Feature Tracking Using Earth Mover's Distance and Global Optimization

Guangfeng Ji*
The Ohio State University

Han-Wei Shen†
The Ohio State University

Abstract

Feature tracking plays an important role in understanding time-varying data sets since it allows scientists to focus on regions of interest and track their evolution and interaction over time. For complex data sets, previous feature tracking methods cannot guarantee the globally best matching results. This is because previous algorithms are primarily local tracking techniques, where individual features are matched independently with local search criteria. In this paper, we propose a novel global tracking technique to track features. The globally best match is the match from the source feature set to the destination feature set with a global minimal matching cost. To amend the problems of using volume-overlapping and aggregate-attribute criteria as in the previous methods, we propose to use the Earth Mover's Distance (EMD) to evaluate the matching cost. EMD takes into account the spatial distribution of the features and provides a more accurate and robust cost metric than those used in previous methods. The global cost evaluation implicitly defines a search tree, for which we use an efficient branch-and-bound approach to find the match with the global minimal cost. With the EMD metric and the branch-and-bound algorithm, the global matching algorithm tracks features in a more accurate and efficient manner.

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 [10], 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. For time-varying data, an effective visualization tool should also compute and track features over time in an accurate and efficient manner.

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

To address the needs, researchers have developed various feature tracking techniques to study time-varying scalar fields. Silver and Wang [18, 19, 20, 21] proposed to match features that have a large degree of volume overlap. Samtaney *et al.*[16] and Reinders *et al.*[12] match features based on some representative aggregate attributes such as positions, volumes, and masses. Ji and Shen [9, 8] tracked time-varying isosurfaces and interval volumes using higher-dimensional geometry. While those methods can successfully track time-varying features in many cases, they are generally *local tracking techniques*, where the tracking of each feature is performed independently with various local search schemes. The local tracking techniques do not guarantee to produce the globally best matching results in many cases, especially when the scene is crowded with many small features and there exist several alternatives to match those features.

In this paper, we propose a novel global tracking algorithm to address the aforementioned problem. All the features from two adjacent time steps are considered together in the global tracking algorithm. We first devise a method to measure the matching cost between a source and a destination feature component. Then, based on the cost metric, our tracking algorithm finds the globally best match between the two feature sets that has the minimal overall cost. In order to obtain meaningful tracking results, it is essential to have a proper measure of the cost to match two features. The cost measure should take proper consideration of the important characteristics of the feature and gives accurate matching cost. It should be robust and always offers a meaningful cost to match features. Previous researchers have mainly used either the volume-overlapping or similar-attribute criterion to match features. The volume-overlapping criterion works well when the temporal sampling rate is high enough so that corresponding features have a large degree of overlap. However, the criterion fails to track small/thin and fast-moving features where corresponding features do not overlap. The similar-attribute criterion tracks features by using aggregate attributes such as centroid positions, volumes and masses. But important characteristics of the feature, such as its shape and orientation, are not considered properly. To amend the problems of these criteria, we propose to use the Earth Mover's Distance (EMD) to measure the cost to match two features. EMD represents each feature with the cells it occupies in space, and computes the minimal cost to transport one cell distribution into the other. EMD evaluates the matching cost by using the spatial occupancy information, which includes all the shape, orientation, position and scale attributes of the feature. The EMD measure also offers a meaningful cost to match non-overlapping features. Compared to the previous criteria, EMD provides a more accurate and robust cost measure. In addition to the EMD cost measure, we also propose a branch-and-bound method which enables efficient search of the global minimal cost. A tight estimation of the minimal cost further speeds up the search process. With the Earth Mover's Distance criterion and the branch-and-bound algorithm, features are tracked in a more accurate and efficient manner.

The organization of the paper is as follows. We first review previous work in section 2, and give an overview of the algorithm in section 3. In section 4, we present the EMD criterion to evaluate the feature matching cost. The branch-and-bound algorithm is introduced in section 5. Test results are presented in section 6 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 [3] used a predictor-corrector method to reconstruct and track vortex tubes from turbulent time-dependent flows. Arnaud *et al.*[1] tracked 2D cloud patterns and used area overlap to determine correspondence. The tracking methods basically fall into two categories: volume overlapping based methods [18, 19, 20, 21, 9, 8] and aggregate attributes based methods [16, 12]. Silver and Wang [18, 19, 20, 21] observed that corresponding features in adjacent time steps usually overlap when the temporal sampling rate of the underlying data is high. Based on the observation, correspondences between features in consecutive time steps are identified using a two-stage process including an overlap and a best matching test. In the overlap test, spatially overlapped features from consecutive time steps are identified and the number of intersecting nodes is also computed. 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.*[16] tracked features using their centroid positions, masses, volumes and circulations (in 2D). Each feature is matched to the feature(s) in the next time step whose centroid position is the closest to its centroid and the volume and mass are also within a prescribed tolerance. Reinders *et al.*[12] also calculated a set of attributes, such as centroid 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.

