Time-varying Isosurface Tracking with Global Optimization

Guangfeng Ji*
The Ohio State University

Han-Wei Shen†
The Ohio State University

ABSTRACT

Feature tracking plays an important role in the understanding of time-varying data sets since it allows scientists to focus on regions of interest and track their evolution and interaction over time. In this paper, we focus on the tracking of time-varying isosurfaces and advocate that the commonly used local tracking techniques based on either volume overlapping or the most similar attributes are often not sufficient. To address this problem, we propose a global optimization algorithm that can identify the best possible matching isosurface component for each tracked component. Our algorithm will first find a list of correspondence candidates for each source isosurface component and then use a global optimization to minimize the overall cost to match the source isosurface to the target isosurface. An efficient method to precompute this correspondence relationship within an isovalue interval is also introduced. With our global optimization algorithm and the precomputed correspondence relationship, isosurfaces can be tracked in a more accurate and efficient manner.

CR Categories: I.3.6 [Methodology and Techniques]: Interaction Techniques—;

Keywords: isosurface tracking, global optimization

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 [14], 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 [17] 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. Visualization of time-varying isosurfaces offers insights into how a time-varying data evolves over time. However, commonly used animation techniques may not be always effective, especially for tracking the evolution of complex features in a long time sequence. In addition, the enormous size of the data set makes the computation of the animation very expensive. To tackle this problem, previously researchers have developed various feature tracking techniques to study time-varying scalar fields. For instance, Silver and Wang developed volume tracking techniques based on spatial overlap [26, 27, 28, 29]. Samtaney *et al.*[24] and Reinders *et al.*[22] track time-varying features based on the feature attributes such as position, volume, mass, etc. Ji and Shen [12, 11] track time-varying isosurfaces and interval volumes using higher-dimensional geometry. These techniques can be generally characterized as *local tracking techniques*, where the tracking of each local feature is performed independently with various greedy search schemes in the local surrounding to find the best match.

In this paper, we advocate that the local tracking techniques do not always produce globally best matching results, in particular when there exist small or fast-moving objects for which the time-varying field is not sufficiently sampled. To address the issues, we propose a global optimization algorithm for tracking time-varying isosurfaces. Our algorithm takes into account the global configuration of the features and ensures that every isosurface component is matched to the best possible corresponding components in the next time step. To achieve this, our algorithm will first find a list of correspondence candidates for each source isosurface component and then use a global optimization scheme to minimize the overall cost to match the source isosurface to the target isosurface. By utilizing the coherence between isosurfaces within adjacent isovalues, we present a method to precompute only a finite number of isosurface correspondences to depict the evolution history of isosurfaces generated by arbitrary isovalues. With our global optimization algorithm and the precomputed correspondence relationship, isosurfaces can be tracked in an accurate and efficient manner.

The organization of the paper is as follows. We first review previous works in section 2, and then present our global optimization algorithm in section 3. In section 4, a way to precompute the correspondence relationship within an isovalue interval is introduced. Test results are presented in section 5 and the paper is concluded with the future work of this research.

2 RELATED WORK

Researchers have proposed various techniques to track time-varying features. Banks and Singer [5] used a predictor-corrector method to reconstruct and track vortex tubes from turbulent time-dependent flows. Arnaud *et al.*[2] tracked 2D cloud patterns and used area overlap to determine correspondence. The tracking methods basically fall into two categories: volume overlapping based methods [26, 27, 28, 29, 12, 11] and attributes based methods [24, 22]. Silver and Wang [26, 27, 28, 29] observed that corresponding features in adjacent frames usually overlap when the temporal sampling rate of the underlying data is high enough. Based on the observation, correspondences between features in consecutive frames are identified using a two-stage process including an overlap and a best matching test. In the overlap test, spatially overlapped features from consecutive frames are identified and the number of intersecting nodes is also computed. The best matching test involves inspecting the ratio of the number of intersecting nodes versus the average volume among all combinations of overlapped features, with the combination of the maximum ratio as the corresponding feature(s). Samtaney *et al.*[24] tracked features using their center point positions, masses, volumes and circulations (in 2D). Each feature is matched to the feature(s) in the next frame whose center point position is the closest to its center point and the volume and mass are also within a prescribed tolerance. Reinders

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

et al.[22] also 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.

In our previous work [12], we tracked isosurfaces and interval volumes efficiently by using higher dimensional isosurfacing. The key observation we had is that the isosurface and interval volume component and the component it overlaps with in the next frame belong to the same isosurface or interval volume component in R^4. Therefore, in order to track where an isosurface or interval volume component in R^3 evolves, our algorithm [12] first extracts the isosurface or interval volume component in R^4 that contains the tracked component in R^3. We then slice the generated component in R^4 in the next frame to get the component in R^3 that corresponds to the tracked component. In [11], we reported that the overlapping relationship between isosurfaces of two consecutive frames can change only at critical isovalues in R^3 or R^4 and remains unchanged between any two adjacent critical isovalues. Therefore, the overlapping relationship between isosurfaces from any two consecutive frames will only change finite amount of times and thus the overlapping lookup table will contain finite entries and can be precomputed. With this overlapping table, isosurface tracking can be achieved by simple table lookup and verification operations.

