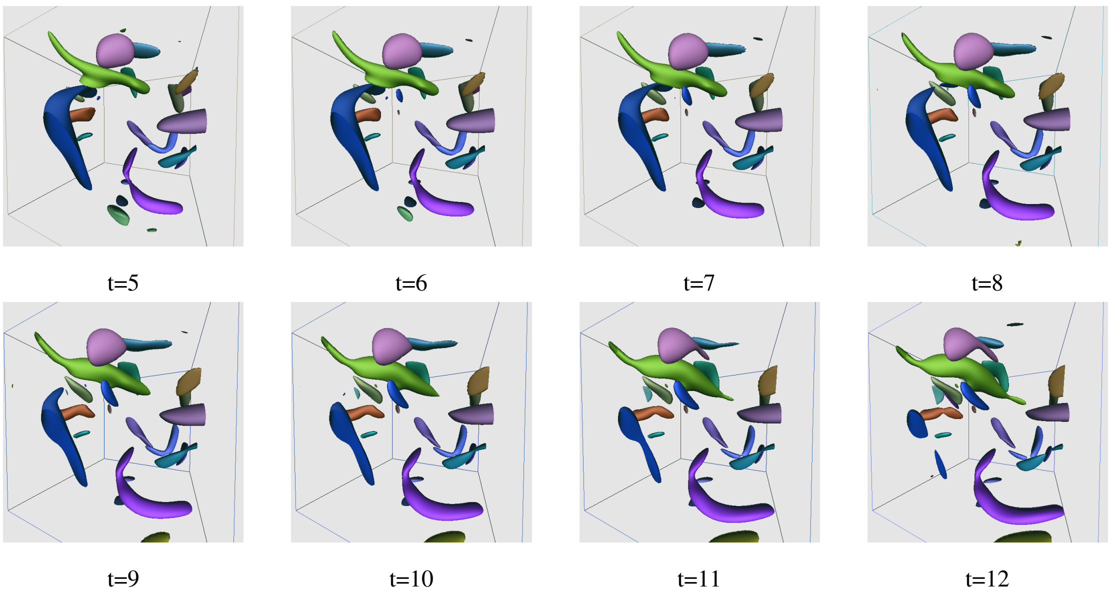

Figure 6: *All isosurface components are tracked and the tracking information determines the coloring of the components. 8 out of 99 time steps are shown.*

4. Results

We have tested our algorithm using a 128*128*128 vorticity magnitude data set with 99 time steps. The machine we used was a Pentium IV 1.4GHz PC with 768MB memory.

Figure 6 shows an example of tracking a whole isosurface that consists of multiple components. The time-varying isosurface was extracted with isovalue 6.3. The correspondence tables between isosurfaces of every two consecutive time steps are precomputed. At run time, these tables are used to determine the correspondence among isosurface components. In figure 6, a component is colored in such a way that it has the same color as the components it would evolve into. When a component is created, a new color is assigned to it. When two or more components merge into a component, the 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 results, we used volume as the criterion to determine the dominance, i.e., the current component would follow the color of the previous largest component.

Figure 7 shows the tracking of two isosurface components. These two time varying components were extracted with isovalue 6.2. Snapshots from six time steps are shown. Again the precomputed correspondence tables are used to determine which components the current one would evolve into. In figure 7, these two component merges into one component at t=4 and it then bifurcates into two at t=15. The larger component bifurcates again at t=21. At t=22, the smallest component dissipates. Figure 8 also shows the tracking of another two isosurface components. The isovalue of the isosurface was 6.9.