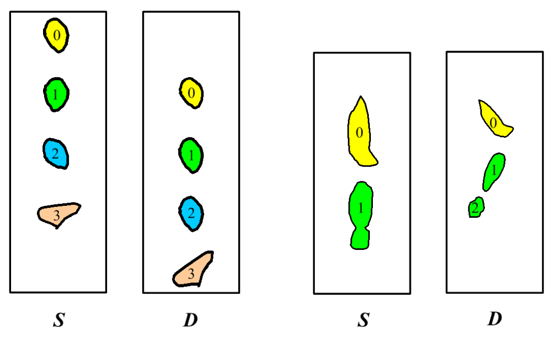
In [9], Ji and Shen tracked local isosurfaces and interval volumes features efficiently by using higher dimensional isosurfacing. The method is based on the volume-overlapping criterion. In [8], they further reported that the overlapping relationship between isosurfaces of two consecutive time steps can change only at critical isovalues in R^3 or R^4 and remains unchanged between any two adjacent critical isovalues. Therefore, the overlapping relationship between isosurfaces from any two consecutive time steps can be precomputed and stored in a correspondence lookup table. With this overlapping table, isosurface tracking can be achieved by simple table lookup and verification operations. In [22], Sohn and Bajaj built the topology of time-varying isosurfaces. The resulting time-varying contour tree can be used to track how isosurfaces evolve.

Chen *et al.*[4] extended the work by Silver and Wang [18, 19, 20, 21] 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 [2, 17]. 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 Algorithm Overview

Given two feature sets in the adjacent time steps with the source feature set $S = \{S_0, S_1, ...S_{n-1}\}$ and the destination feature set $D = \{D_0, D_1, ...D_{m-1}\}$, one way to establish the correspondence between S and D is to associate every feature in S with the feature(s) of its locally best match in D. The locally best match can be defined by either the volume-overlapping [18, 19, 20, 21, 9, 8] or the similar-attribute criterion [16, 12]. While these local tracking techniques are effective for various applications, there are many cases where the local matching techniques do not offer the globally best result (The term "globally" refers to the case that all the features from two adjacent time steps are considered together during feature matching). For example, in the example shown in figure 1(a), the locally best matching



(a) (b)

Figure 1. Two cases to illustrate that the local tracking techniques do not guarantee the globally best matching result.

feature for S_1 is D_0 because they have the maximal overlap, and their centroid positions and volumes are also the most similar. However, globally S_1 should correspond to D_1 because what happens is that the entire feature set moves downwards in figure 1(a). Similarly, in the example shown in figure 1(b), both features in S shrink as time evolves. If S_0 is the first to be matched, it will correspond to D_0 and D_1. This is because they have the maximal overlap with S_0, also the most similar centroid positions and volumes. However, globally only D_0 should be matched to S_0 . From these two examples, we can see that the local matching scheme does not always produce the globally best matching result.

To address the problem, we propose to use a global matching technique that takes into account the configuration of the features all together. We assume a cost is properly defined to match any source feature component to any destination feature component. The goal of the global matching is to find the minimal cost to match the whole source feature set S to the destination feature set D. The matching result with the minimal cost will be the globally best match between S and D with a properly-designed cost function. A feature component may undergo the following evolutionary events:

- Continuation: a single feature at t matches to another single feature at $t+1$.
- Amalgamation: a group of features at t match to a single feature at $t+1$.
- Bifurcation : a single feature at t matches to a group of features at $t+1$.
- Creation : an empty feature at t matches to a single feature at $t+1$.
- Dissipation : a single feature at t matches to an empty feature at $t+1$.

When a group of feature components are involved in an evolutionary event such as amalgamation and bifurcation, we call them a *compound component*. Notice that although the five evolutionary events above do not include matching between two compound components, the matching between two compound components can be split into combinations of the five evolutionary events. If we denote $Cost(A,B)$ as the minimal cost to match A to B, where $A=\{S_i,S_{i+1},\ldots S_{n-1}\}$ and $B\subseteq D$, the following recursive function holds for $Cost(A,B)$:

$$Cost(A,B)=\min_{\substack{TA\subseteq A,TB\subseteq B\\ S_i\in TA,D_j\in TB\\ |TA|\neq 1}}\left\{\begin{array}{l} Cost(S_i,TB)+Cost(A-S_i,B-TB)\\ Cost(TA,D_j)+Cost(A-TA,B-D_j)\end{array}\right. \tag{1}$$

where TA can be an empty or a compound component in A containing S_i, and TB can be an empty, single or compound component of B. This equation states that the minimal cost to match the source feature subset A to the destination feature subset B is the minimum of the following five scenarios: (We list the corresponding evolutionary event that S_i undergoes in each scenario)

- $TB=\emptyset$. In this case, S_i disappears;
- TB is a single component. In this case,S_i continues;
- TB is a compound component. In this case, S_i splits;
- $TA=\emptyset$. In this case, a new component (D_j) appears.
- TA is a compound component containing S_i. In this case, S_i merges with some other components and becomes D_j at the next time step.

Notice that we do not allow TA to contain just only one component, since otherwise the continuation event will be considered twice in equation 1.

