Feature Tracking by Two-Step Optimization

Andrea Schnorr, Dirk N. Helmrich, Dominik Denker, Torsten W. Kuhlen, and Bernd Hentschel

Abstract—Tracking the temporal evolution of features in time-varying data is a key method in visualization. For typical feature definitions, such as vortices, objects are sparsely distributed over the data domain. In this paper, we present a novel approach for tracking both sparse and *space-filling* features. While the former comprise only a small fraction of the domain, the latter form a set of objects whose union covers the domain entirely while the individual objects are mutually disjunct. Our approach determines the assignment of features between two successive time-steps by solving two graph optimization problems. It first resolves one-to-one assignments of features by computing a maximum-weight, maximum-cardinality matching on a weighted bi-partite graph. Second, our algorithm detects events by creating a graph of potentially conflicting event explanations and finding a weighted, independent set in it. We demonstrate our method's effectiveness on synthetic and simulation data sets, the former of which enables quantitative evaluation because of the availability of ground-truth information. Here, our method performs on par or better than a well-established reference algorithm. In addition, manual visual inspection by our collaborators confirm the results' plausibility for simulation data.

Index Terms—Global optimization, simulation output analysis, flow visualization

1 Introduction

The analysis of time-varying phenomena is a key aspect of scientific visualization. Time-dependent data is usually given as a series $\{S_t\}_{1\leq t\leq T}$ of $T \in \mathbb{N}$ discrete snapshots S_t, each of which captures the underlying system's state at time t. A proven strategy to deal with the massive amounts of time-dependent simulation data is *feature-based visualization* [1]: the analysis focuses on objects of interest—the features—which carry a domain-specific meaning. In this regard, each snapshot of a time-dependent data set yields a set of *feature objects*, which represents the state of a spatio-temporal *feature* at a given time. The goal of *feature tracking* is to algorithmically find corresponding feature objects in successive time steps in order to assemble them into spatio-temporal features. Those correspondences across successive time steps represent the individual *explanations* in a feature's evolution, namely continuations, splits, merges, births, and deaths [2].

Classically, feature-based visualization has focused on structures which cover only a small portion of the data domain. In this work, we refer to such features as *sparse*. Given this property and a sufficiently high temporal resolution, existing approaches reliably solve the tracking problem. Beyond this setting, we are particularly interested in features which are densely packed in the data domain. In the extreme case, these features partition the entire domain, i.e., they are *space-filling*. Thus, each data point of every time step is part of exactly one feature object.

While our work targets feature tracking in general, the extension to a space-filling setting is motivated by a close collaboration with domain experts from theoretical fluid mechanics. They have been actively working on the statistical analysis of features that are space-filling: *dissipation elements* (DEs) [3], [4], [5]. A DE is defined as the finite volume from which each trajectory of a scalar gradient starts and ends in the same pair of minimum and maximum. Thus, each domain point is uniquely assigned to exactly one DE and the set of DEs is space-filling by definition. While our collaborators have focused on static snapshots of flow fields so far, they now aim at extending their analysis to a time-dependent setting and are, hence, interested in tracking DEs. Due to the fact that even relatively small data sets contain tens of thousands of elements, their time-dependent analysis requires a robust, fully-automatic method to track them.

The extension to space-filling structures substantially complicates the tracking problem in two ways: the number of potential explanations grows exponentially and these explanations might be mutually contradicting. Regarding the first, the number of feature objects that need to be considered in order to explain a given object's evolution due to their overlap with that specific object in a neighboring time step grows significantly. Because several of these candidate objects might be involved in an event, an exhaustive search for the best fit has to consider the power set of all candidates. Hence, the number of potential explanations grows exponentially. As a result, the feature tracking problem quickly becomes intractable for space-filling structures. A detailed inspection w.r.t. actual problem sizes is given in Section 4.2. Second, explanations correlating one feature object to one or more feature objects in a neighboring time step might be mutually contradicting: colloquially, feature objects might "compete" for an assignment to one or

- *A. Schnorr, D. Helmrich, T. Kuhlen, and B. Hentschel are with JARA – High-Performance Computing and the Visual Computing Institute, RWTH Aachen University, Aachen 52062, Germany. E-mail: {schnorr, helmrich, kuhlen, hentschel}@vr.rwth-aachen.de.*
- *D. Denker is with the Institute for Combustion Technology, RWTH Aachen University, Aachen 52062, Germany. E-mail: ddenker@itv.rwth-aachen.de.*

Manuscript received 22 Dec. 2017; revised 31 Aug. 2018; accepted 8 Sept. 2018. Date of publication 27 Nov. 2018; date of current version 5 May 2020. (Corresponding author: Andrea Schnorr).
Recommended for acceptance by X. M Tricoche.
Digital Object Identifier no. 10.1109/TVCG.2018.2883630

more candidates. This complicates finding a valid assignment, i.e., one which is free of contradictions.

