Eurographics Conference on Visualization (EuroVis) 2017
J. Heer, T. Ropinski, and J. van Wijk
(Guest Editors)

Volume 36 (2017), Number 3

Global Feature Tracking and Similarity Estimation in Time-Dependent Scalar Fields

H. Saikia and T. Weinkauf

KTH Royal Institute of Technology, Stockholm, Sweden

Abstract

We present an algorithm for tracking regions in time-dependent scalar fields that uses global knowledge from all time steps for determining the tracks. The regions are defined using merge trees, thereby representing a hierarchical segmentation of the data in each time step. The similarity of regions of two consecutive time steps is measured using their volumetric overlap and a histogram difference. The main ingredient of our method is a directed acyclic graph that records all relevant similarity information as follows: the regions of all time steps are the nodes of the graph, the edges represent possible short feature tracks between consecutive time steps, and the edge weights are given by the similarity of the connected regions. We compute a feature track as the global solution of a shortest path problem in the graph. We use these results to steer the – to the best of our knowledge – first algorithm for spatio-temporal feature similarity estimation. Our algorithm works for 2D and 3D time-dependent scalar fields. We compare our results to previous work, showcase its robustness to noise, and exemplify its utility using several real-world data sets.

1. Introduction

Natural and technical phenomena often require a time-dependent description, since some of their major properties are only conceivable with a temporal dimension. Examples are the annual climate patterns, the heart beat of a human, or periodic vortex shedding in flows. Without the temporal dimension, significant aspects of these processes cannot be accounted for.

In this paper, we track and compare the temporal development of compact spatial regions in time-dependent scalar fields. Based on a robust tracking, we compare their tracks on a spatio-temporal level. This means that we take their development into account, such as their growing or shrinking in size and intensity, and distinguish features with different developments. For example, this allows us to reveal different types of vortices created due to periodic vortex shedding in flows. Most importantly, our work automates the cumbersome process of finding and tagging similarities in large, time-dependent data sets.

Our work combines two previously separate aspects of feature-based visualization, namely pattern matching and feature tracking, and provides contributions for both. The field of pattern matching recently gained some attention in the community. The basic premise is to find regions or features that are similar to a designed pattern or a selected region/feature. Existing methods address single time steps only, and do not find spatio-temporal similarities. Many existing tracking methods rely on local tracking decisions between two consecutive time steps and suffer from noise in the data, which leads to an overwhelming number of features and broken tracks. See Section 2 for an overview of previous work.

In this paper, we combine pattern matching and feature tracking to present – to the best of our knowledge – the first method for finding structural similarities of spatio-temporal structures in time-dependent scalar fields. This requires robust feature tracking and we contribute in this area with a novel approach that incorporates global information over all time steps to decide on particular feature tracks.

We consider structures defined by the sub/super level sets in a scalar field as given by merge trees and simplified using topological simplification to effectively deal with noisy data sets (Section 3). Essentially, these are compact spatial regions. We measure the pairwise spatial similarity of such regions using histograms similar to [SSW15]. We track the regions by solving a global shortest path problem on a graph data structure spanning all time steps and recording pairwise similarity between consecutive time steps (Section 4). This provides robust tracks. Given a selected track, we enumerate all structures with a similar spatial appearance and similar temporal development in the entire time-dependent scalar field (Section 5). We thoroughly evaluate our approach and compare it to the most related methods of Oesterling et al. [OHW*15] and Reininghaus et al. [RKWH12] (Section 6). We show results on several real-world data sets (Section 7) and conclude with a discussion (Section 8).

2. Related Work

The basic premise of pattern matching is to find regions or features that are similar to a designed pattern or a selected region/feature. Such methods exist for a large variety of data types such as images [Low04], geometry [MPWC13], scalar fields [KWKS11, SSW14, SSW15, TN11, TN13, TN14], vector fields [ES03, HEWK03, BHSH14], and multi-fields [WSW16]. All of

these methods address single time steps only, and are not adequate for finding spatio-temporal similarities: given a pattern, one may find similar features in a number of time steps, but this neglects any temporal evolution, since a progressively changing feature matches a pattern only for a certain amount of time. Furthermore, even if features remain similar to the initial pattern for their entire lifetime, these features may evolve differently and none of the above time-unaware methods can address these differences. In Section 7 we show how two different types of vortices cannot be differentiated by purely spatial measures, but our spatio-temporal approach is able to distinguish between them.

Many methods exist to track features in time-dependent data sets. Examples include the tracking of critical points in vector [TWSH02, GTS04, TS03, WTGP11] and scalar fields [EH04, RKWH12], tracking reeb graphs [EHM*08, WBD*11], contours [SB06] and merge trees [OHW*15], or the tracking of vortex structures [RSVP02, BP02, TSW*05, WSTH07].

Region-based tracking methods [SSZC94, SW97, JS06, MM09, DS16] often refrain from a particular feature definition and track regions by means of their similarity in consecutive time steps. Samtaney et al. [SSZC94] provide one of the first region-based tracking methods, later refined by Silver and Wang [SW97]. They track the connected components of superlevel/sublevel sets defined by a threshold, i.e., without referring to merge trees and topology, and solve the correspondence problem solely based on the information between two time steps. In contrast, we track more and hierarchically nested regions (defined by the subtrees of a merge tree) and solve the correspondence problem globally.

Most of the above tracking methods suffer from noise in the data, which leads to an overwhelming number of features and broken tracks. First attempts at rectifying this involve topological simplification [SN11, RKWH12, BWN*15], but have only been applied to 2D data sets.

Similar to our method, Widanagamaachchi et al. [WCBP12] track regions defined by merge trees and compute a *meta-graph* that connects a region to all overlapping regions in the next time step. The meta-graph captures the hierarchy of the merge trees, which serves to quickly obtain a tracking graph by filtering out nodes not overlapping at a given isolevel. The resulting tracking graph then contains features with similar intensity levels. In contrast, we track a selected merge tree region which may very well change its intensity level over time. Most importantly, we obtain a feature track by means of a global analysis, i.e., by solving the shortest path problem on the entire graph. In a follow-up article, Widanagamaachchi et al. [WCK*15] locally adapt the threshold to produce more temporally cohesive feature tracks.

Wang et al. [WRS*13] and Skraba and Wang [SW14] exploit robustness to solve the correspondence problem when tracking critical points in 2D time-dependent vector fields, i.e., a correspondence is established if two critical points are close to each other and have a similar robustness. This approach is able to deal with noisy data sets.

Related to our approach is the method of Oesterling et al. [OHW*15] for tracking merge trees in time-dependent scalar fields. While their method provides an exact history of changes to a merge tree, it is limited to small data sets as its running time is $O(n^3)$ with n being the number of voxels. In contrast, our method uses simplified merge trees only to define spatially compact regions, and tracks them based on their similarity, which provides a significant speed-up. However, both methods are based on merge trees and it is therefore interesting to make a more detailed comparison, see Section 6.4.

Feature Flow Fields [TS03, WTGP11] are a classic continuous tracking method often applied to tracking critical points. Reininghaus et al. [RKWH12] provide a combinatorial equivalent, which is among the few topology-based method that can robustly deal with noisy data sets. We provide a comparison to both approaches in Section 6.3.

Ozer et al. [OSBM14] detect spatio-temporal patterns in tracking graphs using petri-nets. This work is focused on describing and detecting events and transitions in tracking graphs based on a user-supplied, abstract description. In contrast, our work finds similar tracks to a selected one.

3. Background and Notation

We consider a time-dependent scalar field $s(\mathbf{x},t) : \mathbb{R}^{n+1} \to \mathbb{R}$ with n being the number of spatial dimensions. We restrict ourselves to 2D and 3D time-dependent scalar fields in this paper ($n = 2,3$), but note that our concepts should readily generalize to higher dimensions.

3.1. Merge Trees

We use the topological concept of *Merge Trees* to segment a scalar field into different regions. A merge tree provides a hierarchical segmentation, meaning that its leafs represent the finest segmentation based on all local maxima (or minima) of the scalar field, and higher levels combine these regions into larger ones thereby revealing the most dominant structures. In this paper, we track the regions defined by merge trees over time.