There are two key issues in the formulation and search for the globally best match. First, we need to have a proper cost function to match any source and destination feature component. The evaluation of matching cost should accurately reflect the unlikelihood of matching a source component to a destination component. Furthermore, the definition should be robust enough to take all the cases into account. Second, equation 1 actually defines a recursive search tree structure. The root node of the tree is $Cost(S,D)$, We need to have an efficient way to calculate $Cost(S,D)$ and find the globally best match. In the following sections, we first describe our definition of the matching cost,and then describe an efficient branch-and-bound algorithm to calculate $Cost(S,D)$.

4 Using Earth Mover's Distance as Matching Cost

It is essential to have a proper cost function when matching the source and destination feature components.

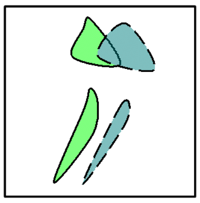


Figure 2. A thin feature is moving fast. The volume-overlapping criterion will not give a meaningful matching cost in this case.

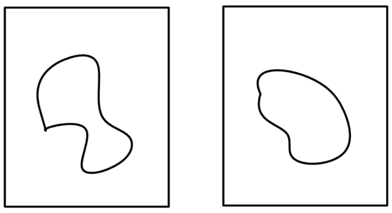


Figure 3. A figure to illustrate why EMD is a better metric than the aggregate-attribute approach.

The matching cost should take proper consideration of the important characteristics of the feature components, such as their shapes, orientations, scales, and positions. The cost definition should also be robust in the sense that it should always give a meaningful cost. Previous researchers have mainly used either the volume-overlapping or similar-attribute criterion to match features. If the volume-overlapping criterion is used, the matching cost can be defined as a function which is inversely proportional to the overlapping degree of the source and destination feature components. However, the volume-overlapping criterion cannot handle the case where two corresponding features do not overlap, for example, when a small and/or thin feature moves quickly so that there is no overlap between their volumes in adjacent time steps (see figure 2). Such a definition fails to give a meaningful cost in this case. The similar-attribute criterion describes a feature by its aggregate attributes, such as the centroid position, volume and mass. Two features correspond when they have close centroid positions and similar volumes and masses. If the similar-attribute criterion is used, the definition of the matching cost would be a function which is proportional to the centroid distance, and the volume and mass difference. But these aggregate attributes can over-simplify the feature description. Specifically, the aggregate attributes do not take proper consideration of the shape and orientation of the feature, and this can lead to inaccurate matching cost measures. For example, in figure 3, the two features have the same centroid positions, volumes (areas) and messes, but different shapes. The aggregate-attribute approach gives an inaccurate matching cost, since the shape information is not properly considered.

Aiming to address the above problems, we propose to use the Earth Mover's Distance (EMD) metric [14, 15] to measure the matching cost between any source and destination feature components. In the EMD computation, each feature is represented by the set of cells it occupies in space, and EMD calculates the minimal amount of work required to transport one set of cells into the other. EMD takes a unified approach to measure the matching cost by taking into account the feature's spatial occupancy information, which includes all the shape, orientation, position and scale attributes of a feature (Specifically, even when the features do not overlap). Therefore, it is more robust than the volume-overlapping criterion and considers more attributes properly than the aggregate-attribute criterion. Furthermore, the aggregate-attribute criterion does not answer the question of how to unify the difference from different attributes together during the calculation of matching cost, for example, how to appropriately integrate the centroid distance and the volume ratio together into the cost definition. In the following sections we first give a brief review of the Earth Mover's Distance and explain how it is used in our feature tracking algorithm.

4.1 Earth Mover's Distance

Distribution information is often used in many areas to represent features. For example, in computer vision, images are often represented by a one-dimensional distribution of image intensities or three-dimensional distribution of image colors. In scientific visualization, a volumetric feature can be represented by the set of cells it occupies in space. To measure the difference (or distance) between two distributions, Rubner *et al.*[14, 15] propose the Earth Mover's Distance (EMD). EMD reflects the minimal amount of work that must be performed to transport one distribution into the other. Intuitively, we can treat one distribution as a mass of earth properly spread in space and the other as a collection of holes in the same space. EMD measures the minimal amount of work needed to fill the holes with earth. We can always assume there is enough earth to fill the holes, since otherwise we can switch what we call earth and hole. Here, a unit of work corresponds to transporting a unit of earth by a unit of distance. Formally, EMD assumes that the ground distance function, which measures the cost to transport a unit of the source distribution into a unit of the destination distribution, is given. EMD then lifts the distance between the individual units to the whole distribution.

In our tracking algorithm, we represent a feature by the cells it occupies in space. For example, isosurfaces [11] extracted from a volume data can be represented by all the cells (isocells) that contain the isosurface patches. Interval volumes [6, 5] can be represented by all the cells lying within the user-specified value range. We use EMD to measure the distance between two features.