Against this backdrop, we propose a new feature tracking algorithm which identifies corresponding objects from successive time steps by means of two graph optimization problems. First, we compute linear continuations from t to $t+1$ by solving a *weighted, bi-partite matching problem*, as described in Section 3.1. This guarantees a globally optimal $1:1$ assignment of feature objects under the selected weight function. The initial solution is, however, limited to linear assignments; it cannot describe $1:n$ or $n:1$ events, i.e., splits or merges. In a second step, we detect these events by augmenting the $1:1$ assignments from the matching phase by additional edges. In order to solve this task, we model it as a *maximum-weight independent set problem* over the set of all potential explanations which are compatible with the initial matching. We detail this step in Section 3.2.

Our design is based on the following rationale. In principle, identifying a tracking solution from the set of all viable explanations for the temporal evolution of features from one time step to the next could be formulated as a maximum-weight independent set problem. However, this problem is NP-hard in general. Thus, unrestricted instances will be intractable in practice. Therefore, we prune the search space by assuming that continuations make up the vast majority of explanations. This is backed by previous work [6] and domain experts [3]. The assumption allows us to compute linear assignments efficiently by means of a weighted, bi-partite matching problem. The restriction of the set of viable explanations to the ones conforming to the matching then brings the independent set problem down to a manageable size.

We analyze the performance of our approach in Section 4 by applying it to a number of different data sets including both sparse and space-filling features. In addition, we present a comparison of our approach to the volume tracking algorithm by Silver and Wang [7]. First, we test the approach on synthetic data which include ground truth results against which to compare. Even though such synthetic data sets do not fully emulate realistic behavior of fluid dynamics, they facilitate a structured assessment of the algorithm's performance in a controlled scenario. This enables us to reason about any evident issues. Second, we test our method on different simulation data sets and perform a phenomenological analysis by visually inspecting the results for plausibility. In Section 5, we discuss our approach with an eye on known limitations, before concluding in Section 6.

In summary, we make the following contributions. First, we propose a new feature tracking algorithm which handles sparse as well as space-filling features by successively solving two graph optimization problems. Second, we demonstrate the functionality of this approach by means of a novel analysis method utilizing different synthetic data sets that provide a ground truth to enable a quantification, and the analysis of real-world application data sets from fluid mechanics research. Observations in the latter case are backed by discussions with domain experts.

2 Related Work

The goal of feature tracking is to algorithmically determine the temporal evolution of features in a series of data snapshots $\{S_t\}_{1\leq t\leq T}$. Before we can track their evolution, however, features have to be defined and extracted. A full review of feature definitions and extraction algorithms is beyond the scope of this paper. Typical examples include vortices [8], [9], [10], shocks [11], or topological structures [12], [13], [14]. As outlined above, our work is motivated by a specific feature definition: dissipation elements [3]. DEs are tightly related to the notion of the 3D-Morse Smale Complex [15]; in fact, by their definition, DEs are equivalent to 3D Morse Smale cells. Gyulassy et al. recently used this link to assess the stability of DEs [16].

The input for feature tracking is a series of sets of feature objects extracted from $\{S_t\}_{1\leq t\leq T}$. It is generally assumed that the snapshots are sampled densely enough in the temporal dimension to allow for a faithful reconstruction of temporal relations; temporal undersampling leads to an ill-posed problem [6].

Finding a globally optimal assignment—i.e., one that maximizes feature similarity over all T time steps—is NP-hard for $T \geq 3$ [17], [18]. Therefore, feature tracking algorithms typically try to find correspondences between two successive time steps. In scientific visualization, this was introduced by Samtaney et al. [2]. They define *correspondence criteria* for the canonical explanations of continuations, split (bifurcation), merge (amalgamation), birth (creation), and death (dissipation). Evolutions are determined according to a greedy strategy, which implicitly resolves conflicting matches by removing feature objects from the search space once a correspondence has been found. While this process is inherently order dependent, it shows good results in a sparse setting where conflicting matches are rare. Silver and Wang compute the normalized volume difference of objects which are extracted from different input data types [6], [7], [19], [20]. Their method implicitly builds a bi-partite graph w.r.t. the overlap between feature objects from t and $t+1$. Event detection is included by comparing a feature object from t to all combinations of feature objects from $t+1$ whose overlap fulfills a given threshold. This procedure effectively enumerates the power set of all candidate objects. It is thus exponential in the number of candidates, which is small for sparse feature definitions, but quickly becomes infeasible in a space-filling setting.

Based on an approach by Sethi et al. [21], Reinders et al. formulate feature object similarity by means of abstract *attribute sets* [22]. They extrapolate a feature object's attribute set to the target time step and compare candidate objects to this extrapolation. Muelder and Ma combine the idea of extrapolation with the original overlap metric to facilitate interactive feature tracking of individual features [23].

Clyne et al. model feature similarity by means of the underlying, governing equations. They use a pathline started at the point of minimal dissipation in a feature object in t in order to identify a match in $t+1$ [24]. Sauer et al. use pre-computed particle data to extract and match feature objects from successive time steps [25].