Time step	5	6	7	8	9
Total time	0.401	0.42	0.421	0.431	0.431
Correspondence determination time	0.00	0.00	0.00	0.00	0.00
Component extraction time	0.341	0.35	0.351	0.361	0.361
Verification time	0.06	0.07	0.07	0.07	0.07
Overhead	0.06	0.07	0.07	0.07	0.07
Size of tracked component(s)	148K	148K	152K	154K	155K
Size of corresponding component(s)	148K	152K	154K	155K	155K

Table 1: *The time to track the whole isosurface illustrated in figure 6 (in seconds). Timing from 5 time steps are shown due to space limitation.*

The total tracking time and how the time breaks down are also shown in table 1 for the case in figure 6, table 2 for the

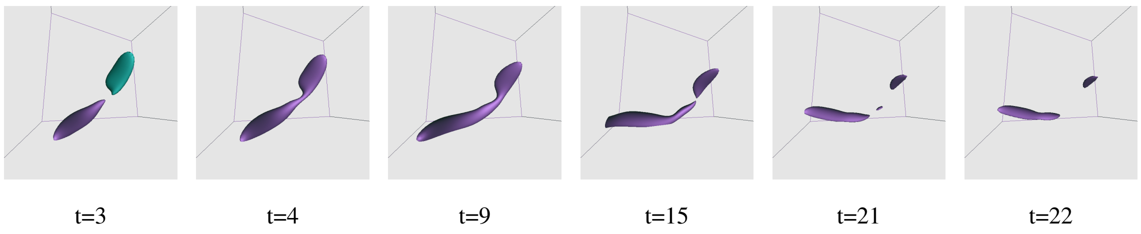

Figure 7: *Two isosurface components are tracked. These two components merge into a component at t=4 and it then bifurcates into two at t=15. The larger component bifurcates again at t=21. At t=22, the smallest component dissipates.*

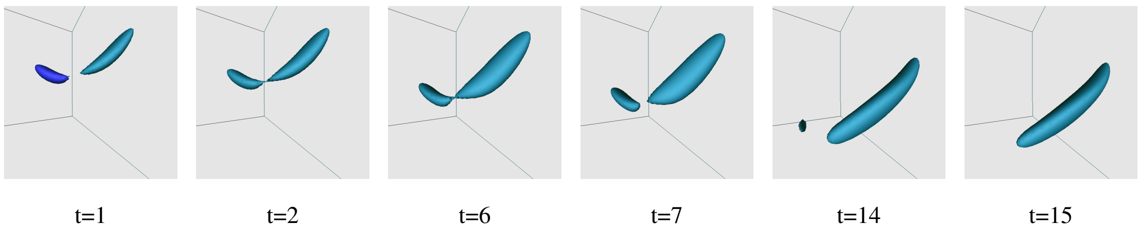

Figure 8: *Two isosurface components are tracked. Amalgamation occurs at t=2; bifurcation occurs at t=7 and dissipation at t=15.*

Time step	4	9	15	21
Total time	0.04	0.05	0.05	0.03
Correspondence determination time	0.00	0.00	0.00	0.00
Component extraction time	0.03	0.04	0.04	0.03
Verification time	0.01	0.01	0.01	0.00
Overhead	0.01	0.01	0.01	0.00
Size of tracked component(s)	13.6K	17.5K	18.5K	13.2K
Size of corresponding component(s)	15.1K	18.0K	17.8K	10.7K

Table 2: *The time to track two isosurface components illustrated in figure 7 (in seconds). Timing from time step 4, 9, 15 and 21 are shown due to space limitation.*

Time step	1	2	6	7	14
Total time	0.01	0.02	0.02	0.02	0.02
Correspondence determination time	0.00	0.00	0.00	0.00	0.00
Component extraction time	0.01	0.01	0.02	0.02	0.02
Verification time	0.00	0.01	0.00	0.00	0.00
Overhead	0.00	0.01	0.00	0.00	0.00
Size of tracked component(s)	3.5K	4.3K	6.3K	7.0K	8.1K
Size of corresponding component(s)	4.3K	5.0K	7.0K	7.4K	7.7K

Table 3: *The time to track two isosurface components illustrated in figure 8 (in seconds). Timing from time step 1, 2, 6, 7 and 14 are shown due to space limitation.*

case in figure 7 and table 3 for the case in figure 8. The total tracking time includes the correspondence determination time, isosurface component extraction time and the verification time. The correspondence determination time is used to determine which components correspond to the current component. Since only very simple table lookup operations are performed, the time is minimized and our result shows 0 all the time, i.e., it is too small to be measured by the system clock. Isosurface component extraction time is the time to extract the corresponding isosurface component, which