In our algorithm, the Euclidean distance between cells in space is used as the ground distance. EMD naturally lifts the distance between two individual cells to define the distance between two features. When two features are close in space with similar shapes and orientations, the EMD value between these two features will be small since they have very similar spatial occupancies. In contrast, if two features are far away and/or have different shapes and/or orientations, the EMD value between these two features will be large, since their spatial occupancies are different. In our implementation, we penalize more on matching two features that are far away. We used the following nonlinear ground distance function GD to achieve this:

$$GD(C_i, C_j) = EuclideanDistance(C_i, C_i)^P \tag{2}$$

where C_i and C_j are two cells, and $P > 1$ is used to penalize matching features that are far away.

Notice that EMD allows partial match. This means if we match a feature distribution F to another distribution which is a subset of F, the EMD value will be 0. The property will be useful during the identification of compound component candidates (this will be explained in section 5.4). However, when calculating EMD between any source and destination feature components, partial match can cause some problem. For example, the EMD value between a feature and any of its subset which has a smaller feature size will be 0 if partial match is allowed, and this is undesired. Moreover, the matching cost definition should penalize matching two features with different scales. To remedy this problem, whenever we calculate the EMD value between any two features, we will make sure the two features have the same capacity. To achieve this, we first assign uniform capacity to each cell of a feature. When matching two features, we scale the capacity of all the cells of each feature uniformly so that the two features to be matched will have the same capacity. By the scaling operation, we avoid the partial matching problem and EMD will penalize matching features with different scales. Actually EMD is a true metric when comparing distributions with the same capacity.

4.2 Efficient EMD Computation

Computing EMD is based on a well-known linear programming problem, the transportation problem. The transportation problem defines a bipartite network with S as a set of suppliers and C as a set of consumers. $X_i (0 \leq i \leq n)$ is the total supply of supplier i and $Y_j (0 \leq j \leq m)$ is the total capacity of the consumer j. T_{ij} is the cost to ship a unit of supply from supplier i to consumer j. The transportation problem is to find a set of flows F_{ij} that minimize the overall cost:

$$\sum_{(0 \leq i \leq n)} \sum_{(0 \leq j \leq m)} (T_{ij} \times F_{ij})$$

subject to the following constraints:

$$F_{ij} \geq 0 \quad (0 \leq i \leq n \text{ and } 0 \leq j \leq m)$$

$$\sum_{(0 \leq i \leq n)} F_{ij} = Y_j \quad (0 \leq i \leq n)$$

$$\sum_{(0 \leq j \leq m)} F_{ij} \leq X_i \quad (0 \leq j \leq m)$$

The constraints force the consumers to fill up all of their capacities and each supplier can not transport more than its supply. We assume the total demand does not exceed the total supply, as we can switch what we call supplier and consumer if necessary. Efficient algorithms are available to solve the transportation problem. In our implementation, we used the transportation-simplex method which is a streamlined algorithm based on the simplex method [7]. Actually we can further improve the efficiency of the EMD computation. During the EMD computation between two distributions, it is unnecessary to use the whole distribution. Researchers have used a so-called *signature* [14, 15] of the distribution to further speed up the EMD calculation.

Now we describe a method to simplify the feature distribution information. This is very similar to the vertex clustering algorithm to simplify meshes [13]. To accelerate the EMD computation, we put the features into a coarser grid by simply merging n^3 cells into a single large cell (A 2D example is shown in figure 4). This large cell may contain many feature cells at the original resolution. We use the average location of the feature cells as the location in the merged cell, and the capacity of the merged cell equals to the total capacity of the feature cells in the original n^3 cells. The process can be efficiently done by a linear scan of all the feature cells. After the simplification, the size of the feature distribution is reduced significantly. However, the simplified feature distribution maintains the same signature with the original one, that is, they show the same spatial

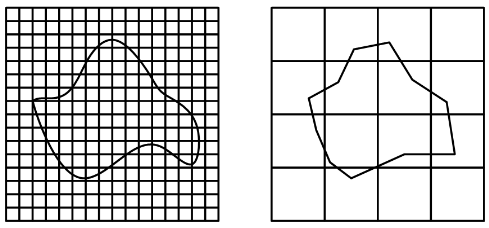


Figure 4. A feature (in a 16×16 grid)) and its simplification (in a 4×4 grid)).

distribution pattern. So the EMD values using the simplified features will be just as accurate.

EMD provides an accurate matching cost between any source and destination feature components. Another important issue is how to calculate $Cost(S,D)$ efficiently. In the next section, we will explain how we achieve this by using a branch-and bound search algorithm.

5 Global Optimization Based on Branch-and-Bound

5.1 The Search Tree

Equation 1 implicitly defines a search tree structure. The root of the tree is $Cost(S,D)$. We assume that the feature components are considered sequentially from S_0 to S_{n-1}. Following equation 1, at the first level of the search tree, each node represents one of the following matching possibilities that involve the first feature S_0:

- Match S_0 to an individual component in D.
- Match S_0 to a compound component in D.
- Match S_0 to an empty component.
- Match any compound component containing S_0 to an individual component in D.