The aforementioned approaches target volumetric features in 3D space. Other algorithms focus on the tracking of vector field singularities, e.g. *feature flow fields* [26], [27]. Garth et al. propose a method which is able to track extrema of a scalar field [28]. This would provide an alternative way to track DEs. However, tracking extrema limits the

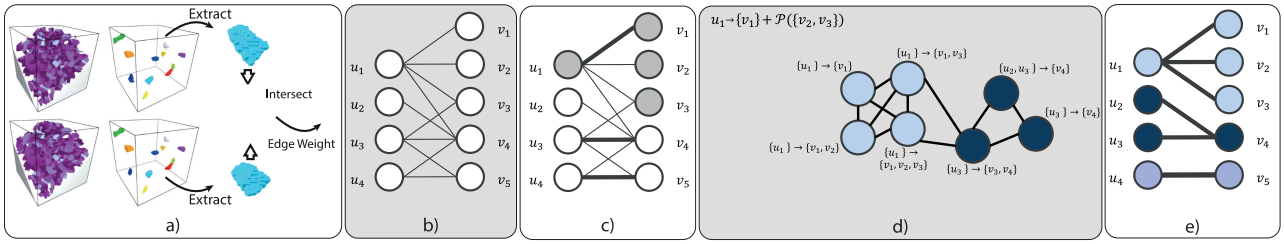


Fig. 1. Illustration of our feature tracking approach. a) Feature objects are extracted from successive time steps and overlap is computed. b) In a bi-partite graph, objects are represented by nodes; edges denote a potential correspondence; the edge weight quantifies similarity. c) The weighted, bi-partite matching gives a series of $1:1$ assignments (thick edges). The nodes considered as participating in a (split) event with u_1 (upper left) are highlighted in gray. d) For each unmatched—and hence unexplained node—a set of explanations is constructed. This results in a second graph where nodes resemble explanations and edges model mutual contradictions between these explanations. e) Solving for a maximum weight, independent set results in a set of non-conflicting explanations and thus the final evolution of feature objects in the bi-partite graph.

approach to features that can be defined by extremal points such as DEs. In contrast, we aim at a more versatile tracking approach which is able to cope with a variety of volumetric feature definitions. In doing so, we offer our collaborators the flexibility to alter the underlying feature criterion without having to re-develop large parts of the tracking methodology. Thus, we note this different option for tracking DEs but do not further investigate it.

Feature tracking enables subsequent analysis of feature properties over time. Ozer et al. detect and monitor group behavior of multiple features [29]. Subsequently, they propose to use Petri Nets for the detection of inter-feature interactions [30].

Feature tracking has been used in a number of different application scenarios. Griffith et al. use tracking to interactively analyze simulated cloud formations [31]. Doraiswamy et al. target a similar scenario, but base their approach on 2D satellite imagery [32]. Laney et al. use topological concepts to segment the interface surface in turbulent mixing processes [33]. They propose to track the resulting surface segments over time using an approach akin to the method of Samtaney et al. [2]. Bremer et al. follow a similar approach in order to analyze the flame surface in turbulent combustion simulations [34].

Recent work, which is similar in spirit to our approach since it incorporates ideas from graph optimization, has been presented by Saikia and Weinkauf [35]. They formulate the tracking problem over all T time steps in terms of a directed acyclic graph. They then use Dijkstra's single source shortest path algorithm to identify the temporal evolution of a single source feature. This track maximizes similarity over the *entire temporal* range for this *single source feature*. In contrast, our approach maximizes similarity for *all features* in *two successive* time steps.

3 Tracking by Two-Step Optimization

A common approach for feature tracking algorithms is to reduce complexity by solving the correspondence problem for two successive time steps. All candidate objects in one time step are tested for their similarity with candidate objects in a neighboring time step. From these candidates, a set of potential explanations is constructed. The goal then is to select a subset of these explanations that maximizes similarity under a given similarity metric while being conflict free. Algorithms that follow this approach usually assume that the structures under consideration cover only a small fraction of the data domain.

As stated in the introduction, naively extending this approach to a space-filling setting is infeasible, because the number of potentially matching feature objects grows substantially. The number of potential event explanations grows exponentially with the number of overlapping feature objects. Thus, solving the tracking problem exactly quickly becomes intractable in a space-filling setting. We will elaborate on this challenge in Section 4.2.

A second issue arises from the use of a *greedy* selection strategy as used, e.g., by Silver and Wang [7]: once an explanation has been selected, all participating feature objects are deleted from the search space and thus all other explanations containing them are no longer considered. In this way, the solution process is susceptible to terminate with a locally optimal solution; a potentially better solution is precluded.

To address these challenges, we propose a combination of two successively solved graph optimization problems, as illustrated in Fig. 1. In order to reduce the set of potential explanations to a tractable size, we assume that continuations explain the vast majority of features. Hence, we first try to find these $1:1$ assignments in a dedicated step. This is done by computing a maximum-weight maximum-cardinality matching on a bi-partite graph (cf. Section 3.1). The rationale behind this step is that the matching either captures a continuation directly or identifies the largest component of a split or merge event, and, hence, results in the most plausible candidate across all features in contrast to the locally optimal solution provided by a greedy strategy.