Feature tracking identifies how a feature evolves and interacts over time. Based on how the topological structure of a feature evolves, one of the following events can occur:

- 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.*[9] extended the work by Silver and Wang [26, 27, 28, 29] 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 [4, 1, 25, 7]. 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.

3 Isosurface Tracking by Global Optimization

Given two isosurfaces in two time steps (hereafter referred to as two *frames*) each of which contains multiple components, one way to establish the correspondence between them is to associate every component in one frame with the component(s) of its locally best match in the other frame. The best local match criterion can be defined either by the component which gives the maximum overlap [26, 27, 28, 29, 12, 11], or the component which has the most similar attributes (center point position, volume, etc) [24, 22]. While those local matching algorithms are effective for various applications, there are many cases where the local matching schemes do not offer the best result globally. For example, in the example shown in figure 1, both isosurfaces contain multiple fast-moving components. The locally best matching component for C_0^1 (C_i^j refers to component j at t=i) is C_1^0 because it has maximum overlap with C_0^1 and their center point positions and volumes are also the most similar. However, globally C_0^1 should correspond to C_1^1

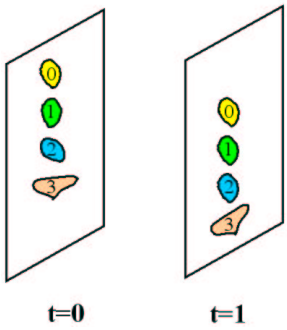

Figure 1: The locally best matching component for C_0^1 is C_1^0. But globally C_0^1 should be matched to C_1^1.

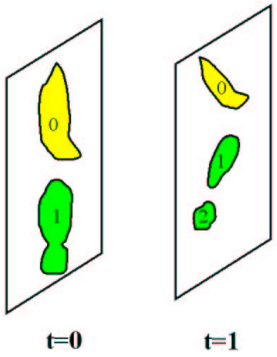

Figure 2: If C_0^0 is first matched, it will correspond to C_1^0 and C_1^1, no matter the maximum overlapping criterion or the most similar attribute criterion is used. However, C_0^0 should be matched to only C_1^0.

because the entire isosurface in fact moves downwards. Similarly, in the example shown in figure 2, both components shrink as time evolves. If C_0^0 is the first to be matched, it will be matched to C_1^0 and C_1^1. This is because that they have the maximum overlap with C_0^0 if volume overlapping criterion is used, or they have the most similar center point position and volume if the most similar attribute criterion is used. However, globally only C_1^0 should be matched to the C_0^0. From these two examples, we can see that the scheme of local matching does not always produce the best results globally.

To address the problem, a global matching technique is needed to ensure that every component is matched to the best possible corresponding components in the next frame. In the following, we present a global optimization algorithm to track time-varying isosurfaces. For every single component and *compound component* where multiple single components are involved in a single topological event (merge, split, disappear, or create), we will first identify a list of correspondence candidates in the next frame. A cost is evaluated for matching the component (single or compound) to each of its possible correspondence candidates. Based on the costs, a global optimization is performed to minimize the overall cost to match the source isosurface to the target isosurface. The output of the global optimization then defines the correspondence between two isosurfaces.

3.1 Correspondence Candidates

As mentioned above, in many cases the local matching scheme will not produce the best matching result. A global optimization is needed to guarantee that every component matches to its best possible candidate. In our time-varying isosurface tracking algorithm, the first step is to identify from the next frame all the possible correspondence candidates for each of the isosurface components (single or compound). This will reduce the search space so that the global optimization can be done in a more efficient manner.

3.1.1 Correspondence Candidates for Single Component

Although in theory every isosurface component in the next frame can be considered as a correspondence candidate, this is often too conservative since when a component evolves, it usually moves to a nearby position with its shape deforming gradually and its volume changing slightly. To reduce the search space during optimization, the choice of the correspondence candidates for a component can be limited to its overlapping components in the next frame and all of the neighbors of the overlapping components. Note that this can be relaxed if the time varying data set is very sparsely sampled. For example, the neighbors of the neighbors can be also included. Based on this principle, all the possible corresponding candidates of a component can be found out by first locating its overlapping components and finding all the neighbors of the overlapping components. The overlapping components can be quickly identified based on a precomputed overlap lookup table [11]. For a component which has no overlapping components in the next frame, its closest component in the next frame can be used. To get the neighborhood of the component, the voronoi diagram composed of all the center points of each component is first calculated. The neighborhood relationship can then be quickly located by looking up the voronoi diagram.

3.1.2 Correspondence Candidates for Compound Component

When a component evolves, it may disappear, split into several, or merge with other components. A new component may also be created. Therefore, in order to detect those evolutionary events, we will need to consider the possibility of matching a group components to a single component, matching a single component to a group of components, matching a component to an empty component, and matching an empty component to a single component. Such groups of components are called as *compound components*. Therefore, in addition to finding correspondence candidates for single components, correspondence candidates for compound components should also be identified. Furthermore, both single and compound component can be the correspondence candidate for a single component. Compound components are identified to detect evolutionary events, therefore, they are named as merging candidates, splitting candidates, disappearing candidates and creating candidates, respectively.