A more formal definition follows, see also Figure 1. Given is a single time step as a Morse function $f : \mathbb{R}^n \to \mathbb{R}$. We consider the parts of the domain where f attains a value larger than a value $c \in \mathbb{R}$ and define these as the superlevel sets $L_c^+ = \{\mathbf{x} | f(\mathbf{x}) \geq c\}$. Superlevel sets contain one or more connected components. Considering a decreasing value c, a component is born at every local maximum, and two or more components merge at saddle points (join saddles). Once all components have merged into one, this final component dies at the global minimum. A *merge tree* records this behavior by having the local maxima as its leaves, the saddles as its inner nodes, and the global minimum as its root. We call it also a *join tree* when it pertains to superlevel sets.

Similarly, we define the sublevel sets $L_c^- = \{\mathbf{x} | f(\mathbf{x}) \leq c\}$. They are born at minima, merge at split saddles, and unite at the global maximum. The merge tree, in this case, is called a *split tree*. For the sake of simplicity, we will refer only to superlevel sets and join trees from now on. The descriptions for sublevel sets are made in a similar way.

Our segmentation of the scalar field is given by the superlevel sets. Their count is equal to the number of edges in the merge tree [SSW14]. Note that this segmentation provides mutually exclusive regions only for a specific value of c, but we consider the merge tree in its entirety consisting of a hierarchical arrangement of regions

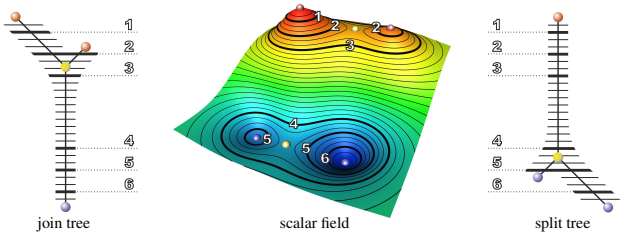


Figure 1: *Superlevel and sublevel sets of a scalar field may appear, join, split, and disappear with changing isovalues. The join and split tree represent that behavior. They are collectively referred to as merge trees.*

that are not mutually exclusive. Henceforth, we will refer to them as *subtree regions* or simply *regions*.

3.2. Simplification

Noise may affect the leafs and lower levels of the merge tree, but higher levels are increasingly robust against noise. To avoid unnecessary computations in later stages of our pipeline, it is beneficial to simplify the merge trees. The general idea is to repeatedly prune leafs off the tree, which conforms to a saddle-extremum cancellation. The order of cancellations can be determined in different manners. Persistence [ELZ02] is often used in this context, height difference [GRP*12] removes the currently smallest intensity fluctuation, or the volume of a region combines its area with its intensity [Car04]. We choose the latter for our experiments and refer the reader to a detailed discussion in [Car04].

4. Global Tracking of Subtree Regions

This section explains how we track subtree regions, while the next section uses these tracks to identify spatio-temporally similar features.

We process the given time-dependent scalar field in a streaming fashion, i.e., we keep only two time steps in main memory, compute their merge trees, compare their subtrees (Section 4.1), and record possible tracking connections in a graph data structure (Section 4.2). The graph is constructed from all time steps and serves us to identify the feature tracks by solving a global shortest path problem (Section 4.3). To the best of our knowledge, this is the first feature tracking approach that incorporates global information.

For brevity, we refrain from describing the computation of merge trees here and refer the interested reader to [Car04].

4.1. Comparison of Subtree Regions

Consider two subtree regions $\mathcal{S}_a$ and $\mathcal{S}_b$ from two consecutive time steps t_a and t_b. We want to measure their similarity to assess the likelihood that $\mathcal{S}_a$ tracks to $\mathcal{S}_b$. We combine two distance measures to serve this purpose: the volume overlap between the regions, and the similarity of their data.

The *volume overlap* d_o between two non-empty regions $\mathcal{S}_a$ and $\mathcal{S}_b$ is determined from the number of voxels they have in common and the total number of voxels covered by both regions:

$$d_o(\mathcal{S}_a,\mathcal{S}_b) = \frac{|\mathcal{S}_a \cap \mathcal{S}_b|}{|\mathcal{S}_a \cup \mathcal{S}_b|} \quad . \tag{1}$$

As can be seen, $d_o \in [0,1]$. It becomes 0 when there is no overlap, and 1 when $\mathcal{S}_a = \mathcal{S}_b$. Here, we determine the volume overlap on a voxel basis, assuming all voxels have the same volume. It is straightforward to extend this to other grids with varying voxel sizes. Please refer to the supplemental material for a detailed explanation on how this overlap is computed for all subtree pairs in two successive timesteps.

To compare the data of two regions we require a signature with enough discriminative power and being invariant against translation and rotation. Saikia et al. [SSW15] used the histogram of voxel intensities for this purpose. Birchfield and Rangarajan [BR05] propose spatiograms as a generalization of histograms including higher order moments and apply them in the context of computer vision. Thomas et al. [TN11] cluster similar regions together by grouping subtrees of the contour tree. We observe in our experiments that the histogram of voxel intensities works well and will use this signature in the following. Note that histograms are independent of translation and rotation, since just the data values are used and not their position.

We compute the distance between the histograms of two regions $\mathbf{h}_a$ and $\mathbf{h}_b$ using the Chi-Squared histogram distance (see e.g. [PW10])

$$\chi^2(\mathbf{h}_a,\mathbf{h}_b) = \frac{1}{2}\sum_i \frac{(h_{a,i}-h_{b,i})^2}{h_{a,i}+h_{b,i}} \ , \tag{2}$$

where $h_{a,i}$ and $h_{b,i}$ denote the bins of the histograms $\mathbf{h}_a$ and $\mathbf{h}_b$, respectively. The χ^2 distance can be computed as quickly as the L_2 norm, and additionally it provides a normalization that reduces the influence of large bins – a property that proves useful for increased discrimination of regions. However, we note that other distances could be used as well. We experimented with the the Earth Mover's Distance $\widehat{EMD}$ [PW09], but found it to be significantly slower and not providing more discriminative power in our experiments than the χ^2 distance.

The maximum value of the χ^2 distance is half the number of voxels in the data set, namely when one histogram is empty. We normalize the χ^2 values accordingly to the interval $[0,1]$ and refer to it henceforth as the *signature distance* d_s.

Both the volume overlap d_o and the signature distance d_s bear importance for tracking regions. If two regions $\mathcal{S}_a$ and $\mathcal{S}_b$ have high values for overlap and low values for the signature distance, then it is likely that $\mathcal{S}_a$ tracks to $\mathcal{S}_b$. We express this in a combined measure d_e using a linear combination

$$d_e(\mathcal{S}_a,\mathcal{S}_b) = \lambda \cdot (1-d_o) + (1-\lambda)\cdot d_s \tag{3}$$

with the weighting factor $\lambda \in [0,1]$. Note that $d_e \in [0,1]$ and low values indicate a high likelihood for a track between $\mathcal{S}_a$ and $\mathcal{S}_b$.