Second, we detect events by augmenting the matching by additional edges to create $1:n$ or $n:1$ relationships. To this end, we construct a graph of non-trivial, potential explanations—i.e., splits and merges—for all feature objects that have *not* been covered by the matching phase. This construction, detailed in Section 3.2, is informed by the matching, i.e., the connections defined by the matching are assumed to be part of any valid explanation. As the explanations are potentially conflicting, we compute a valid solution by formulating the problem as the search for a maximum-weight independent set in a graph representation of all possible explanations. This set of mutually non-adjacent nodes represents a selection of conflict-free explanations; the maximum-weight property ensures the global maximization of similarity. Nodes which are not assigned after these two steps are assumed to be the result of either birth or death events.

In the following we will detail the two algorithmic phases.

3.1 Weighted Bi-Partite Matching

We represent the assignment problem between two successive time steps by means of a weighted, bi-partite graph $G = (U \cup V, E, w)$ with disjoint node sets U and V, edge set $E \subseteq U \times V$ and weight function $w : E \mapsto \mathbb{R}$. We assume that distinct sets of feature objects for snapshots S_t and S_{t+1} have been generated by previous feature extraction, as depicted in Fig. 1a. Every feature object in t and $t+1$ is represented by a node in U and V, respectively.

Edges between the nodes of U and V model potential correspondences between the underlying feature objects. To this end, we compute a similarity value for every pair of nodes $(u, v) \in U \times V$. Currently, we use the *normalized volume overlap* between the two feature objects. Assuming $O_u, O_v \subseteq \Omega$ are sets of domain points which make up the respective feature objects, normalized overlap is defined by:

$$c(u, v) = \frac{|O_u \cap O_v|}{max(|O_u|, |O_v|)}. \quad (1)$$

We note that the similarity criterion can be exchanged in favor of other, possibly more sophisticated metrics, which might be more appropriate in specific application scenarios [22], [24], [25] without changing the algorithmic framework. The metric is designed to provide similarity values in $[0\,; C_{max}]$ with $C_{max} = 1$, where a higher value corresponds to more similar feature objects. For all pairs (u, v) with an overlap $c(u, v) > 0$, we insert an edge (u, v) into E and use the similarity value as edge weight $w(u, v) := c(u, v)$. For all other combinations (u, v) edges are not explicitly stored and their weight is assumed to be zero.

Fig. 1b gives a schematic example. In order to avoid a full comparison for all pairs of feature objects, we build the similarity matrix by a linear traversal of all vertices in both time steps and count the number of occurrences of object IDs u, v. From this count, we then compute $c(u, v)$ using equation 1. This method avoids an explicit representation of features.

Most matching algorithms require a square weight matrix, i.e., $|U| = |V|$. Hence, we extend the smaller set by pseudo-nodes. These are connected with every node in the other node set by a zero-weight edge, which is not explicitly stored.

The correspondence between the feature objects of the given time steps is determined by finding a maximum-weight, maximum-cardinality matching $M \subset E$ on G. This corresponds to finding a subset of edges in the graph such that each node in U and V is incident to exactly one edge and the sum of weights associated with these edges is maximized among all potential maximum-cardinality matchings. Solving the matching is equivalent to solving a linear sum assignment problem (LSAP) which is a well-studied problem (see, e.g., [36]). Any node which is matched via a zero-weight edge, i.e., assigned either with zero overlap or to a pseudo-node, is subsequently assumed to participate in an event.

We observe that—despite our assumption of having space-filling structures—the weight matrices are sparse; only a small number of features overlap and, hence, generate a non-zero entry. Given this property, we chose the pseudo-flow algorithm by Goldberg and Kennedy [37] with a complexity of $O(\sqrt{|U|}\,|E|\,log(|U|\,C_{max}))$ to solve the matching problem. The algorithm is based on the idea that an LSAP can be solved by transforming it into a flow network N and finding a maximum-weight flow in it. The transformation introduces a source node and connects it to every node in U. Analogously, a sink node is introduced and connected to every node in V. The arc set of N consists of the arcs emanating from the source, entering the sink and the edge set E of G. All arcs have a capacity of one. The arcs leaving the source or entering the sink have zero weight; all arcs representing edges of G maintain their original weights. The correspondence problem is solved by finding a flow with maximum cardinality and maximum weight. For a detailed description and analysis we refer to [37].

This first phase provides a matching $M \subset E$ which corresponds to one-to-one assignments for a maximal number of feature objects between t and $t+1$. Each matching edge $m \in M$ describes either a continuation or the largest component of a split or merge event, respectively. In Fig. 1c the matching M for the example is highlighted by thick edges.

3.2 Event Detection by Finding an Independent Set

Based on the matching M, we perform event detection in our algorithm's second phase. As outlined above, we make the simplifying assumption that matching edges will be part of the eventual solution. We now construct a set of possible explanations for each unassigned feature object by augmenting the matching edges with additional edges from E. This invalidates the matching property, because we construct $1 : n$ and $n : 1$ assignments. Our goal is to find fitting explanations for as many objects as possible by increasing the overall sum of similarity values of our solution.