To identify compound components, we do not necessarily want to consider all the possible combinations of single components from an isosurface since the number of compound components will grow exponentially in this way. In fact, a specific evolutionary event can happen to only a subset of the components that have specific characteristics. For example, two components that are very far away from each other will be less likely to merge. Also a component will not split into two components which are very far away from each other. Therefore, we will first need to identify reasonable compound components before finding correspondence candidates for them.

- Merging Candidates
  Not all combinations of components will merge. One obvious criterion for merging candidates is that these components should be close enough to each other spatially. One way to measure the spatial closeness between any two components is the center point distance between them. However, it is not always a good choice, as illustrated in figure 3. Although the distance between the two large components and that between the two small ones are the same, these two small components are less likely to be a merging candidate, since the distance between these two components is quite large compared to their own sizes. Therefore, both the center point distance and the size of the component should be considered to identify merging candidates. A simple and effective way to measure the size of a component is its bounding box size. Another good criterion for merging candidates is the shape continuity. The merging candidates can merge into a single component in the next frame, and each single component of the merging candidate should be similar to some part of the single merged component. Therefore, the shape of the merging candidate should exhibit some continuity. A good way to measure this characteristic is the continuity of the skeleton. Together with the spatial closeness, it gives good criteria to identify merging candidates. We will describe how to extract the skeleton in a later section. Notice that merging candidates can only be in the first frame of a pair of frames.

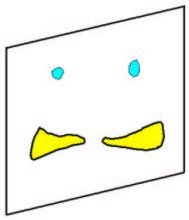

Figure 3: Distance should not be the only criterion to identify merging/splitting candidates.

- Splitting Candidates
  Split is the opposite event of merge. It can be treated as a merge if the time sequence is reversed. Therefore, the criteria to identify splitting candidates are the same as the criteria to identify merging candidates. Splitting candidates can only be in the second frame of a pair of frames.
- Disappearing Candidates
  The first criterion that should apply to a disappearing candidate is that its volume should be small. Another criterion is that there should be no component similar to this component in its neighborhood in the next frame, since otherwise this component could correspond to that component. Disappearing candidates can only be in the first frame of a pair of frames.
- Creating Candidates
  Creation is the opposite event of disappearance. Creation can be treated as disappearance if time sequence is reversed. Therefore, the criteria to identify creating candidates are the same as the criteria to identify disappearing candidates. Creating candidates can only be in the second frame of a pair of frames.

After all the compound components are identified, their correspondence candidates need to be found. The correspondence candidates for a merging candidate is the union of the corresponding candidates of every single component it contains. The correspondence candidates for a disappearing candidate should be the correspondence candidates of the component it contains plus an empty component. Furthermore, if the correspondence candidate of a single component contains part of a splitting candidate, the whole

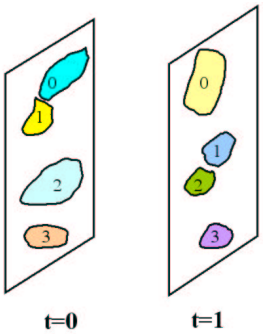

Figure 4: An example to show the correspondence candidates of single and compound components.

splitting candidate should be also included into the correspondence candidates of that single component.

Compound components are only candidates for an evolutionary event. Whether an evolutionary event will take place or not still depends on the result of the global optimization. Therefore, even though the correspondence candidates for a compound component are put into the choice list for the global optimization, the correspondence candidates for each single component it contains will also be in the choice list. As an example, in figure 4, C_0^0 and C_0^1 are identified as merging candidates. C_1^1 and C_1^2 are identified as splitting candidates. Therefore, in addition to find the correspondence candidates for single component C_0^0, C_0^1, C_0^2 and C_0^3, correspondence candidates for compound component $\overline{C_0^0 C_0^1}$ also needs to be found. If the correspondence candidates of some component include C_1^1 or C_1^2, compound component $\overline{C_1^1 C_1^2}$ should also be included in the correspondence candidates of that component. Based on these rules, the following lists of correspondence candidates will be the input to the global optimization algorithm:

C_0^0:$\{C_1^0, C_1^1, C_1^2, \overline{C_1^1 C_1^2}\}$
C_0^1:$\{C_1^0, C_1^1, C_1^2, \overline{C_1^1 C_1^2}\}$
C_0^2:$\{C_1^0, C_1^1, C_1^2, C_1^3, \overline{C_1^1 C_1^2}\}$
C_0^3:$\{C_1^1, C_1^2, C_1^3, \overline{C_1^1 C_1^2}\}$
$\overline{C_0^0 C_0^1}$:$\{C_1^0, C_1^1, C_1^2, \overline{C_1^1 C_1^2}\}$

3.2 Global Optimization

A global optimization is needed to find the best possible matching result from the multiple lists of correspondence candidates. In this section, we will first define the cost function used to match a single or compound component in the current frame to each of its correspondence candidates in the next frame. Then an optimization algorithm will be introduced to minimize the overall cost to match the isosurface components in two frames. The correspondence criteria, as a way to measure how much two components correspond, will be defined in the following.