4.2. Recording Tracking Information in a Graph

Typical region-based tracking methods base their tracking decisions solely on a comparison of the regions between two consecutive time

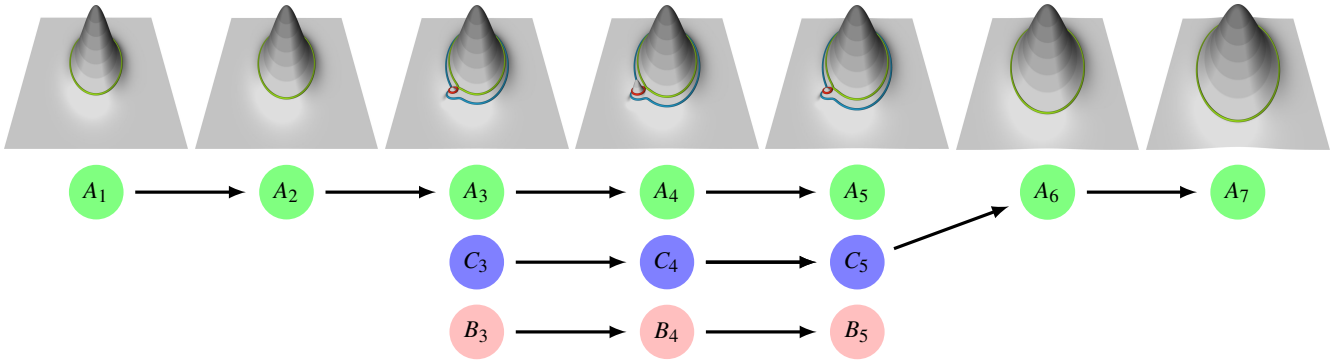


Figure 2: *Tracking regions solely based on local decisions leads to broken tracks. In this simple example, a small fluctuation between time steps t_3 and t_5 causes the creation of a region C that has significant overlap and similarity with region A. Assigning the locally best match neglects that there can be more than one suitable track between two time steps (e.g., between t_5 and t_6), and causes tracks to break. See Figure 3 for our graph structure solving this issue.*

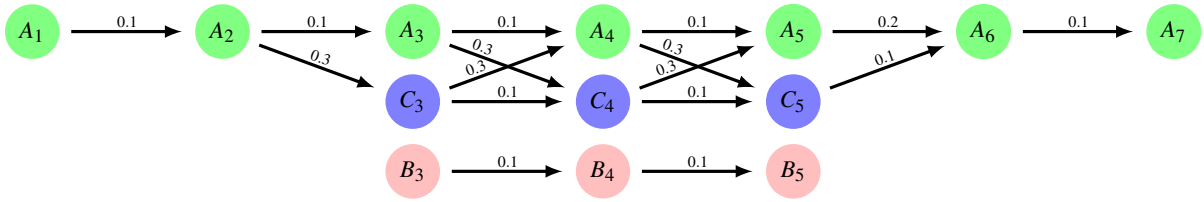


Figure 3: *We record suitable tracking information in a directed acyclic graph. The edges connect overlapping regions between consecutive time steps and are weighted with our combined region distance d_e. This minimal memory overhead allows us to track the regions without breaks, since we can solve ambiguities on a global level. See Figure 5 for region tracks that have been computed from this graph.*

steps, i.e., they try to find the best match between the regions. In certain situations, this leads to broken tracks.

Figure 2 exemplifies this: a region A can easily be tracked over several time steps, but then a small fluctuation creates another branch B in the merge tree. One can think of this as A being a large mountain and B being a small mountain right next to it. The saddle between A and B gives rise to a third region C encompassing both A and B. In topological terms, C is the parent of A and B in the merge tree. This new region C is very similar to A and they have significant overlap, since B is rather small. The tracks for A and C will run parallel as long as B exists, but the appearance/disappearance of B can lead to broken tracks. For example between t_5 and t_6: both A_5 and C_5 are very similar to A_6. In fact, if A is a growing region, then the distance between A_5 and A_6 could be larger than the distance between C_5 and A_6. If we only make a local decision based on the information between these two consecutive time steps, then we will assign C_5 to A_6. This leads to a broken track for the region A.

This issue cannot be fixed locally. Note in Figure 2 how the ending track for A overlaps for several time steps with the continuing track for C. One needs to understand this pattern globally (over several time steps) to address it.

Our approach fixes this issue by incorporating global information into the tracking decision. We build a graph data structure over all time steps as shown in Figure 3. The nodes are the subtree regions in each time step. The edges of the graph connect regions in consecutive time steps. Note that a region in t_i can be connected to more than one region in t_{i+1}, and vice versa. The edge weights are given by our combined region distance d_e. Since we assume unidirectional edges pointing from t_i to t_{i+1}, our graph is a *directed acyclic graph* (DAG).

Recording this information allows us to do tracking decisions based on a global view. We will detail in the next section how we solve a shortest path problem on the DAG to obtain robust tracking results with low computational effort. For the rest of this section, we will focus on the size of the graph.

Consider the nodes of two consecutive time steps t_i and t_{i+1} in the DAG. Let us assume there are n nodes for t_i and m nodes for t_{i+1}. If we were to create edges from all nodes in t_i to all nodes in t_{i+1}, then this would give us $n \times m$ edges. This becomes quickly a memory issue, and is therefore not a viable option.

The overlap between regions and the edge weights are the key to the sparsity of the DAG:

- We establish an edge between two regions only if they have some overlap, i.e., $d_o > 0$.
- Furthermore, the edge weight needs to be below a certain threshold, i.e., $d_e < \tau$.

These two rules reduce the amount of edges drastically. In particular,

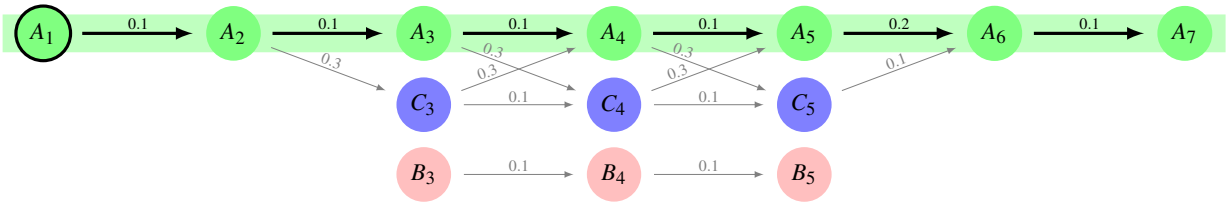


Figure 5: *Starting from a given region, we use the Dijkstra algorithm to find the shortest path through the DAG, which represents the track of this region. In this example, the shortest path was computed starting from A_1 and is shown as a green band.*

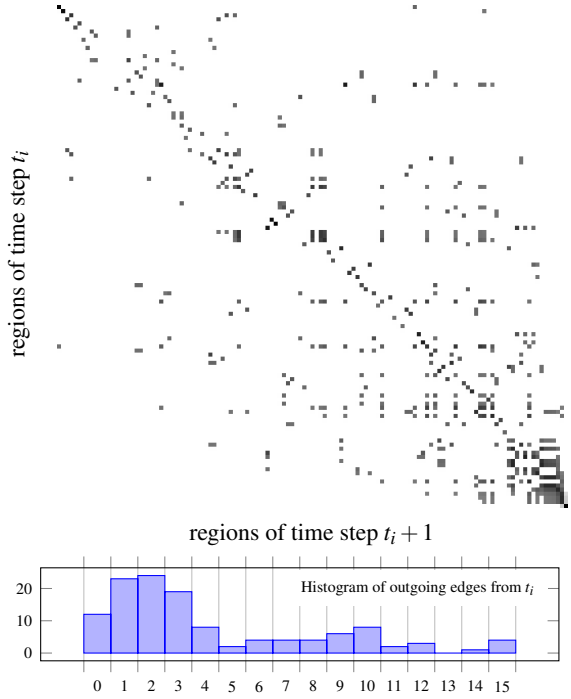


Figure 4: *The number of edges between two consecutive time steps in the DAG is very small, since we only establish edges between overlapping regions. The dot matrix on the top reveals this sparsity of the DAG for the Square Cylinder flow. Edges are depicted as grayscale dots where darker dots refer to lower distances between the regions according to the edge weight d_e. The histogram at the bottom shows that most nodes connect to 0 to 4 nodes in the next time step.*

the requirement that two regions have some overlap ($d_o > 0$) is responsible for most of the reduction.

Figure 4 reveals the sparsity of the DAG between two consecutive time steps in the Square Cylinder flow (see also Section 7). The merge trees of the two time steps give rise to 123 and 125 regions, respectively. An all-to-all connectivity would lead to 15375 edges. However, only 505 pairs have some overlap, which is a mere 3 percent. Figure 4 (top) shows the established edges as a dot matrix, where the grayscale level of the dots encodes the edge weight d_e. Note that we did not filter on d_e in this example, yet the number of edges is very small. We observe this behavior for all our data sets.