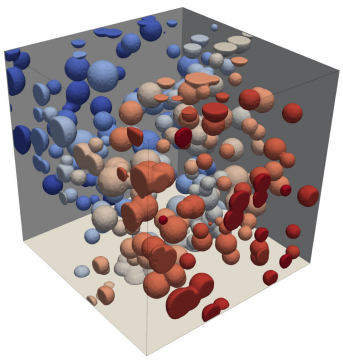
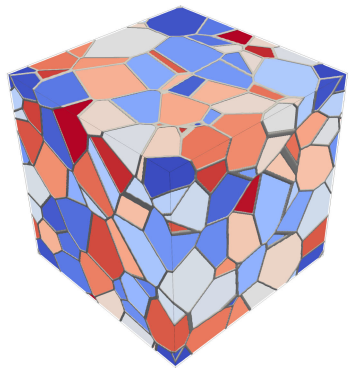
First, we construct all possible split events; these contain one matched node of U and a set of nodes from V. For each matched node $u \in U$, all currently unmatched nodes $\tilde{v} \in V$ with $(u, \tilde{v}) \in E$ and $c(u, \tilde{v}) > 0$ are enumerated. In Fig. 1d this step is illustrated for the node u_1. In this example, the nodes u_1 and v_1 are connected by a matching edge. Additional candidates that are connected to u_1 by a valid edge are v_2 and v_3. The power set of the additional candidates $\{v_2, v_3\}$ is computed and for each element of the power set, including the empty set, an event explanation is generated. An explanation consists of the two original nodes connected by the matching edge—in this case u_1 and v_1—and the nodes of one element of the power set. In the case of the empty set, the event explanation corresponds to the continuation established by the original matching; no additional nodes are added.

For u_1 in the example, we receive four possible explanations: "u_1 continues as v_1"; "u_1 splits into $\{v_1, v_2\}$"; "u_1 splits into $\{v_1, v_3\}$"; and "u_1 splits into $\{v_1, v_2, v_3\}$". Additionally, for each of these explanations, a *benefit*, in our case the union overlap normalized by the union volume, is stored, which is defined by:

$$c(u, \cup\, v_i) = \frac{\sum |O_u \cap O_{v_i}|}{max(|O_u|, \sum |O_{v_i}|)}.$$

All possible merge events containing the matched nodes $v \in V$, $u \in U$, and a set of unmatched nodes $\tilde{u}$ from U are constructed analogously. We note that constructing the events in reverse order—merges first and than splits—will not affect the results.

TABLE 1
Summary of Data Sizes and Runtime Results for the Synthetic Data Sets Used in our Evaluation

sparse sphere data set		space-filling 3D voronoi data set	
			
Dimensions	256^3	Dimensions	256^3
Size per field [MB]	64	Size per field [MB]	64
Number of time steps T	200	Number of time steps T	200
Avg. #features per TS	189	Avg. #features per TS	141
Avg. overlap test time per TS [s]	0.4702	Avg. overlap test time per TS [s]	0.6732
Avg. match time per TS [s]	< 0.0001	Avg. match time per TS [s]	< 0.0001
Avg. indep. set time per TS [s]	0.0247	Avg. indep. set time per TS [s]	0.0282
Avg. total time per TS [s]	0.4950	Avg. total time per TS [s]	0.7015

All explanations which share at least one node from U or V, respectively, are conflicting: they provide different explanations for the same feature object. Consequently, all explanations built from the same power set and expanding the same matching edge are conflicting by definition. From the set of all event explanations, we construct a second graph which contains a weighted node for each event explanation. The node weight is given by the benefit of the node's explanation. If two explanations are conflicting, the corresponding nodes are connected by an edge. Thus nodes constructed from the same matching edge form a clique as shown in Fig. 1d. To keep this graph tractable, we only include explanations with a benefit exceeding the weight of the matching edge which spawned the explanation set. This restriction ensures that the explanation by an event yields a higher similarity than a $1:1$ assignment with respect to the given similarity metric.

In addition, we further restrict the problem size with a user-controlled threshold $\hat{n}$ on the number of explanations which expand each matching edge. If there are more potential explanations, we only keep the best $\hat{n}$ explanations for subsequent optimization. Thus, for each edge $(u, v) \in M$, at most $\hat{n}$ nodes are inserted into the explanation graph. Currently, we take $\hat{n} = 2^8$ possible explanations into account. While advisable in a space-filling setting, our experiments indicate that the threshold can be omitted for sparse features.