3.2.1 Correspondence Criteria

Correspondence criteria determines what components should be considered as correspondence, and measures how much two components correspond. Because an isosurface can contain fast-moving or small components which may have no overlap with their corresponding components, overlap can not be used as the sole correspondence criterion. When a component evolves, it often moves to a nearby position with its shape deforming gradually and its volume changing slightly. Therefore, its corresponding component should

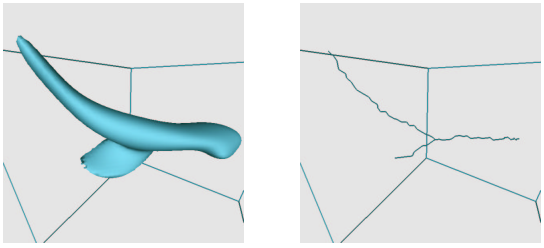

Figure 5: A component and its skeleton. Notice that the skeleton contains three end nodes and one junction node.

be some component in the next frame close to it spatially with similar shape and volume. So spatial closeness and volume and shape similarity should be used as the correspondence criteria.

Spatial closeness is one of the criteria that measure whether two components match. The center position, which can be calculated by averaging all vertex positions of the isosurface component, gives a good measurement of the average position of the component. Thus it could be used to measure the spatial closeness of two components. However, as pointed out in figure 3, the component size should also be considered when measuring the spatial closeness. The volume of a component, which can be calculated by a precomputed B-Spline function [3] or by a voxel propagation method [12], is another criterion to measure if two components correspond.

Many methods can be used to determine if two components have a similar shape [10, 16, 19, 21], although most of the methods are very expensive. For the purpose of interactive isosurface tracking, having an efficient method is important. Among all the shape descriptions, the skeleton [13, 21], which is composed of all the points lying in the center of the object with respect to its boundary, gives a good and compact description of the object shape. We use a topological thinning method [13] to extract the skeleton from an isosurface. This topological thinning method [13] is based on a hit-or-miss evaluation using a set of $3 \times 3 \times 3$ neighborhood masks. Each mask represents a case where the center voxel should remain. The set of masks are applied sequentially to all the voxels. If a mask matches the neighbors of a voxel, it is a skeleton voxel; otherwise it is removed. The scanning process is applied iteratively until the result is stable and no more voxels can be removed. After the skeleton is extracted, all the skeleton voxels are connected together. This process is simple since the topological thinning method guarantees the connectivity of the skeleton voxels. However, care should be taken to ensure that a loop does not occur at a junction node. Figure 5 shows an example of a component and its skeleton. A skeleton is composed of three types of nodes: end nodes with degree 1, normal nodes with degree 2 and junction nodes with degree more than 2. The total number of end and junction nodes gives a description of the topology of the skeleton. If the difference between the total number of end and junction nodes of the skeletons of two components is larger than a threshold, their shape similarity degree is 0. Otherwise, the ratio between the lengths of their skeletons is used to evaluate their shape similarity degree. This simple method is efficient and effective for isosurface tracking.

3.2.2 Cost Function

A given single or compound component may have many corresponding candidates. Some candidates are more likely to be the best matching component than others, if they are spatially closer to the source component and have more similar volume and shape. To evaluate the likelihood, we define a function to measure the cost of matching a component to its correspondence candidate. The closer between the component and its correspondence candidate and the

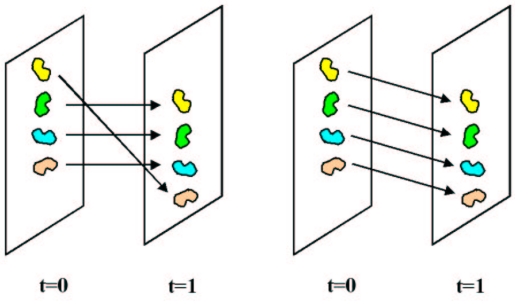

Figure 6: An example to illustrate that a linear function is not a good choice to penalize spatial distance.

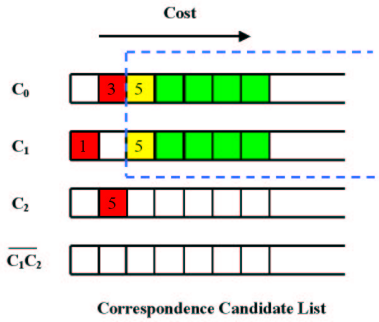

Figure 7: An example to illustrate that the current minimal cost can be utilized to prune the search space.

more similar between their volumes and shapes, the smaller the cost is. Therefore, the cost should be inversely proportional to spatial closeness, volume similarity and shape similarity.