It is interesting to look at the edge statistics of this data set in more detail. Figure 4 (bottom) shows a histogram over the number of outgoing edges in time step t_i. Note how the majority of regions have 0 to 4 outgoing edges. The median is 3, the average is 4.11.

4.3. Tracking a Subtree Region in the Graph

Consider a region $\mathcal{S}_{t_i}$ at time step t_i. We want to track it forward and backward in time. This means to follow the edges in the DAG in forward or backward direction, and we want to incorporate a global constraint that effectively avoids broken tracks due to local decisions as discussed earlier.

We define the forward track of $\mathcal{S}_{t_i}$ as the *shortest path* starting from $\mathcal{S}_{t_i}$ in forward direction. This means, we are searching for the path for which the normalized squared sum of edge weights d_e is smallest. This ensures that consecutive regions in this path have considerable overlap, meaning minimal local displacement of features, as well as considerable similarity in signatures, ensuring minimal variation in size and shape. Similarly, we define the backward track of $\mathcal{S}_{t_i}$ as the shortest path starting from $\mathcal{S}_{t_i}$ in backward direction. The combination of backward and forward track give the desired track of $\mathcal{S}_{t_i}$. See Figure 5 for an illustration.

A formal description follows. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote the entire DAG as created in the previous section. A node $\mathcal{S}_{t_i} \in \mathcal{V}$ represents a region in the data set at time step t_i. We define a forward path $\mathcal{P}^+ = \langle \mathcal{S}_{t_i}, \ldots, \mathcal{S}_{t_n} \rangle$ as a sequence of nodes with connecting edges $(\mathcal{S}_{t_j}, \mathcal{S}_{t_{j+1}}) \in \mathcal{E}$ for $t_i \leq t_j < t_n$. Let us further require that $(\mathcal{S}_{t_n}, \mathcal{S}_{t_{n+1}}) \notin \mathcal{E}$, i.e., the path reached a node $\mathcal{S}_{t_n}$ that has no outgoing edges. Since $\mathcal{G}$ is a directed acyclic graph, paths always progress in time, i.e., each time step between t_i and t_n is represented by exactly one node. Let $\mathcal{H}^+$ denote the induced subgraph that contains all nodes and edges reachable from $\mathcal{S}_{t_i}$ via any forward path $\mathcal{P}^+$. We find the shortest path with respect to the normalized squared sum of edge weights d_e as follows:

$$\mathcal{P}^+_{\min} = \underset{\mathcal{P}^+ \in \mathcal{H}^+}{\arg\min} \sqrt{\frac{\sum d_e(\mathcal{S}_{t_j}, \mathcal{S}_{t_{j+1}})^2}{|\mathcal{P}^+| - 1}}. \tag{4}$$

Similarly, we find $\mathcal{P}^-_{\min}$ as the shortest path in backward direction. Finally, we stitch both shortest paths together to obtain the track of the region $\mathcal{S}_{t_i}$:

$$\mathcal{P}_{\min} = \left\langle \mathcal{P}^-_{\min}, \mathcal{P}^+_{\min} \right\rangle. \tag{5}$$

We solve (4) using Djikstra's shortest path algorithm [Dij59]. Let V be the number of vertices and E the number of edges. Djikstra's algorithm can be computed in $O(VlogV+E)$ time using a min-priority queue on a general graph. Since we have a directed acyclic graph, Djikstra's algorithm can be computed in $O(V)$ time, if we sort the vertices using *Topological Sorting* [CLRS01] which requires $O(V+E)$ time itself. Hence, the total runtime of our algorithm is $O(V+E)$.

On current commodity hardware, tracking a region is a matter of a few milliseconds even for larger data sets. See Section 6 for details.

4.4. Tracking all Subtree Regions in the Graph

A straightforward algorithm to track *all* subtree regions is to apply the algorithm from the previous section to each node in the DAG, and then remove duplicated tracks. Such an algorithm spends more computational effort than necessary, since a large number of duplicated or partly overlapping tracks can be expected in most data sets. We leave an optimized version to future research.

Our main application of finding spatio-temporally similar structures does not require extracting all tracks of all regions at once, as we will show in the next section.

5. Finding Spatio-Temporally Similar Structures

To the best of our knowledge, we present the first algorithm for finding spatio-temporally similar structures. These features exhibit similarity not only in a given time step, but also their temporal development is similar. This allows distinguishing features from each other based on their development, or it allows quantifying changes to periodic structures.

In a nutshell, our algorithm works as follows. The user selects a subtree region $\mathcal{S}$ in a time step. We search for spatially similar regions in the entire time-dependent data set, where similarity is determined using the signature distance d_s (normalized histogram comparison). We track each of these candidate matches over time and compare these tracks to the track of the initially selected $\mathcal{S}$ using Dynamic Time Warping. The best matches are selected via thresholding and visualized in a volume rendering framework.

A detailed description follows. We enable the user to inspect any region $\mathcal{S}$ in the DAG and view its temporal development by means of an animated volume rendering. Given a selected region $\mathcal{S}$ and its track, we are now tasked with finding similar tracks. The basic premise is that similar tracks will also contain a time step where the region is similar to the selected $\mathcal{S}$. Hence, we start with finding a set of *spatially* similar regions $\mathcal{S}'$. To do so, we compute the signature distance $d_s(\mathcal{S},\mathcal{S}')$ for all nodes in the DAG. This requires us to store the histogram at each node. We obtain a set of candidate match regions by thresholding d_s conservatively – more candidate matches avoid false negatives at the expense of slightly more computational effort.

We compute the track for each candidate match region $\mathcal{S}'$ and compare it to the track of the selected $\mathcal{S}$ using *Dynamic Time Warping* (DTW). This algorithm is often employed to compute the difference between two time series while allowing for temporal contraction and expansion. For a more detailed analysis of this behavior, please refer to the supplemental material. Consider the tracks $\mathcal{P}_{\min} = \langle \mathcal{S}_1,\ldots,\mathcal{S}_n\rangle$ and $\mathcal{P}'_{\min} = \langle \mathcal{S}'_1,\ldots,\mathcal{S}'_m\rangle$ to be compared, noting that they can have different lengths and can cover different time spans. The goal is now to match these tracks such that the sum of signature distances between matches is minimal. More formally, we describe a match as a sequence of index pairs $\{(p_1,p'_1),\ldots,(p_{\max(m,n)},p'_{\max(m,n)})\}$. Those indices are to be applied to the elements of the tracks $\mathcal{P}_{\min}$ and $\mathcal{P}'_{\min}$. They progress monotone through time, i.e., $p_i \le p_j$ and $p'_i \le p'_j$ for any $i \le j$. We want to find a match such that the sum of signature distances

$$\sum_{1\le k\le \max(m,n)} d_s(\mathcal{S}_{p_k},\mathcal{S}'_{p'_k}) \tag{6}$$

is minimized. This problem is similar to computing the Levenshtein distance [Nav01] and is conveniently solved using dynamic programming in $O(nm)$ time. An illustration of DTW matches is shown in Figure 6.

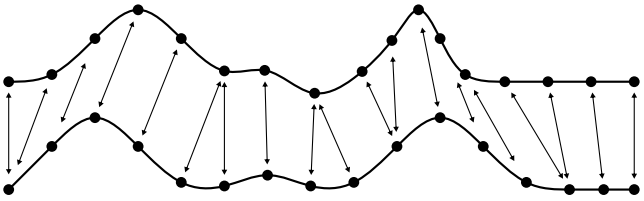


Figure 6: *Dynamic Time Warping matches two signals while allowing for temporal contractions and expansions. The arrows indicate the matched time steps. DTW employs an optimality criterion such that the sum of distances between the matches is minimal.*

6. Evaluation

6.1. Runtime and Memory Analysis

All our experiments were performed on a machine with a 2.3GHz Intel i7 processor and 16GB main memory.

Our method spends most of its time in computing the merge trees in each time step, all other steps of our approach are in the order of milliseconds per time step. Detailed numbers are given in Table 1. As can be seen, the method is quite fast.