In the example shown in figure 5 where we want to find the correspondence between three source features and three destination features, we will consider the following possible matches at the first level of the tree: $S_0 \to D_0$, $S_0 \to D_1$, $S_0 \to D_2$, $S_0 \to D_{12}$, $S_0 \to \emptyset$, $S_{01} \to D_0$, $S_{01} \to D_1$, and $S_{01} \to D_2$. For simplicity, we limit the possible compound components to only S_{01} and D_{12} (A method to identify compound component candidates will be introduced in section 5.4). From the first level of the tree, following each branch we will consider matching the first feature in the remainder of the source features at the second level. Similarly to the first level, all the four combinations listed above will be considered as long as the feature has not been covered in the ancestor nodes. Using the same example, when following the branch of $S_0 \to D_0$, the following matches will be considered at the second level: $S_1 \to D_1$, $S_1 \to D_2$, $S_1 \to D_{12}$, $S_1 \to \emptyset$. Notice that all matches involving the compound component S_{01} are excluded following this branch since S_0 has already been matched in the parent node. Any matches to D_0 are excluded for the same reason. This process will repeat until all the source feature components have been matched. Note that there might be still some destination features remaining with no correspondence. We then match an empty component to each of the remaining destination features and add these to the search tree. We can see that the search tree considers all the possible match combinations. The match with the minimal overall cost, which defines a

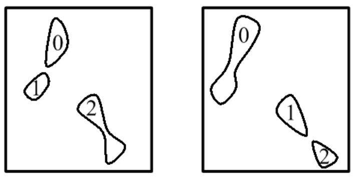


Figure 5. An example of two feature sets to be matched.

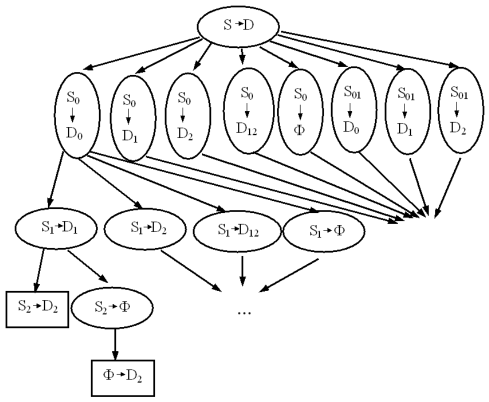


Figure 6. Part of the search tree for the case in figure 5.

unique path from the root to one of the leaf nodes, provides the globally best match for each source feature component.

If we follow the previous description to build the tree, the size of the tree can be huge. Accordingly it will take a long time to find the minimal $Cost(S,D)$. In the next two sections, we introduce methods to reduce the size and accordingly the search time of the tree. We first present a branch-and-bound method that is able to significantly reduce the size of the search tree. Then a way to get a tight estimation of the minimal cost is introduced to further speed up the process. We also describe a method to identify compound component candidates.

5.2 Branch and Bound

The search tree contains all the possible matches between the source and destination feature sets. At each branch of the tree, we consider all the matches which correspond to all the evolutionary events that can happen to the first feature in the current source feature set. We calculate the EMD value for each match and sort the matches based on the EMD value in an ascending order. The tree is traversed using a depth-first method, with the node that has the smallest EMD traversed first at each level of the tree.

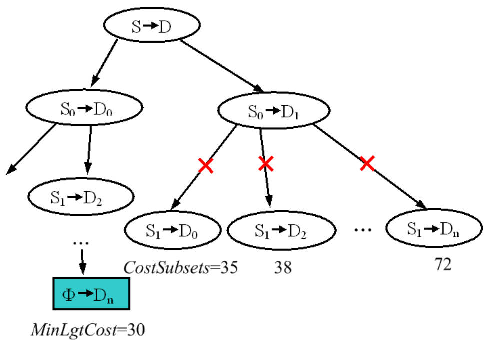


Figure 7. An example to illustrate how the branch-and-bound operation is applied to enable efficient tree search.

This enables a branch-and-bound operation, which can significantly reduce the size of the search tree.

During the depth-first traversal of the tree, we keep the minimal cost of all the legitimate matches (*MinLgtCost*) that have been found so far. A match is legitimate if and only if every component in the source and destination feature sets occurs in the match exactly once. Whenever a legitimate match is reached, we use its cost to update *MinLgtCost* if the cost is smaller. This *MinLgtCost* can be used to prune the tree in the following way. As the tree is being traversed, at each node we evaluate the total cost of the path from the root to the current node, and store it as *CostSubsets*. *CostSubsets* is the overall cost to match a subset of the source features and a subset of the destination features. If *CostSubsets* is smaller than *MinLgtCost*, the depth-first traversal continues. Otherwise the entire branch following the current node can be pruned. This is because if the cost to match only the subsets of the features (*CostSubsets*) is already larger than *MinLgtCost*, all the matches following the current node will certainly have matching cost larger than *MinLgtCost*, and hence none of these matches can be optimal.

Figure 7 shows an example of this branch-and-bound operation. The search process currently reaches the node $S_1 \to D_0$ and the $MinLgtCost = 30$. Suppose the matching cost of the path from the root to the current node, which is a summation of the costs $S_0 \to D_1$ and $S_1 \to D_0$, is 35. If the search process keeps traversing down the node $S_1 \to D_0$, all the legitimate matches found along the branch will have cost greater than 35, and of course greater than *MinLgtCost*, which is currently 30. Therefore, there is no need to traverse down this branch. The pruning operation can be applied to remove this branch.