Another principle in choosing the cost function is that it should grow quicker than a linear function to penalize spatial distance, volume dissimilarity and shape dissimilarity. Figure 6 illustrates why the cost function should penalize spatial distance more quickly than a linear function. Isosurfaces at both frames contains four components with the same volume and shape. Thus only the spatial distance between the components influences the matching cost. Assume the center point distance between successive components is 10. If the cost function is linear, the total matching cost illustrated at the left figure will be 40, which is the same as that at the right figure. Thus, a linear function can not distinguish the wrong result (left figure) from our desired result (right figure). However, if the function grows faster than a linear function, for example a quadric function, the total matching cost illustrated at the left figure will be 1600, while that at the right figure is only 400. This is exactly what is desired. Similarly this also applies to volume and shape. A fast growing cost function will also benefit the pruning of the optimization search space, which is introduced next.

3.2.3 Optimization with Fast Prune of Search Space

Now that every single and compound component has a list of correspondence candidates and the respective matching costs, the best global match can be found by an optimization stage which attempts to minimize the overall cost. For each component (single or compound), its corresponding candidates are first sorted in an increasing order according to the cost value. The optimization algorithm will then find the best possible correspondence for each component.

The optimization stage is essentially a search algorithm. It uses a similar method as alpha pruning [18, 23] in the field of Artificial Intelligence. The goal of the optimization is to find a match result where all the isosurface components at both frames are matched and the overall cost is minimal. Each single and compound component has a list of correspondence candidates. The search space is composed of all combinations of single and compound components with their correspondence candidates. Therefore, the search space can be huge. If there are 20 single and compound components and each of them has 5 correspondence candidates, the search space will contain 5^{20} combinations. If each of them is evaluated, the algorithm will take forever to finish. Notice that some combinations are illegitimate. A combination is legitimate if every component at the current frame is included exactly once except disappearing candidates, and every component in the next frame is also included exactly once except creating candidates. For disappearing and creating candidates, they can be included 0 or 1 times.

The search process will start with the best correspondence candidate for each single and compound component. If an illegitimate combination is encountered, it will be skipped. When a legitimate combination is found, if its cost is smaller than that of any previously encountered legitimate combination, the cost will be recorded as the current minimal cost. We can prune the search space based on the current minimal cost as explained in the following.

Remember that the goal of the global optimization is to find a legitimate combination with minimal cost, therefore, any combination whose cost is larger than the current minimal cost, can not be the result of the optimization. Due to the fact that the correspondence candidates of every component are sorted in an increasing order according to the costs, it is very likely that a subcombination that will be tried later, which we call *SC*, will have a cost greater than the current minimal cost. When such a subcombination is identified, any combination that contains *SC* as a subset will surely has a cost larger than the current minimal cost. These combinations can not be the global optimization result and hence can be quickly pruned out of the search space.

In fact, we can prune the search space even more when such an *SC* is found. Assume *SC* contains correspondence candidates of multiple components in the first frame. Since the correspondence candidates of every component are sorted in an increasing order according to their costs, any subcombinations that are composed of correspondence candidates which lie behind those candidates in *SC* in the respective candidate list will for sure have larger costs. Therefore, any combination containing any of these subcombinations can be pruned away from the search space.

As an example, in figure 7, an isosurface composed of three components is tracked. The global optimization is performed among correspondence candidates for single component C_0, C_1, C_2 and compound component (merging candidate) $\overline{C_1C_2}$. The red blocks, the 2nd correspondence candidate for C_0 and C_2 and the 1st correspondence candidate for C_1, represent a legitimate combination with cost 9. A later tried subcombination *SC*, which contains the third correspondence candidate for C_0 and C_1 (the two yellow blocks in the figure), have a cost 10, which is greater than the current minimal cost (9). Then any combination containing *SC* will have larger cost and can be pruned out of the search space. Furthermore, any combination of the candidates from the single component C_0 and C_1 which lies in the green area (behind the correspondence candidate in *SC* in the corresponding component) will have cost larger than the current minimal cost (9). Thus, any combination containing those subcombinations can also be pruned away. Therefore, once such a subcombination *SC* is encountered, a large number of candidate combinations, which in the example are any combination containing correspondence candidate in the dotted region in C_0 and C_1, can be pruned out of the search space.

Once a legitimate combination is encountered, its cost can be used to prune the search space. Actually when an illegitimate sub-

combination is encountered, it may also be used to prune the search space. If the reason that causes a subcombination to be illegitimate is that it includes some component more than once, then any combination containing this subcombination will also include that component more than once. Therefore any of these combinations is illegitimate and can also be pruned away. Overall, these pruning operations make the search process very efficient.

4 Precomputation of the Correspondence Relationship

Given two isosurfaces each of which consists of multiple components, the optimization algorithm presented above can find the best global match. When the isovalue is changed, the optimization algorithm needs to be executed again to identify the new correspondence relationship. In fact, it is possible to compute a finite number of isosurface correspondences to depict the evolution history of isosurfaces generated by arbitrary isovalues. In the following, we describe our approach.

We observed that the center point position, volume and shape often exhibit great coherence for the isosurfaces of nearby isovalues. In addition, an isosurface changes its number of components only at the critical isovalues. If we combine the critical isovalues from two frames into a single sorted list, the number of isosurface components remains the same for both frames within adjacent critical isovalues in that list. The value interval between adjacent critical isovalues is usually very small. Therefore, the isosurface exhibits great coherence within adjacent critical isovalues. In case that a large interval is encountered, a subdivision can always be applied if desired. With such a sorted list of critical isovalues, the correspondence relationship within each interval will most likely remain the same. To take advantage of this property, we can choose a representative isovalue, such as the midvalue in the interval, and then extract the correspondence relationship of the isosurface for the interval in a preprocessing stage.