The DAG itself does not require much memory as it is sparse, but the histograms attached to each node require a medium amount. In total assuming a 10% overlap requirement: 2D Checkerboard (34 MB), 2D Streak Line Curvature (540 MB), 3D Square Cylinder (96 MB), 3D Trefoil Knot (195 MB).

6.2. Translation and Rotation Invariance

To test whether our method is invariant against translation and rotation of features, we took a static scalar field and applied these transformations to it, thereby making it a 3D time-dependent field. We used the electrostatic field of the Benzene molecule for this test as shown in Figure 7. Our method tracks the regions of this data set flawlessly.

Data Set	Dimensions $x \times y \times z \times t$	Merge Tree and Simplification per time step	Average number of regions per time step	Overlap distance (d_o) per time step	Signature distance ($d_s = \chi^2$) per time step	Average time for Dijkstra's Shortest Paths	Average time for DTW comparison
2D Checkerboard	$128 \times 128 \times 1 \times 128$	19ms	44	0.9ms	0.5ms	0.4 ms	1.5 ms
2D Streak Line Curvature	$750 \times 136 \times 1 \times 368$	134ms	92.9	29ms	11.4ms	1.9 ms	25.6 ms
3D Square Cylinder	$192 \times 64 \times 48 \times 134$	2053ms	125.7	24ms	6.7ms	0.1 ms	0.2 ms
3D Trefoil Knot	$128 \times 128 \times 128 \times 500$	4241ms	64.4	12ms	1.3ms	0.2 ms	0.7 ms

Table 1: *Runtime statistics for merge tree and distance computations. We used 100 bins for our histogram signatures for all data sets. The average shortest path and DTW times are computed from all tracks started from a time step and a subsequent DTW comparison of all pairs.*

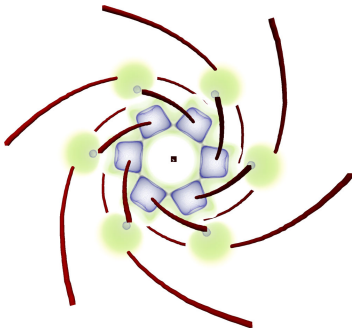
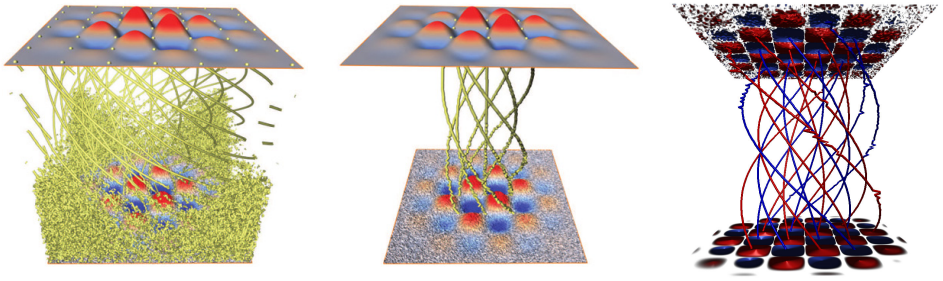


Figure 7: *Our method tracks rotated and translated regions flawlessly, which exemplifies its invariance against these transformations.*



(a) Stable FFF [WTGP11]. (b) Combin. FFF [RKWH12]. (c) Our method.

Figure 8: *Our tracking method is robust to noise and produces qualitatively as good results as Combinatorial Feature Flow Fields [RKWH12], since both methods can deal with noise by means of topological simplification. Continuous methods such as Stable Feature Flow Fields [WTGP11] are strongly affected by noise. Images reproduced with permission of the respective authors.*

6.3. Comparison to Reininghaus et al. [RKWH12] and Weinkauf et al. [WTGP11]

Feature Flow Fields are a well-established tool for extracting and tracking a large variety of features in different types of data. The general concept has been introduced by Theisel and Seidel [TS03]. As a numerical method, it builds upon derivatives, interpolation, and ordinary differential equations, which make it susceptible to noise. The main idea is to describe feature tracks as tangent curves in derived vector fields. A numerical stabilization has been developed by Weinkauf et al. [WTGP11], not to tackle noise, but to increase the stability of the numerical tangent curve integration.

Reininghaus et al. [RKWH12] introduced Combinatorial Feature Flow Fields for tracking critical points in 2D time-dependent scalar fields. While a generalization to 3D may be quite feasible, it is still missing. Their article contains a challenging example where a checkerboard of hills and valleys is rotated while noise is being added.

We use this test case to compare the three approaches. Figure 8 shows how the continuous approach fails in the presence of noise and produces only reasonable tracks near the smooth part of the data. Our method is able to track the hills and valleys in this data set as shown in Figure 8. This result is on a par with [RKWH12], which does not come as a surprise: both methods employ topological simplification to deal with noisy structures. While [RKWH12] uses Discrete Morse theory and tracks here only the saddles of the Morse-Smale complex, our method tracks regions defined by subtrees of merge trees. Although different features have been tracked, the results are clearly similar on a qualitative level.

We obtained our result here by first computing the join tree and tracking the hills (shown in red), and then computing the split tree and tracking the valleys (shown in blue). Tracking has been performed for all regions at $t = 0$ (bottom slice in Figure 8c). Since we track regions, it is slightly misleading to represent our tracking results as lines. However, it makes for a clearer overview and allows better comparison to [RKWH12]. We computed the lines via the center of mass of a tracked region in each time step, which explains the small zig-zag fluctuations. This example shows that our method is robust against noise.

6.4. Comparison to Oesterling et. al. [OHW*15]

Oesterling et. al. [OHW*15] present a method for tracking merge trees. In contrast to our method, their approach takes care to track changes to the hierarchy in the tree. In that sense, their output is significantly more detailed than ours.

On the flip side, this requires significantly more computation time. Its runtime complexity is polynomial in the data size, more precisely, it is $O(n^3)$ with n being the number of voxels. In contrast, our method depends on the topological complexity of the data, i.e., it is polynomial in the size of the number of subtrees of the merge tree.

We ran both methods on a very small 2D time-dependent example shown in Figure 9. Both methods gave the same result. Table 2 reports the timings. Note how tracking the merge tree depends on the data size and the method requires more than 50 minutes for a 60×60 data set with 11 time steps. Note how the computation times for our method practically remain constant as they depend on

Figure 9: *Simple Blobs data set. Two circular blobs emerge at the center, move apart and collapse into each other again.*

Size	Merge Tree Tracking [OHW*15]		Our Merge Tree Region Tracking	
	Avg. per time step	Total	Avg. per time step	Total
20x20	0.2 s	2 s	0.4 ms	4 ms
40x40	12 s	136 s	0.4 ms	4 ms
60x60	277 s	3052 s	0.5 ms	5 ms

Table 2: *Our method performs significantly faster than Oesterling et. al. [OHW*15] and may serve as a faster alternative when the focus is on tracking Morse cells independent of their hierarchy. However, if a record of the merge tree hierarchy evolution is desired, the method of Oesterling et. al. [OHW*15] is to be preferred. See Figure 9 for the data set. Note also that we report the timings in seconds and milliseconds, respectively.*

the topological complexity. The reported numbers for our method involve computing the overlap and signature distances between all pairs of regions, creating the DAG, and tracking all regions starting from a given time step. Note that we report these numbers in milliseconds.

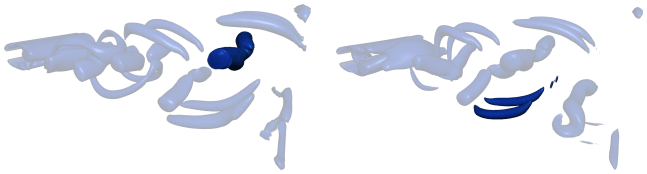
The timings for the merge tree tracking have been measured with our own implementation, but we contacted the authors of [OHW*15] and they confirmed this behavior [Hei16].

The method of Oesterling et. al. [OHW*15] is useful for providing exact information about the fate of every single element of a merge tree. Our method does not provide this information, but may serve as a faster alternative when the focus is on tracking Morse cells independent of their hierarchy.