In fact, whenever a node with its current overall cost (*CostSubsets*) greater than *MinLgtCost* is found out, an even larger portion of the tree can be pruned away in addition to the branch following the node. Recall that for each source feature, we have sorted all of its matching candidates based on their EMD values in an ascending order. Thus, at the current level of the tree, there is no need to try any remaining matching candidates of the current source feature, since the overall cost of the path from the root to any of those nodes will be larger than *CostSubsets*, and hence larger than *MinLgtCost*. An example is also illustrated in figure 7. The branch at the node $S_1 \to D_0$ has already been cut away. We know that the matching candidates of S_1 have been sorted in the EMD-value ascending order. Therefore, for any matching candidates D_i of S_1 that appear behind D_0, the matching cost of $S_1 \to D_i$ is greater than that of $S_1 \to D_0$, therefore, the cost of the path from the root to any of these matching candidates is greater than 35. None of the legitimate matches along these branches can have cost smaller than *MinLgtCost* (30). Therefore, all these branches can be pruned right away.

5.3 Further Speedup by a Good Estimation of the Minimal Cost

Branch-and-bound is a very powerful operation. As soon as a good evaluation of the current minimal cost (*MinLgtCost*) is reached, it can be used to cut away the search tree efficiently. When the global optimization is performed by the branch-and-bound operation, *MinLgtCost* of all the legitimate matches found so far keeps decreasing until a global minimum is reached, which defines the globally best match. We can initialize *MinLgtCost* to an arbitrary large value and let the tree pruning process to find the global minimum. However, a tighter bound of *MinLgtCost* can accelerate the tree pruning process significantly. If *MinLgtCost* starts with a very large value, at the beginning stage of tree traversal, very few pruning operations can be applied. In fact, it may take many runs to reach a reasonable *MinLgtCost*. However, a tighter *MinLgtCost* can accelerate the tree pruning process from the very beginning and speed up the whole process.

We find a tighter bound of *MinLgtCost* using a greedy method. First among all the matches between any source and destination feature components, we find the match with the smallest EMD. This may be a match between two single components, between single and compound components, or between single and empty components. After the match is discovered, we remove the features associated with this match from the feature space. Then the above process repeats recursively in the remainder of the feature space. When at some point either the source or destination feature set becomes empty, we will match an empty component to those remaining unmatched components. Although the result of this greedy process is not the global minimum in most cases, it is a good estimation of *MinLgtCost*. Using this estimation, the tree pruning operation can be applied from the very beginning of the tree buildup process. The whole search process therefore can be substantially accelerated using this good estimate of the minimal cost.

5.4 Using EMD to Identify Compound Component Candidates

To identify all the evolutionary events, at each branch of the search tree, we need to consider matches between single and empty components (for creation and dissipation), between two single components (for continuation), and between single and compound components (for amalgamation and bifurcation). For better efficiency, we need to have a good way to identify compound component candidates, rather than trying every possible combination of the feature components.

Our method to identify compound component candidates is based on the EMD's property that EMD allows partial matching. Suppose feature O_i and O_j form a compound component and they merge to feature O_k in the next time step. If we represent O_i, O_j and O_k by the cells they occupy in space and assign the weight of each individual cell to be 1, the EMD value between feature O_i and O_k (EMD_{ik}) and that between feature O_j and O_k (EMD_{jk}) should be small compared to the EMD value between other components and O_k. In addition, the summation of EMD_{ik} and EMD_{jk} should be close to the EMD value between O_k and the union of O_i and O_j. Furthermore, compound component should lie within the close neighborhood of each other. Notice that we want EMD to allow for partial matching here, therefore, during the identification of compound components, we do not scale the capacity of any feature distributions, which was used to make their capacities equivalent.

In our algorithm, we stay conservative during the identification of compound components. For any features that map to the same feature O_k with small EMD values and the summation of the EMD values is also close to the EMD value between the union of these features and O_k, we consider it as a compound component candidate. Additional criterion such as spatial closeness can be applied to rule out some plausible cases. We may identify some false-positive cases, but this will be taken care of by the global optimization algorithm. Notice that compound components identified by this method are only candidates to identify possible amalgamation/bifurcation. Whether amalgamation/bifurcation takes place still depends on the result of the global optimization.

It is also necessary to calculate the EMD value between a feature component and an empty component, which is used to detect creation/dissipation. We assume the empty component be a cell in space with its weight equal to that of the feature component being matched. The centroid of the empty component, i.e, the position of the cell, can be predicated by using the trajectory along which the feature has evolved. The Euclidean distance can be used as the ground distance function. In our implementation, we want to penalize more on creating or dissipating large components. So the nonlinear function in equation 2 is used as the ground distance.