We can use this approach to precompute the correspondence relationship for any isovalue interval in which the user might be interested. It is noteworthy, however, that sometimes there can be a large number of critical isovalues across the whole isovalue range and the interval between critical isovalues becomes really tiny. For example, for the vorticity data set we used for testing, frame 1 contains more than 12K critical isovalues from the minimum value 0.007009 to the maximum value 12.139389. When this happens, precomputing the correspondence for an isovalue interval is time consuming. One way to remedy this is to remove unnecessary critical isovalues.

The method by [6] can be used to simplify the topology of 2D scalar fields by progressively canceling critical points in pairs. In our algorithm, we use a simple but effective method to remove unnecessary critical isovalues. We observe that some isosurface components remain too small to be noticeable after it appears, till it disappears or merges with other components. These components can be caused by noise and often is not of interest to the user. Our method starts with the contour tree [20, 15, 8, 30] of a 3D scalar field. If the number of triangles of an isosurface component at any isovalue within a superarc is always smaller than a user-specified threshold, the superarc can be removed from the contour tree. After all the thresholded superarcs are removed, many superarcs can be combined together, since some critical points are not critical any more. Even after the simplification, the critical points are still dense enough that the isosurface component will not change much within any adjacent critical isovalues. After removing these unnecessary critical isovalues, the number of critical isovalues within a user-specified value range is reduced. Thus, the correspondence precomputation time will be reduced accordingly.

To stay conservative, a verification stage will be performed to see if the matching cost for each component is approximately the same as that at the midvalue. In the verification stage, because the skeleton extraction is quite expensive, we only use the volume and center point position to get the cost to see if it is approximately the same as that in the midvalue, which is also calculated only using volume and center point position. If the values are close, the correspondence at the midvalue can be used as the correspondence for this isovalue. In case that they are quite different, we can still use the global optimization process to extract the correspondence at this isovalue. In this way, the correspondence determination time will be reduced significantly for most cases.

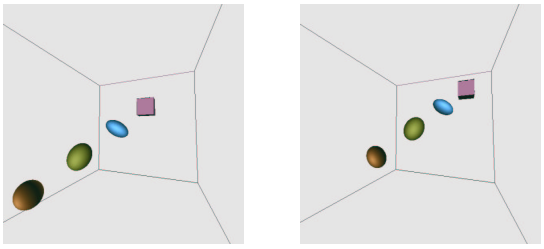

Figure 8: Four fast moving components are tracked, which is similar to the case in figure 1.

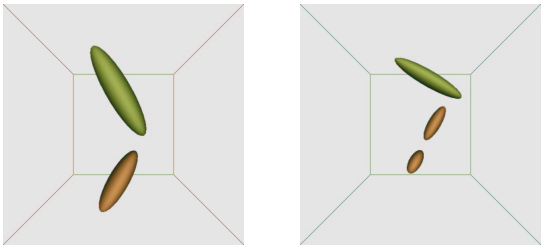

Figure 9: Two components are tracked, which is similar to the case in figure 2.

5 Results

We have tested our algorithm using a $128 \times 128 \times 128$ vorticity magnitude data set with 99 time steps and our synthesized data sets. The machine we used was a dual processor Pentium IV 2.4GHz with 2GB main memory.

Figure 8 shows the tracking result from a synthetic data which contains multiple fast moving components, similar to the case in figure 1. Components which were determined as correspondence in two frames are rendered with the same color. Since the global configuration of all the components is considered, our algorithm can track every component correctly while the local tracking techniques will give some nonglobally optimized results. Another example similar to the case in figure 2 is shown in figure 9. Based on the global optimization result, our algorithm concluded that the larger component in the first frame continues while the smaller one splits, even though the larger component has the maximum overlap with the first and second components from the top in the next frame and also has the most similar attributes (center point position and volume).

Figure 10 shows an example of using our global optimization algorithm to track an isosurface with multiple components. Six snapshots are shown. The time-varying isosurface was extracted with an isovalue of 6.8 from the vorticity data set. To show that our algorithm can successfully track the isosurfaces even when the data is not densely sampled in time, we use only the data from every three time steps of the original data set to perform the tests. There