A conflict-free selection of explanations is equivalent to a maximum independent set on this graph. In order to additionally maximize the overall benefit, we compute a maximum-weight independent set using a branch and bound strategy. In the current implementation, we formulate the independent set as an integer linear programming (ILP) problem, which we solve using the *CBC* solver of the *COIN-OR* project [38]. Finding a maximum weight independent set means finding a maximal set of nodes with a maximum overall weight, where no pair of nodes is adjacent. While this problem is NP-hard in general, the preceding matching step in combination with the restriction of the possible explanations allows us to reduce the problem to a size tractable in practice. In addition, the *CBC* solver features several heuristics for convergence detection. We currently operate with an empirically determined convergence threshold θ of 0.01, such that the weight of the independent set is within 1 percent of an estimated upper bound provided by the solver, i.e., the result is a 0.99-approximation of the optimal solution regarding the above mentioned independent set problem. In practice, we find that this step significantly improves performance at a modest cost in terms of result quality.

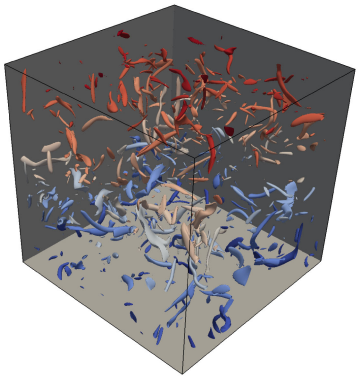
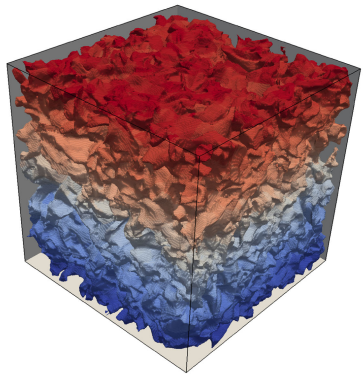
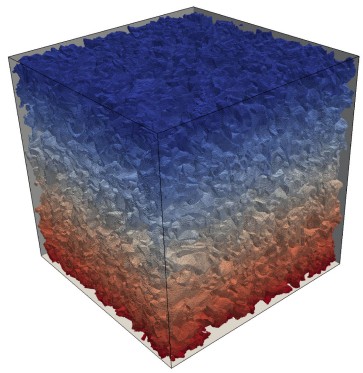
The solution is a set of nodes that are mutually unconnected. These correspond to a set of non-conflicting explanations, i.e., splits, merges, and continuations relating feature objects of two subsequent time steps. By extending M with the corresponding edges in G, we receive our final tracking solution as depicted in Fig. 1e. The resulting graph represents the evolution of all feature objects between the two time steps. All nodes that are not connected by an edge represent feature objects that could not be assigned in the neighboring time step and are thus assumed to be the result of birth and death events.

4 Results

In this section, we evaluate our approach on a variety of data sets which contain both sparse and space-filling features. For a comprehensive evaluation, we consider the tracking results of our algorithm regarding both synthetic and simulation data sets. In addition, we compare our algorithm to a previously published reference solution.

While a comprehensive, formal evaluation requires access to ground truth tracking data, for simulation data this is not available and infeasible to obtain. Therefore, we decided to create synthetic data sets including algorithmically generated

TABLE 2
Summary of Data Sizes and Runtime Results for the Experimental Data Sets Used in our Evaluation

vortices of isotropic turbulence	dissipation elements in isotropic turbulence (small)	dissipation elements in isotropic turbulence (large)
		
Dimensions 256^3	Dimensions 256^3	Dimensions 512^3
Size per field [MB] 64	Size per field [MB] 64	Size per field [MB] 512
Number of time steps T 2,000	Number of time steps T 2,000	Number of time steps T 100
Avg. #features per TS 1,135	Avg. #features per TS 28,979	Avg. #features per TS 109,893
Avg. overlap test time per TS [s] 0.0855	Avg. overlap test time per TS [s] 1.05	Avg. overlap test time per TS [s] 9.53
Avg. match time per TS [s] 0.0015	Avg. match time per TS [s] 5.82	Avg. match time per TS [s] 145.77
Avg. indep. set time per TS [s] 0.0139	Avg. indep. set time per TS [s] 91.67	Avg. indep. set time per TS [s] 1,021.17
Avg. total time per TS [s] 0.1009	Avg. total time per TS [s] 98.54	Avg. total time per TS [s] 1,176.47

ground truth results. Even though synthetic data sets do not fully reflect the realistic dynamics of the actual simulation data, they facilitate measuring the algorithm's tracking performance in a controlled setting. Specifically, we composed synthetic data sets which cover different aspects of the simulation data sets under consideration. Table 1 presents an overview along with an exemplary image of the inspected features.

Subsequently, we present tracking results for actual simulation data sets containing vortices and DEs, respectively. For all cases, features are computed in a pre-processing step. The input for the tracking consists of a scalar field per time step labeled with the unique feature object ID per grid point. Due to the lack of ground truth data, we resort to a phenomenological analysis of the resulting feature paths based on manual visual inspection. This is in line with previous work [2], [6], [23], [25]. The characteristics of the simulation data sets and an exemplary image of the inspected features are given in Table 2.