7. Results

Square Cylinder Figure 12 shows the 3D time-dependent flow around a confined square cylinder. This is a direct numerical Navier Stokes simulation by Simone Camarri and Maria-Vittoria Salvetti (University of Pisa), Marcelo Buffoni (Politecnico of Torino), and Angelo Iollo (University of Bordeaux I) [CSBI05] which used to be publicly available at the now defunct International CFD Database. We use a uniformly sampled version [WHT12] for which we computed the Okubo-Weiss criterion, which is a scalar field indicating vortex behavior [SWTH07].

This flow exhibits periodic vortex shedding leading to the well known von Kármán vortex street. There are so-called primary and secondary vortex structures in this flow. The primary ones have a spanwise orientation, while the secondary ones have a streamwise orientation. We validate our method by finding and distinguishing these complicated periodic patterns. Note that this flow simulation is initiated from an impulsive start-up and the periodic vortex shedding develops with time. The flow becomes increasingly unsteady, which increases the total number of features and changes the shape and



(a) Primary vortex at $T = 65$. (b) Secondary vortex at $T = 31$.

Figure 10: *The shown primary and secondary vortex structures are close matches when considering only spatial information, but including their temporal development by means of DTW allows us to tell them apart: the secondary vortex is the 10th best spatial match of the primary vortex out of over 16000 regions in all time steps, but only the 82nd best spatio-temporal match out of the 100 best spatial matches. Compare to Figure 11.*

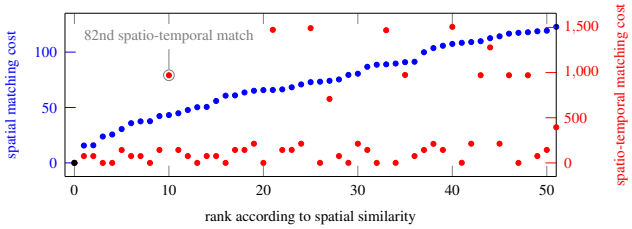


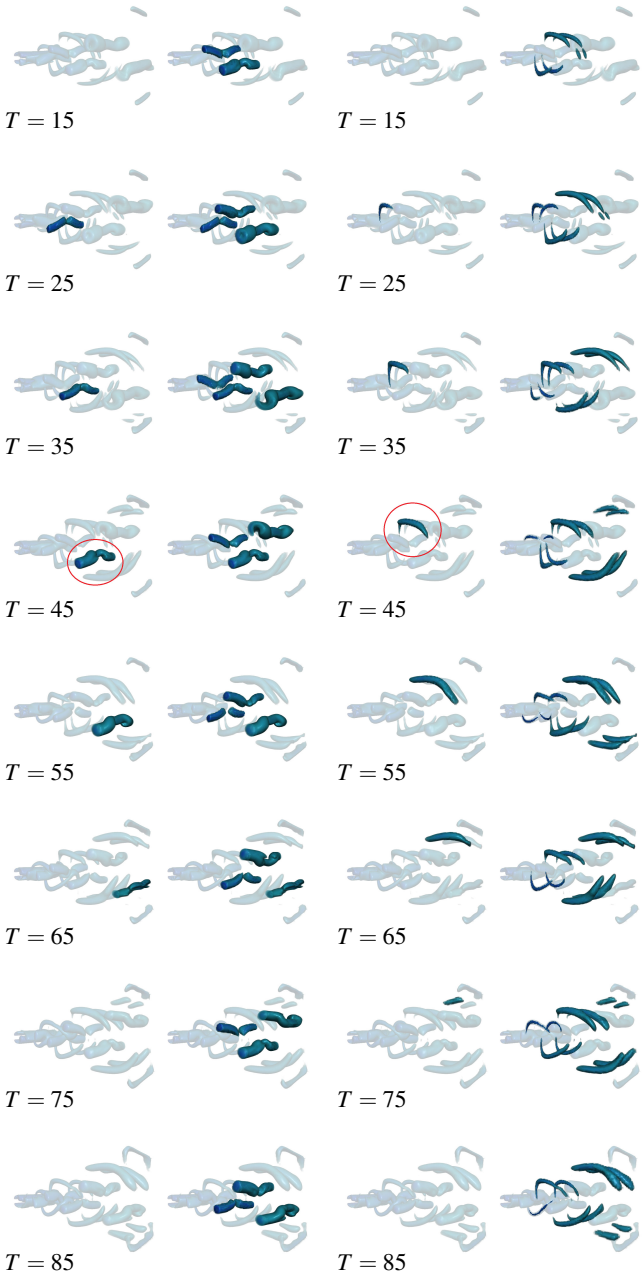
Figure 11: *Plot of spatial (blue) and spatio-temporal (red) matching costs of the selection from Figure 10(a). The discrepancy between spatial and spatio-temporal ranks reveals that adding the temporal dimension aids in discriminating structures with similar spatial scores based on their temporal evolution. Compare to Figure 10.*

intensity of the vortex structures to some extent. This is another stress test for our method.

A primary vortex has been selected in time step 45 as indicated in the left column of Figure 12a. This vortex was tracked backwards and forward in time using our DAG. Its entire track spans from time step 25 to 65. Based on this, we find all spatio-temporally similar structures as shown in the right column, and indeed these are the other primary vortices in this data set. Note how the life times of the other seven primary vortices extend over different time steps.

The secondary structures have been discovered in the same manner, see Figure 12b. This also shows that our method has enough discriminative power to distinguish these different vortex types. Figures 10 and 11 reveal the reason for this: it turns out that the spatial similarity measure (signature distance d_s) is not sufficient to distinguish between these different vortex types, since the secondary vortex from Figure 10b is the 10th best match for the primary vortex from Figure 10a out of over 16000 regions in all time steps. Starting from this, our method takes the best spatial matches including this false positive, tracks each of them over time, and then compares these tracks against the track of the primary vortex using DTW. Differences in the temporal evolution are revealed this way and the false positive has been identified.

Streak Line Curvature We tracked drastically changing regions in a 2D parameter-dependent scalar field denoting the curvature of



(a) Primary vortex structures. Selected track (left) and matching tracks (right).

(b) Secondary vortex structures. Selected track (left) and matching tracks (right).

Figure 12: *A primary and secondary vortex structure have been selected at $T = 45$ and tracked backwards and forwards in time. Their tracks have been used to find spatio-temporally similar structures in the entire data set. Note how the discriminative power of our signature distance d_s enables us to distinguish between both types of vortices, and how the matched vortex tracks span through the entire time of the data set.*

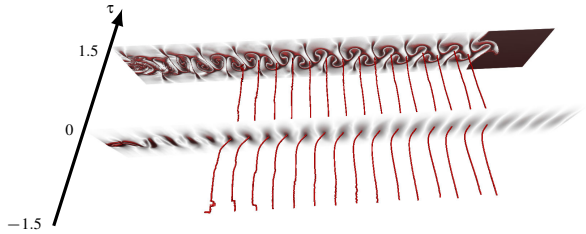


Figure 13: *Our method is able to track regions even if they change their properties drastically over the course of the track. Shown are tracked regions in the streak line curvature field for a 2D flow around a cylinder. These regions indicate vortex activity.*

streak lines in a 2D flow around a cylinder, see Figure 13. Based on the streak line vector field of Weinkauf and Theisel [WT10], the curvature of streak lines can be computed without actually computing any streak lines, but just by means of partial derivatives. Streak lines are characteristic curves of flows and describe how smoke or dye is being transported in a flow when released from a fixed position. An important parameter is their integration time τ, which is the amount of time that the oldest particle of a streak line spent in the flow. With increasing τ, the shape of a streak line becomes more complex due to the flow, and its curvature changes accordingly. Figure 13 reveals how strongly the curvature field changes with increasing τ. Note the difference between the shown slices for $\tau = 0$ and $\tau = 1.5$.

We tracked the regions from $\tau = 0$ through the data set and obtained tracks that reveal the periodicity of the flow and show how our method is able to track regions even if they change their properties drastically.