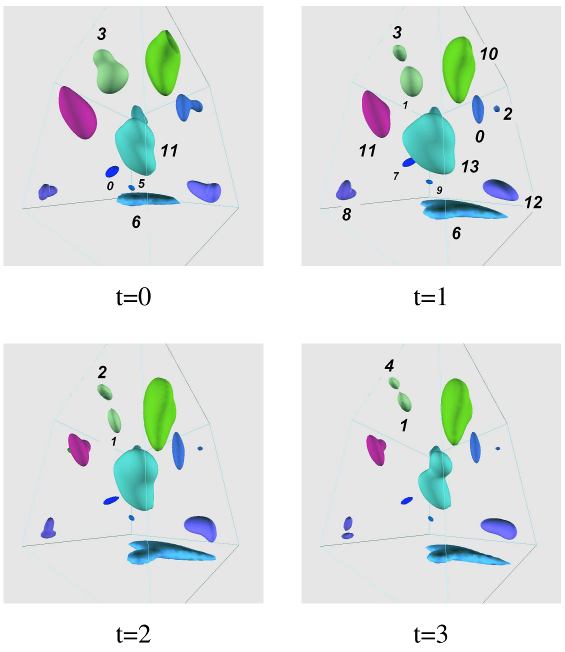


Figure 8. Time varying features from a force field data set are tracked and the tracking result determines the coloring of the features. Four snapshots are shown.

In order to detect amalgamation/bifurcation, we need to calculate the EMD value between single and compound components. This is simply achieved by merging the distribution of each individual component in the compound component as the distribution of the compound component. Then the EMD computation can be applied in a straightforward way.

6 Results

We have tested our algorithm using a 128^3 time-varying force field data set and a vorticity magnitude data set of the same resolution. The machine we used was a Pentium XEON 3GHz with 3G main memory.

The force field data set was generated by distributing ellipsoidal sites in space. The force of each point in space is the summation of the force it gathers from every site. Each ellipsoidal site has its own weight, which determines its influence to any given point in space. The force from

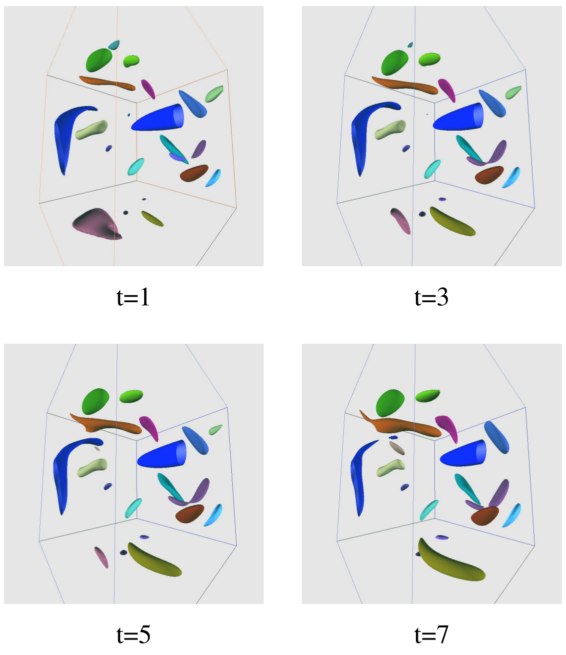


Figure 9. Time varying features from the vorticity data set are tracked and the tracking result determines the coloring of the features. Four snapshots are shown.

Time step	0-1	1-2	2-3
EMD computation time	0.042	0.037	0.029
Global optimization time	0.039	0.031	0.041

Table 1. The timing (in seconds) for the case in figure 8.

Feature	1st EMD	2nd EMD	3rd EMD	4th EMD
F_0^3	$F_1^1 F_1^3$ (2.65)	F_1^1 (4.03)	F_1^3 (8.46)	F_1^{10} (22.09)
F_0^6	F_1^6 (7.68)	*DIS* (23.25)	F_1^{13} (36.30)	F_1^{10} (40.61)
F_0^{11}	F_1^{13} (3.29)	F_1^7 (23.65)	F_1^0 (33.67)	F_1^9 (34.10)

Table 2. The EMD values for F_0^3, F_0^6 and F_0^{11}. The first four smallest values are shown. *DIS* means a feature dissipates.

an ellipsoidal site is anisotropic, i.e, stronger in some directions and weaker in other directions. Those ellipsoidal sites are moving in space with their weights changing too. Figure 8 shows the tracking result from the time-varying force field. The feature we track is the time-varying isosurface with an isovalue of 0.9. In the figure, each feature is colored in such a way that it has the same color as the feature it evolves from. When amalgamation happens where a feature evolves from multiple features, the current feature will get the color of the previous dominant feature. The dominance is defined by volume in our example, i.e., the current component would follow the color of the previous largest feature from which it evolves. In the case that a new feature is created, a new color is assigned to it. We also numbered part of the features for later explanation. These numbers are generated by feature extraction software and do not reflect the correspondence relationship between features.