Our approach tracks and stores the evolution of all features over all time steps. In our analysis, we investigated the tracking results for a large number of features. For the sake of clarity, we selected a number of representative cases from this overall set. For each of these cases, we only show the actual features which participate in the respective events. In all depictions, we add a grid in the images to better show the features' movement trend. Please note that this grid is in no way related to the underlying simulation domain but rather included for spatial reference only. In addition, the images are annotated with a tracking graph, which depicts the evolution and events of the selected features in detail.

In order to distinguish different features over time, they are color coded by a randomly assigned color. If a feature continues in the next time step, it keeps its color. In case of a split, only the largest part keeps the color of the previous time step's object and all other features participating in the split receive a new, randomly picked color. We chose this technique due to the special interest in the tracking of space-filling structures in order to ease the distinction of elements which are close to each other. Similarly, when a feature is about to merge, the merging feature is shown in a random color in the previous time step. The final merged feature receives the color of the merge's largest component.

In order to compare our approach to an acknowledged solution, we reproduced the *volume tracking* approach by Silver and Wang [7], which has been shown to produce good results on sparse data sets, was refined in several extensions and used in multiple scenarios. This approach was chosen as baseline since it operates—like ours—on the assumption that feature objects, which make up the same time-dependent feature, generate an overlap in neighboring time steps. Thus, the structured reasoning about differences between both approaches is not based on the specific metric but can rather be attributed to the assignment strategy used by the respective algorithm. Silver and Wang use a threshold value for assignments regarding the similarity of features which is based on a volume difference test. For a valid assignment, the similarity value of two features has to be below the threshold. Based on the recommendation in their work [7], the threshold value was set to 0.55, which is in the suggested range between 0.5 and 0.6 for the investigated sparse data sets. The first step of the reference algorithm is to find all pairs of overlapping feature objects in successive time steps and to compute their similarity value. The assignment is then done by first testing for continuations and split events. Afterwards, a test for merges is performed. The best assignment is chosen greedily and all objects belonging to identified events are removed from the search space. Thus, if objects are used for a split, they are no longer investigated in the subsequent search for merges.

After evaluating tracking performance for synthetic and simulation data in Sections 4.1 and 4.2, respectively, we discuss systematic effects due to the different assignment

TABLE 3
Ground Truth and Tracking Results for our Method and the Volume Tracking Approach [7] Regarding the Individual Events and the Sum of all Events within the Sphere Data Set

		Our two-stage tracking			Volume Tracking [7]; $T = 0.55$		
Event Type	#Events Ground Truth	#Detected Events	#Correctly Detected	% Correctly Detected	#Detected Events	#Correctly Detected	% Correctly Detected
Continuation	32,272	32,258	32,186	**99.73**	29,240	27,897	86.44
Split	1,744	1,713	1,696	**97.25**	1,689	1,638	93.92
Merge	1,926	1,979	1,847	**95.90**	619	554	28.76
Birth	810	919	801	**98.89**	5,273	797	98.40
Death	577	717	560	97.05	6,500	575	**99.65**
Sum	37,329	37,586	37,090	**99.36**	43,321	31,461	84.28

schemes of our approach and the volume tracking and the respective outcomes in Section 4.3. We conclude this section with a brief discussion of our approach's runtime behavior in Section 4.4.

4.1 Synthetic Data Sets

As mentioned above, simulation data sets typically do not feature ground truth data against which to evaluate a tracking solution. However, this would be necessary in order to quantify the accuracy of a tracking algorithm. In addition, a comparison against ground truth data enables us to reason about specific algorithmic mishaps in a structured manner. Hence, we composed two types of synthetic data sets and the corresponding ground truth results including all relevant temporal developments, i.e., continuations, splits, merges, births, and deaths (cf. [2]). The objects, the temporal evolution, and the predefined events in these ground truth data sets are generated to exhibit dynamics similar to certain aspects of simulation data sets. The ground truth data is represented by a tracking graph similar to the output of our tracking algorithm.

In order to assess the tracking performance of our approach, we tracked the synthetic data sets using our method and the reference algorithm. We then compared the respective results to the ground truth and the approaches to each other regarding their accuracy w.r.t. the ground truth. In the following sections, we discuss both data generation and tracking results in detail.

4.1.1 Sparse Sphere Data Set

The first synthetic data set represents a sparse setting containing randomly distributed objects—spheres in this case. Time variance is simulated by a temporal displacement of the spheres inside the volume. To this end, the spheres move independently of each other across the domain. Feature objects in this data set consist of connected components of mutually intersecting spheres. Splits and merges result from connected components splitting up or amalgamating due to the movement of the constituent spheres. For birth and death events, spheres either leave and enter the domain, new spheres are added in empty regions, or single non-connected spheres are deleted.