Trefoil Knot The time-dependent Trefoil field represents three interlocked magnetic flux tubes and is used to study magnetic energy decay processes like coronal mass ejections of the sun [CDSB11]. We are investigating the magnitude of the magnetic field lines. The initial configuration is a trefoil knot of flux tubes, but the simulation quickly progresses to a decayed state. It is challenging for a user to investigate these structures manually. Our tool can help in this process by automating parts of the analysis: in Figure 14, the user selects a region near the inner ring in time step $T = 200$, then our system tracks it through time and finds spatio-temporally similar tracks. They reveal the still prevailing 3-symmetry in this data set despite the obvious decay. This example shows how our method can be useful to reveal structure and symmetry in seemingly chaotic data.

8. Conclusion

Tracking features in complex and noisy data sets requires some form of noise reduction or simplification technique to give meaningful results. The classic continuous methods such as Feature Flow Fields [TS03, WTGP11] or the linear-element methods [TWSH02, GTS04] lack this. Still, assuming one can deal with the numerics, these continuous methods provide the exact solutions that every method has to live up to. Combinatorial methods such as Jacobi sets [EH04] or Combinatorial Feature Flow Fields [RKWH12] take their inspiration from the continuous methods and enable simplification

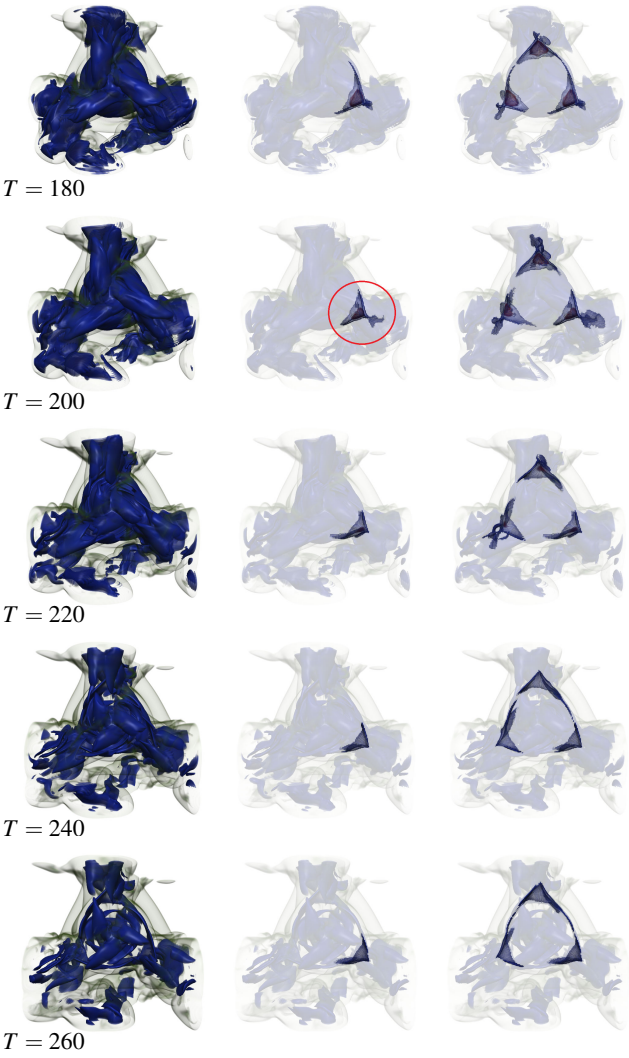


Figure 14: *Our method is useful to reveal structure and symmetry in seemingly chaotic data. The Trefoil data set shows the decay of three interlocked magnetic flux tubes. The left column shows a full volume rendering. The middle column shows the user selection and its track. The right column shows the spatio-temporally similar regions revealing the 3-symmetry of the data set.*

[SN11, BWN*15]. Yet, these methods still require further development as they have practically been shown only for 2D data, and the 3D case is particularly difficult for Jacobi sets. Merge tree tracking [OHW*15] is an alternative combinatorial method that supports simplification, but unfortunately requires very high computational effort.

Our tracking method supports simplification to deal with noise. It is based on topological features that have proven highly useful in the past for e.g. vortex analysis in flows. But in contrast to other combinatorial methods, it employs a very fast, histogram-based tracking scheme. As shown in Sections 6 and 7, the results are of high quality and on a par with other methods, but we obtain them significantly faster, and for 2D and 3D data.

This has been made possible by our first contribution: a tracking graph that records several alternative local tracking decisions to support a global optimization of feature tracks. In future research we would like to lift the restriction of requiring overlap between regions in consecutive time steps.

Our second contribution is the first method for finding spatio-temporally features in time-dependent data sets. As shown in Section 7, it is highly useful for automating parts of the analysis process and for revealing structure and symmetry in seemingly chaotic data. A possible option for future work in this regard is to find all interesting groups of spatio-temporal features by means of clustering the set of all feature tracks.

Acknowledgments

This work was supported through a grant from the Swedish e-Science Research Centre (SeRC). The presented concepts have been developed and evaluated in the Inviwo framework.

References

[BHSH14] BUJACK R., HOTZ I., SCHEUERMANN G., HITZER E.: Moment invariants for 2d flow fields using normalization. In *Proc. IEEE Pacific Visualization* (2014), pp. 41–48. 1

[BP02] BAUER D., PEIKERT R.: Vortex tracking in scale space. In *Proc. VisSym* (2002), pp. 233–240. 2

[BR05] BIRCHFIELD S. T., RANGARAJAN S.: Spatiograms versus histograms for region-based tracking. In *IEEE CVPR* (June 2005), vol. 2, pp. 1158–1163 vol. 2. 3

[BWN*15] BHATIA H., WANG B., NORGARD G., PASCUCCI V., BREMER P.-T.: Local, smooth, and consistent Jacobi set simplification. *Computational Geometry 48*, 4 (2015), 311–332. 2, 10

[Car04] CARR H.: *Topological Manipulation of Isosurfaces*. PhD thesis, The University of British Columbia, 2004. 3

[CDSB11] CANDELARESI S., DEL SORDO F., BRANDENBURG A.: Decay of trefoil and other magnetic knots. In *Proc. Advances in Plasma Astrophysics* (2011), pp. 461–463. 9

[CLRS01] CORMEN T. H., LEISERSON C. E., RIVEST R. L., STEIN C.: *Introduction to algorithms*, vol. 6. MIT press Cambridge, 2001. 6

[CSBI05] CAMARRI S., SALVETTI M.-V., BUFFONI M., IOLLO A.: Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate reynolds numbers. In *XVII Congresso di Meccanica Teorica ed Applicata* (2005). 8

[Dij59] DIJKSTRA E. W.: A note on two problems in connexion with graphs. *Numerische Mathematik 1*, 1 (1959), 269–271. 6

[DS16] DUTTA S., SHEN H.-W.: Distribution driven extraction and tracking of features for time-varying data analysis. *IEEE TVCG 22*, 1 (2016), 837–846. 2

[EH04] EDELSBRUNNER H., HARER J.: Jacobi sets of multiple Morse functions. In *Foundations of Computational Mathematics: Minneapolis 2002*, Cucker F., DeVore R., Olver P., Süli E., (Eds.). Cambridge Universtiy Press, 2004, pp. 37–57. 2, 9

[EHM*08] EDELSBRUNNER H., HARER J., MASCARENHAS A., PASCUCCI V., SNOEYINK J.: Time-varying reeb graphs for continuous space–time data. *Computational Geometry 41*, 3 (2008), 149–166. 2

[ELZ02] EDELSBRUNNER H., LETSCHER D., ZOMORODIAN A.: Topological persistence and simplification. *Discrete and Computational Geometry 28*, 4 (2002), 511 – 533. 3

[ES03] EBLING J., SCHEUERMANN G.: Clifford convolution and pattern matching on vector fields. In *Proc. IEEE Visualization* (2003), pp. 193–200. 1

[GRP*12] GÜNTHER D., REININGHAUS J., PROHASKA S., WEINKAUF T., HEGE H.-C.: Efficient computation of a hierarchy of discrete 3d gradient vector fields. In *Topological Methods in Data Analysis and Visualization II*. Springer, 2012, pp. 15–30. 3