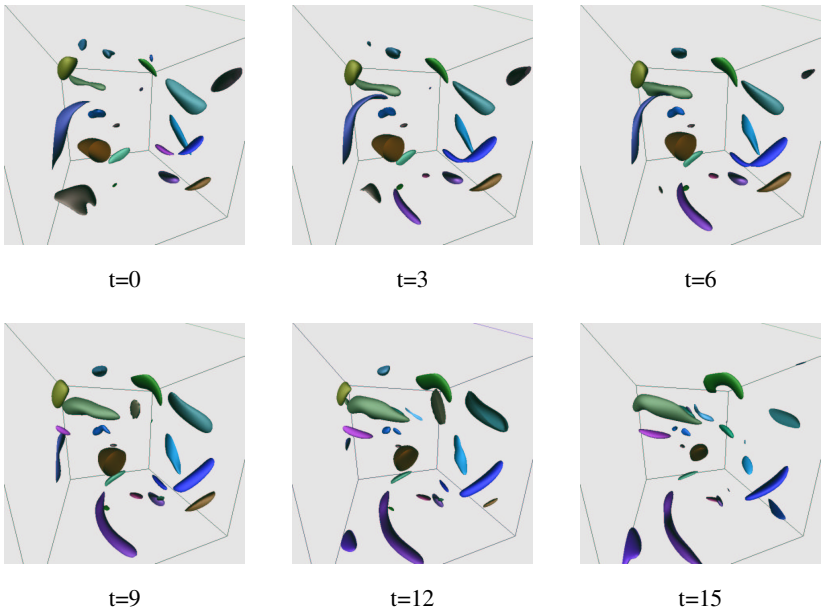

Figure 10: A time varying isosurface with multiple components are tracked and the tracking information determines the coloring of the components. The data from every three time steps is used. Six snapshots are shown.

Frames	0-3	3-6	6-9	9-12	12-15
Total time	1.916	2.065	2.079	2.202	2.219
Isosurface extraction time	0.169	0.179	0.184	0.169	0.157
Attribute calculation time (Skeleton extraction time)	1.725 (1.62)	1.864 (1.75)	1.874 (1.76)	2.011 (1.90)	2.033 (1.89)
Optimization time	0.022	0.022	0.021	0.022	0.029

Table 1: The timing (in seconds) for the case in figure 10 is shown.

Frames	0-3	3-6	6-9	9-12	12-15
Number of critical isovalues	31	28	29	36	46
Precomputation time	139.1	111.2	129.8	158.5	197.8

Table 2: The correspondence relationship within the critical isovalue interval 6.5-7.5 is precomputed for the case in figure 10. The number of critical isovalue within the range and the timing (in seconds) are shown. Notice that the average tracking time is larger that in table 1, because when the isovalue changes, all the attributes of the isosurfaces from both time steps need to be recomputed. However, when the time step changes, the attributes at the new source frame need not to be recomputed, since their values are already computed previously.

Frames	0-3	3-6	6-9	9-12	12-15
Total time	0.241	0.251	0.259	0.240	0.239
Isosurface extraction time	0.170	0.178	0.185	0.169	0.158
Verification time	0.071	0.073	0.074	0.071	0.081

Table 3: The tracking time (in seconds) using the precomputed correspondence table is shown. The tracking time is reduced significantly since only simple table lookup operation and verification are involved.

are quite a number of cases where corresponding components have quite different attributes or they have no overlap. In figure 10, each component is colored in such a way that it has the same color as the component it is evolved from. When merge happens where a component is evolved from multiple components, the current component will get the color of the previous dominant component. The dominance is defined by volume in our example, i.e., the current component would follow the color of the previous largest component. In the case that a component is created, a new color will be assigned to it. Our global optimization gives good result and it is the same as the result when the full data set is used.

The tracking speed of our algorithm and the breakdown of the execution time in different algorithm stages for the case in figure 10 are shown in table 1. In this test, all computations were done on the fly and there was no precomputation of the correspondence relationship involved. Our tracking algorithm includes three stages: isosurface extraction, attribute calculation and global optimization. In the first stage, isosurfaces from both time steps are extracted. Attributes such as volume, center point position and skeleton are then calculated in the second stage, and additionally compound components are identified. In the third stage, correspondence candidates for each single and compound component are identified and then the global optimization with a fast pruning of the search space is performed. From the results, we can see that the global optimization stage only took a small fraction of the total tracking time. A majority of the computation is spent on the extraction of skeletons. The overhead of tracking includes attribute calculation time and global optimization time. These overhead can be mitigated when the correspondence relationships are precomputed, as described in section 4. The performance of this precomputation will be studied next.

The overhead to calculate attributes and perform global optimization takes the major part of the overall tracking time. For a user

who only needs to track isosurfaces from a small number of predefined isovalues, this may be tolerable. However, for a user who wants to study the time-varying scalar field by frequently changing the isovalue and visualizing the tracking results, a precomputation of the correspondence relationship for the time-varying isosurface across a range of values will be beneficial. When the correspondence relationship is precomputed, tracking can be achieved by simple table lookup and verification operations. We precomputed the correspondence relationship for the case in figure 10 within the isovalue interval 6.5 and 7.5. Before the precomputation, all unnecessary critical isovalues of each frame were first removed, which correspond to the superarcs in the contour tree which will generate less than 50 triangles. When this was done, nearly 85% of the original critical isovalues were classified as redundant and removed. The precomputation time is shown in table 2. The run-time tracking speed when the precomputed correspondence was utilized for the case in figure 10 is shown in table 3. Note in this case, the tracking time is significantly reduced compared with the performance in table 1. In our test, all the verifications return positive results. A number of other runs for testing different isovalues were also performed and consistent results were observed. In case that the verification returns negative result, the globally optimization can still be performed.