There are multiple thin/small or fast-moving features in this example. For instance, F_0^6 (feature 6 at t=0) is a thin feature. Although it is not moving very fast, it does not overlap with F_1^6 which is the feature it will evolve into. F_0^0 and F_0^5 are small components which move fast relative to their own sizes. They have no overlap with the features they will evolve into either. It is hard for the tracking algorithms based on volume overlapping to detect the correspondence. But by using the EMD metric and global optimization, our algorithm detected the correspondence for those features correctly, as shown in the figure. In t=2, feature F_2^1 and F_2^2 move quickly. Feature F_2^1 and F_2^2 should correspond to F_3^1 and F_3^4, respectively. However, from the overlapping relationship, F_3^1 overlaps with both F_2^1 and F_2^2 and the overlap between F_3^1 and F_2^2 are larger than that between F_3^1 and F_2^1. Therefore, it is very likely that the tracking algorithms based on volume overlapping correspond F_3^1 to F_2^2 and treat F_2^1 and F_3^4 as dissipation and creation respectively. If the features are matched by using attributes such as centroid and volume, F_3^1 is closest to F_2^2 and they have similar volumes. Therefore, tracking algorithms based on similar-attribute are likely to correspond F_3^1 to F_2^2 as the volume-overlapping algorithms. By taking the global configuration into account and performing a global optimization based on the EMD values, our algorithm gave the correct result, as illustrated in the figure.

Timing results for the previous example are shown in table 1. The EMD values of some features are also shown in table 2. For each feature, the first four smallest EMDs are shown. We can see that the EMD value between a feature and the feature(s) it will evolve into is very small (in many cases it is the smallest). The EMD value between the feature and other features is usually much larger. This property makes the branch-and-bound method very efficient.

We also performed tests on the vorticity magnitude data set. The features we tracked are the time-varying isosurface with an isovalue of 6.9. Features are colored in the same way as in the previous example and reflect how the fea-

Time step	0-1	1-2	2-3	3-4	4-5
EMD computa -tion time	0.063	0.079	0.079	0.094	0.093
Global optimi -zation time	0.042	0.039	0.010	0.012	0.017

Table 3. The timing (in seconds) for the case in figure 9.

ture evolves. Four snapshots from the tracking results are shown in figure 9. Our global tracking algorithm gives accurate results with high efficiency. The result of the volume-overlapping based algorithms depends on a threshold value which measures what degree of overlapping should be considered as correspondence. If the threshold is set too high, there will be many creation/dissipation events. If the threshold is too small, features may get classified as continuation when they are not. Similar problems exist for the similar-attribute based algorithms. The result of those algorithms depends on the threshold values that are used to determine if the centroid positions, volumes, and masses etc are within the tolerance. However, by taking the global configuration of the features into consideration, our algorithm can determine the tracking results effectively without using threshold values. Timing results for the example in the figure 9 is shown in table 3.

7 Conclusions and Future Work

In this paper, we propose a global optimization algorithm to track time-varying features. Our algorithm is readily used to track any features that can be represented by the spatial distribution of cells. The algorithm can also be easily extended to track any features which can be represented by some other distribution information so that EMD can be used to measure feature dissimilarity. EMD is used as a better metric to measure the matching cost between any source and destination feature components. An efficient global optimization process is applied to find the globally best match among two feature sets. The efficiency comes from the branch-and-bound search algorithm, and a tighter estimate of the minimal cost further speeds up the search.

Reinders *et al.*[12] used a predication scheme extensively in their research to predict how the feature attributes change over time. We believe that the predication scheme will also be beneficial to our algorithm, especially during the EMD computation between a source and destination feature component. In the future work, we will attempt to incorporate the predication scheme into our work. It is also interesting to track sub-features, such as the tip on a surface, and to see how the sub-features change when time evolves.

References

[1] 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.

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

[3] 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.

[4] 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.

[5] I. Fujishiro, Y. Maeda, and H. Sato. Interval volume: A solid fitting technique for volumetric data display and analysis. In *Proceedings of IEEE Visualization 1995*, pages 151–158, 1995.

[6] B. Guo. Interval set: A volume rendering technique generalizing isosurface extraction. In *Proceedings of IEEE Visualization 1995*, pages 3–10, 1995.

[7] F.S. Hillier and G.J. Liberman. *Introducetion to Mathematical Programming*. McGraw-Hill, 1990.

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

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

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

[11] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of ACM SIGGRAPH 1987*, pages 163–169, 1987.

[12] 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.

[13] J. Rossignac and P. Borrel. Multi-resolution 3d approximations for rendering complex scenes. In *Geometric Modeling in Computer Graphics 1993*, pages 455–465, 1993.

[14] Y. Rubner, L.J. Guibas, and C. Tomasi. The earth mover's distance, multi-dimensional scaling, and color-based image retrieval. In *Proceedings of DARPA Image Understanding Workshop*, pages 661–668, 1997.

[15] Y. Rubner, C. Tomasi, and L.J. Guibas. A metric for distributions with applications to image databases. In *IEEE International Conference on Computer Vision 1998*, pages 59–66, 1998.

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

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

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

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

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

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

[22] B.S. Sohn and C. Bajaj. Time-varying contour topology. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):14–25, 2006.