Time interval	0-1	1-2	2-3	3-4	4-5
Running time speedup	16.9	18.8	16.3	15.3	16.0
Output file size reduction rate	5.1	5.5	5.4	5.0	4.1

Table 4: *The comparison between two methods to generate the correspondence lookup table.*

involves operations such as finding a seed associated with a superarc and propagating the seed to get the component. Verification time is used to verify if the current component has significant overlap with the generated component. If the overlap is significant, the correspondence holds; otherwise there will be no correspondence. Overhead shown in each table is the sum of correspondence determination time and verification time and it indicates the cost of tracking. The running time of our tracking algorithm is proportional to the size of the corresponding isosurface components. The size information is also shown in each table. The complexity of our tracking algorithm is dependent on the feature size rather than the size of the data. Furthermore, compared with our previous work [JSW03] in which the corresponding isosurfaces are identified by isosurfacing in R^4, our current work minimizes the correspondence determination time to almost 0. Other overhead is also highly reduced. For the cases in figure 7 and figure 8 where two isosurface components are tracked, the overhead is reduced by at least 5 times compared with our previous work. The overhead reduction rate will be even larger when more components are tracked.

The precomputed correspondence lookup tables are used in all of our examples for the correspondence query. We have two methods to generate the lookup table, the brute force method and the optimized difference updating method. In table 4, we show the comparison of these two methods in terms of running time and output file size. In this test, we only generate the correspondence table between the maximum value and 6.0 for every two consecutive time steps. The reason is simply that it takes too long for the brute force method to finish the computation of the correspondence relationship across the whole value range. For the optimized difference updating method, the average output file size is 8K and the average running time is 17 seconds, with approximately 2 percent of the total critical isovalue intervals processed when the correspondence tables between the maximum value and 6.0 are generated. We can see from table 4 that the optimized method performs much better in terms of both running time and generated file size. Actually if the correspondence tables for the whole value range was generated, the performance difference would be bigger. The reason is as follows. Recall that the brute force method extracts the correspondence relationship between all isosurface components from two adjacent time steps within each critical isovalue interval and stores it to disk. Conversely the optimized method will only extract the correspondence difference with the previous interval and only the difference is stored into disk. Furthermore, for the vorticity data set, the complexity of the isosurface increases as the isovalue decreases. For example, at time step 0, the isosurface at isovalue 8.0 only contains 10 components and 30K triangles. But the isosurface at isovalue 4.0 contains 43 components and 600K triangles. So the brute force method will become slower and generate a larger size of correspondence relationship as the isovalue decreases. In contrast, the optimized method only deals with the difference with respect to the previous value interval, which contains a very small number of components (usually 1 or 2). Its running time remains almost constant across different intervals and the same is true for the size of generated correspondence relationship. Therefore, with the whole value interval computed, we should see a much bigger performance difference. An estimation of the performance difference between these two would be at least 30 times in running time and 10 times in generated file size.

5. Conclusions and Future Work

In this paper, we introduce an efficient algorithm to track isosurfaces. A precomputed correspondence lookup table enables us to find out how an isosurface component evolves by simply table lookup operations with the isovalue as the index. The rationale behind our approach is that the correspondence relationship between isosurfaces of two adjacent time steps can change only at critical isovalues in R^3 or R^4 and remains unchanged between adjacent critical isovalues. The computation of the correspondence lookup table is also optimized with better efficiency and less storage overhead. By using this correspondence lookup table, isosurface tracking is performed in an extremely efficient manner with minimal overhead.

Our current work is based on the assumption that the data set has enough temporal samples so that corresponding isosurface components will overlap. But in practice, this may not be always guaranteed, especially for small or fast moving isosurface components. In our future work, we will investigate an efficient tracking technique for small and fast moving isosurface components when the temporal sampling rate of the data is insufficient.

6. Acknowledgement

This work is supported in part by NSF grant ACR-0118915, NASA grant NCC-1261, DOE Early Career Principal Investigator Award DE-FG02-03ER25572, and NSF ITR grant ACI-0325934. We wish to thank the anonymous reviewers for their valuable suggestions.