6 Conclusions and Future Work

In this paper, we propose a global optimization algorithm to track time-varying isosurfaces. The algorithm will first identify a list of correspondence candidates for each single and compound component. A cost to match a component to each of its correspondence candidates is also calculated. A global optimization algorithm is then performed to minimize the overall cost to match the source isosurface to the target. This global optimization technique ensures that every component will match to its best possible component in the next frame. We also propose a method to precompute the correspondence relationship within any isovalue range. With the global optimization algorithm and the precomputed correspondence relationship, the time-varying isosurface is tracked in a more accurate and efficient manner.

In our future work, we will incorporate ideas from other research to perform the prediction of correspondence candidates. For example, in [22], the path continuity characteristic is used extensively. If a path of a component is found, it can be used to predict the center point position and volume of the component in the next frame. We believe that incorporating this idea into our framework will be very beneficial. For example, the correspondence candidates for a component can be found based on where it is predicted to evolve. In the evaluation of the matching cost, the predicted volume and center point position can be used instead of the current volume and center point position. More methods on shape description and matching will also be investigated.

References

[1] J.K. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images – a review. *Proceedings of the IEEE*, 76(8):917–935, 1988.

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

[3] C.L. Bajaj, V. Pascucci, and D.R. Schikore. The contour spectrum. In *Proceedings of Visualization 1997*, pages 167–173, 1997.

[4] D.H. Ballard. *Computer Vision*. Prentice-Hall,Inc, Englewood, New Jersey, 1982.

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

[6] P.T. Bremer, H. Edelsbrunner, B. Hamann, and V. Pascucci. A multiresolution data structure for two-dimensional morse-smale functions. In *Proceedings of Visualization 2003*, pages 139–146, 2003.

[7] I. Carlbom, I. Chakravarty, and W. Hsu. Integrating computer graphics, computer vision, and image processing in scientific applications. *Computer Graphics*, 26(1):8–17, 1992.

[8] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2), 2003.

[9] J. Chen, D. Silver, and L. Jiang. The feature tree: Visualizing feature tracking in distributed amr datasets. In *Proceedings of IEEE symposium on Parallel and Large-Data Visualization and Graphics 2003*, pages 103–110, 2003.

[10] M. Hilaga, Y. Shinagawa, T. Komura, and T. Kunni. Topology matching for fully automatic similarity estimation of 3d shapes. In *Proceedings of SIGGRAPH 2001*, pages 203–212, 2001.

[11] G. Ji and H-W. Shen. Efficient isosurface tracking using precomputed correspondence table. In *Joint Eurographics - IEEE TCVG Symposium on Visualization 2004*, 2004.

[12] G. Ji, H-W. Shen, and R. Wenger. Volume tracking using higher dimensional isosurfacing. In *Proceedings of Visualization 2003*, pages 209–216, 2003.

[13] P.P. Jonker and A.M. Vossepoel. On skeletonization algorithms for 2, 3,.. n dimensional images. In *Proceedings of Shape, Structure and Pattern Recognition 1994*, pages 71–80, 1994.

[14] W. Koegler. Case study: Applications of feature tracking to analysis of autoignition simulation data. In *Proceedings of IEEE Visulazation 2001*, pages 461–464, 2001.

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

[16] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.

[17] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[18] N.J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufamnn Publishers, Inc. San Francisco, California, 1998.

[19] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Matching 3d models with shape distributions. In *Proceedings of the International Conference on Shape Modeling and Applications 2001*, pages 154–166, 2001.

[20] V. Pascucci and K. Cole-McLaughlin. Efficient computation of the topology of level sets. In *Proceedings of Visualization 2002*, pages 187–194, 2002.

[21] F. Reinders, M. Jacobson, F. Post, and E. Association. Skeleton graph generation for feature shape description. In *Proceedings of Data Visualization 2000*, pages 73–82, 2000.

[22] F. Reinders, F.H. Post, and H.J.W. Spoelder. Visualization of time-dependent data using feature tracking and event detection. *The Visual Computer*, 17(1):55–71, 2001.

[23] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall,Inc, Upper Saddle River, New Jersey, 2003.

[24] R. Samtaney, D. Silver, N. Zabusky, and J. Cao. Visualizing features and tracking their evolution. *IEEE Computer*, 27(7):20–27, 1994.

[25] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition 1994*, pages 593–600, 1994.

[26] D. Silver. Object-oriented visualization. *IEEE Computer Graphics and Applications*, 15(3), 1995.

[27] D. Silver and X. Wang. Volume tracking. In *Proceedings of Visualization 1996*, pages 157–164, 1996.

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

[29] D. Silver and X. Wang. Tracking scalar features in unstructured datasets. In *Proceedings of Visualization 1998*, pages 79–86, 1998.

[30] S.P. Tarasov and M.N. Vyalyi. Construction of contour trees in 3d in o(nlog n) steps. In *Proceedings 14th Annual ACM Symposium on Computational Geometry 1998*, pages 68–75, 1998.