[GTS04] GARTH C., TRICOCHE X., SCHEUERMANN G.: Tracking of vector field singularities in unstructured 3D time-dependent datasets. In *Proc. IEEE Visualization* (2004), pp. 329–336. 2, 9

[Hei16] HEINE C.:, 2016. private communication. 8

[HEWK03] HEIBERG E., EBBERS T., WIGSTRÖM L., KARLSSON M.: Three dimensional flow characterization using vector pattern matching. *IEEE TVCG 9*, 3 (2003), 313–319. 1

[JS06] JI G., SHEN H.-W.: Feature tracking using earth mover's distance and global optimization. In *Pacific Graphics* (2006). 2

[KWKS11] KERBER J., WAND M., KRÜGER J., SEIDEL H.-P.: Partial symmetry detection in volume data. In *Vision, Modeling, and Visualization* (2011), pp. 41–48. 1

[Low04] LOWE D. G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (Nov. 2004), 91–110. 1

[MM09] MUELDER C., MA K. L.: Interactive feature extraction and tracking by utilizing region coherency. In *IEEE Pacific Visualization Symposium* (April 2009), pp. 17–24. 2

[MPWC13] MITRA N. J., PAULY M., WAND M., CEYLAN D.: Symmetry in 3D geometry: Extraction and applications. *Computer Graphics Forum 32*, 6 (2013), 1–23. 1

[Nav01] NAVARRO G.: A guided tour to approximate string matching. *ACM Comput. Surv. 33*, 1 (Mar. 2001), 31–88. 6

[OHW*15] OESTERLING P., HEINE C., WEBER G. H., MOROZOV D., SCHEUERMANN G.: Computing and visualizing time-varying merge trees for high-dimensional data. In *Topology-Based Methods in Visualization (TopoInVis)* (2015). 1, 2, 7, 8, 10

[OSBM14] OZER S., SILVER D., BEMIS K., MARTIN P.: Activity detection in scientific visualization. *IEEE TVCG 20*, 3 (March 2014), 377–390. 2

[PW09] PELE O., WERMAN M.: Fast and robust earth mover's distances. In *IEEE ICCV* (2009), IEEE, pp. 460–467. 3

[PW10] PELE O., WERMAN M.: The quadratic-chi histogram distance family. In *ECCV* (2010), Springer, pp. 749–762. 3

[RKWH12] REININGHAUS J., KASTEN J., WEINKAUF T., HOTZ I.: Efficient computation of Combinatorial Feature Flow Fields. *IEEE TVCG 18*, 9 (September 2012), 1563–1573. 1, 2, 7, 9

[RSVP02] REINDERS F., SADARJOEN I. A., VROLIJK B., POST F. H.: Vortex tracking and visualisation in a flow past a tapered cylinder. *Computer Graphics Forum 21*, 4 (Nov. 2002), 675–682. 2

[SB06] SOHN B. S., BAJAJ C.: Time-varying contour topology. *IEEE TVCG 12*, 1 (Jan 2006), 14–25. 2

[SN11] SUTHAMBHARA N., NATARAJAN V.: Simplification of Jacobi sets. In *Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications*, Pascucci V., Tricoche X., Hagen H., Tierny J., (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 91–102. 2, 10

[SSW14] SAIKIA H., SEIDEL H.-P., WEINKAUF T.: Extended branch decomposition graphs: Structural comparison of scalar data. *Computer Graphics Forum (Proc. EuroVis) 33*, 3 (June 2014), 41–50. 1, 2

[SSW15] SAIKIA H., SEIDEL H.-P., WEINKAUF T.: Fast similarity search in scalar fields using merging histograms. In *TopoInVis* (Annweiler, Germany, May 2015), Carr H., Garth C., Weinkauf T., (Eds.), pp. 1–14. 1, 3

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

[SW97] SILVER D., WANG X.: Tracking and visualizing turbulent 3d features. *IEEE TVCG 3*, 2 (Apr. 1997), 129–141. 2

[SW14] SKRABA P., WANG B.: Interpreting feature tracking through the lens of robustness. In *Topological Methods in Data Analysis and Visualization III, Theory, Algorithms, and Applications*, Bremer P.-T., Hotz I., Pascucci V., Peikert R., (Eds.). Springer, 2014, pp. 19–37. 2

[SWTH07] SAHNER J., WEINKAUF T., TEUBER N., HEGE H.-C.: Vortex and strain skeletons in eulerian and lagrangian frames. *IEEE TVCG 13*, 5 (September - October 2007), 980–990. 8

[TN11] THOMAS D. M., NATARAJAN V.: Symmetry in scalar field topology. *IEEE TVCG 17*, 12 (2011), 2035–2044. 1, 3

[TN13] THOMAS D. M., NATARAJAN V.: Detecting symmetry in scalar fields using augmented extremum graphs. *IEEE TVCG 19*, 12 (2013), 2663–2672. 1

[TN14] THOMAS D., NATARAJAN V.: Multiscale symmetry detection in scalar fields by clustering contours. *IEEE TVCG 20*, 12 (Dec 2014), 2427–2436. 1

[TS03] THEISEL H., SEIDEL H.-P.: Feature flow fields. In *Proc. VisSym* (2003), pp. 141–148. 2, 7, 9

[TSW*05] THEISEL H., SAHNER J., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Extraction of parallel vector surfaces in 3D time-dependent fields and application to vortex core line tracking. In *Proc. IEEE Visualization* (2005), pp. 631–638. 2

[TWSH02] TRICOCHE X., WISCHGOLL T., SCHEUERMANN G., HAGEN H.: Topology tracking for the visualization of time-dependent two-dimensional flows. *Computers & Graphics 26* (2002), 249–257. 2, 9

[WBD*11] WEBER G., BREMER P.-T., DAY M., BELL J., PASCUCCI V.: Feature tracking using reeb graphs. In *Topological Methods in Data Analysis and Visualization: Theory, Algorithms, and Applications*, Pascucci V., Tricoche X., Hagen H., Tierny J., (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 241–253. 2

[WCBP12] WIDANAGAMAACHCHI W., CHRISTENSEN C., BREMER P.-T., PASCUCCI V.: Interactive exploration of large-scale time-varying data using dynamic tracking graphs. In *IEEE LDAV* (2012), Barga R. S., Pfister H., Rogers D. H., (Eds.), IEEE, pp. 9–17. 2

[WCK*15] WIDANAGAMAACHCHI W., CHEN J., KLACANSKY P., PASCUCCI V., KOLLA H., BHAGATWALA A., BREMER P.-T.: Tracking features in embedded surfaces: Understanding extinction in turbulent combustion. In *IEEE LDAV* (2015), Bennett J., Childs H., Hadwiger M., (Eds.), IEEE Computer Society, pp. 9–16. 2

[WHT12] WEINKAUF T., HEGE H.-C., THEISEL H.: Advected tangent curves: A general scheme for characteristic curves of flow fields. *Computer Graphics Forum (Proc. Eurographics) 31*, 2 (April 2012), 825–834. 8

[WRS*13] WANG B., ROSEN P., SKRABA P., BHATIA H., PASCUCCI V.: Visualizing robustness of critical points for 2d time-varying vector fields. *Computer Graphics Forum 32*, 3 (2013), 221–230. 2

[WSTH07] WEINKAUF T., SAHNER J., THEISEL H., HEGE H.-C.: Cores of swirling particle motion in unsteady flows. *IEEE TVCG (Proc. IEEE Visualization) 13*, 6 (November – December 2007), 1759–1766. 2

[WSW16] WANG Z., SEIDEL H.-P., WEINKAUF T.: Multi-field pattern matching based on sparse feature sampling. *IEEE TVCG (Proc. IEEE VIS) 22*, 1 (January 2016), 807–816. 1

[WT10] WEINKAUF T., THEISEL H.: Streak lines as tangent curves of a derived vector field. *IEEE TVCG (Proc. IEEE Visualization) 16*, 6 (November - December 2010), 1225–1234. 9

[WTGP11] WEINKAUF T., THEISEL H., GELDER A. V., PANG A.: Stable Feature Flow Fields. *IEEE TVCG 17*, 6 (June 2011), 770–780. 2, 7, 9