References

[ADM92] ARNAUD Y., DESBOIS M., MAIZI J.: Automatic tracking and characterization of african convective systems on meteosat pictures. *Journal of Applied Meteorology 31*, 5 (1992), 443–453. 2

[AN88] AGGARWAL J., NANDHAKUMAR N.: On the computation of motion from sequences of images – a review. *Proceedings of the IEEE 76*, 8 (1988), 917–935. 2

[Bal82] BALLARD D.: *Computer Vision*. Prentice-Hall,Inc, Englewood, New Jersey, 1982. 2

[BS95] BANK D., SINGER B.: A predictor-corrector technique for visualizing unsteady flow. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (1995), 151–163. 2

[CCH92] CARLBOM I., CHAKRAVARTY I., HSU W.: Integrating computer graphics, computer vision, and image processing in scientific applications. *Computer Graphics 26*, 1 (1992), 8–17. 2

[CS03] CARR H., SNOEYINK J.: Path seeds and flexible isosurfaces using topology for exploratory visualization. In *Joint Eurographics - IEEE TCVG Symposium on Visualization 2003* (2003), pp. 49–58. 4, 6

[CSA03] CARR H., SNOEYINK J., AXEN U.: Computing contour trees in all dimensions. *Computational Geometry 24*, 2 (2003). 4, 6

[CSJ03] CHEN J., SILVER D., JIANG L.: The feature tree: Visualizing feature tracking in distributed amr datasets. In *Proceedings of IEEE symposium on Parallel and Large-Data Visualization and Graphics 2003* (2003), pp. 103–110. 2

[GP00] GERSTNER T., PAJAROLA R.: Topology preserving and controlled topology simplifying multiresolution isosurface extraction. In *Proceedings of Visualization 2000* (2000), pp. 259–266. 4

[JSW03] JI G., SHEN H.-W., WENGER R.: Volume tracking using higher dimensional isosurfacing. In *Proceedings of Visualization 2003* (2003), pp. 209–216. 2, 3, 6, 9

[KOB*97] KREVELD M., OOSTRUM R., BAJAJ C., PASCUCCI V., SCHIKORE D.: Contour trees and small seed sets for isosurface traversal. In *Proceedings of 13th Annual ACM Symposium on Computational Geometry* (1997), pp. 212–220. 4, 6

[Koe01] KOEGLER W.: Case study: Applications of feature tracking to analysis of autoignition simulation data. In *Proceedings of IEEE Visulazation 2001* (2001), pp. 461–464. 1

[PCM02] PASCUCCI V., COLE-MCLAUGHLIN K.: Efficient computation of the topology of level sets. In *Proceedings of Visualization 2002* (2002), pp. 187–194. 4, 6

[RPS01] REINDERS F., POST F., SPOELDER H.: Visualization of time-dependent data using feature tracking and event detection. *The Visual Computer 17*, 1 (2001), 55–71. 2

[Sil95] SILVER D.: Object-oriented visualization. *IEEE Computer Graphics and Applications 15*, 3 (1995). 2

[SSZC94] SAMTANEY R., SILVER D., ZABUSKY N., CAO J.: Visualizing features and tracking their evolution. *IEEE Computer 27*, 7 (1994), 20–27. 2

[ST94] SHI J., TOMASI C.: Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition 1994* (1994), pp. 593–600. 2

[SW96] SILVER D., WANG X.: Volume tracking. In *Proceedings of Visualization 1996* (1996), pp. 157–164. 2, 6

[SW97] SILVER D., WANG X.: Tracking and visualizing turbulent 3d features. *IEEE Transactions on Visualization and Computer Graphics 3*, 2 (1997), 129–141. 2

[SW98] SILVER D., WANG X.: Tracking scalar features in unstructured datasets. In *Proceedings of Visualization 1998* (1998), pp. 79–86. 2, 6

[TV98] TARASOV S., VYALYI M.: Construction of contour trees in 3d in o(nlog n) steps. In *Proceedings 14th Annual ACM Symposium on Computational Geometry* (1998), pp. 68–75. 4, 6

[WSHH02] WEBER G., SCHEUERMANN G., HAGEN H., HAMANN B.: Exploring scalar fields using critical isovalues. In *Proceedings of Visualization 2002* (2002), pp. 171–178. 4