Each object in the data set is moved along the local velocity field of a direct numerical simulation (DNS) of homogeneous, isotropic turbulence similar to the simulation data sets used later in our evaluation. Each connected component is considered as a single feature object which is assigned its own, unique ID. When a connected component contains the same defining spheres after displacement, we record a continuation. If objects overlap or break apart due to the displacement of their defining spheres, a merge or split event is stored, respectively. To enable death and birth events, a boundary layer of ghost cells is stored which surrounds the actual data domain. If objects leave the actual data domain, a death event is stored. Re-entering objects appear as birth event in the ground truth result. In addition, birth events in the inner parts of the data domain are enabled by finding empty regions and seeding new spheres with a small radius, which is increased in the following few time steps. Analogously, death events are integrated by searching for single, non-connected spheres whose radius is reduced until they disappear. Additionally, the combination of a merge and a split event including the same sphere in the same time step is excluded since it does not reflect the dynamics of simulation data sets for a sufficiently high temporal resolution. This is done by splitting the corresponding connected components artificially in a second iteration by assigning different IDs to the constituent spheres.

Table 1 (left) shows an overview of a single time step and the characteristics of this data set. In this case, the feature objects are sparse with respect to the domain, i.e., most of the domain is classified as *background* which does not contain feature information.

In order to measure the algorithm's tracking performance for this scenario, 200 time steps of the sphere data set were tracked with both algorithms. Table 3 reports the results with respect to the ground truth data. For both approaches the absolute number of detected events, the number of correctly detected events, and the percentage of correctly detected events are listed per event type. In addition, the sum of all events is listed.

Both approaches provide a correct tracking result for more than 90 percent of all splits, births, and deaths, respectively; only the detection of continuations and merges differs notably. In case of the continuations the gap between our approach and the reference algorithm is more than 10 percent. For merges, our two-step tracking approach substantially outperforms the reference algorithm by correctly identifying 95.90 percent of all merges compared to 28.76 percent. The tracking results for the other data sets, which will be discussed in the following sections, reveal similar behavior. This suggests a systematic effect related to the tracking algorithms rather than a data-dependent observation. Hence, we will discuss this point in more detail in Section 4.3.

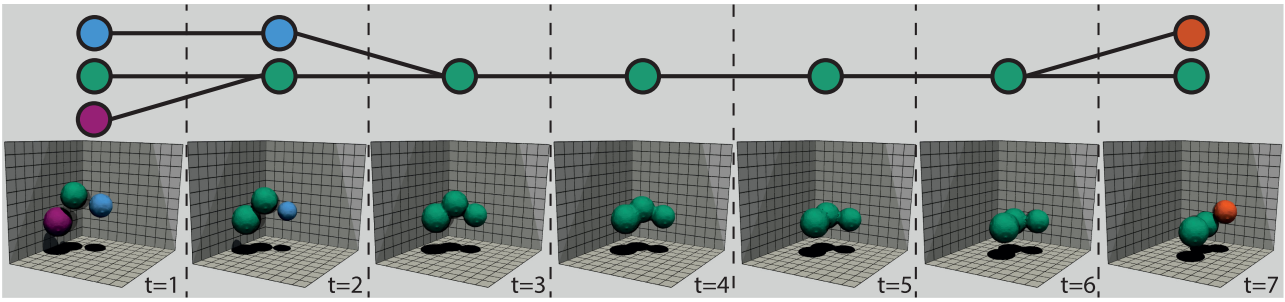


Fig. 2. Depiction of the temporal evolution of features within the synthetic sphere data set (from left to right, bottom row) and the corresponding tracking graph (top row) tracked with our algorithm. The selected features consist of three non-overlapping spheres, which merge together from first to third time step and split into two features from the sixth to seventh time step due to the spheres' displacement in different directions over time. The grid is added as a reference and does not resemble the outline of the simulation domain.

In addition to this statistical analysis, we inspected several resulting temporal evolutions of features in this data set for our algorithm. This included both evolutions that are in line with the ground truth as well as evolutions that differ from it. Here, we focus on the former, deferring the latter to Section 4.3, too. Fig. 2 shows an exemplary resulting temporal evolution of a feature within the sphere data set (bottom row, from left to right) and the corresponding tracking graph (top row) which is achieved by tracking with our algorithm. The selected features in this case are three individual, non-overlapping spheres, which are considered as three single feature objects. Since each sphere in the data set moves independently, two of the spheres (green and purple) are considered a single feature object in the second time step due to their overlap, which our tracking algorithm correctly detects as a merge event. An additional edge connecting the purple feature object in the first time step and the green one in the second time step is inserted into the tracking graph. In the third time step the blue and the green feature object overlap due to their displacement in different directions, which is again correctly detected as a merge by our algorithm. Due to the independent movement of the spheres, the overall shape of the resulting feature object changes in the following time steps and a general movement trend can be seen. In the seventh time step one of the spheres (orange) breaks apart again from the cluster, which our tracking algorithm correctly detects as a split event. Taken together, the quantitative results and the visual inspection suggest that our algorithm provides reliable tracking output for this data.

4.1.2 Space-Filling Voronoi Data Set

The second synthetic data set is defined by the Voronoi diagram of a randomly distributed 3D point set. Each Voronoi cell defines a single feature, resulting in a space-filling data set due to the definition of Voronoi diagrams (cf. Table 1 (right)). Each seed point is moved across the domain by a single translation vector to simulate time variance. We use periodic boundary conditions, i.e., seed points leave the domain and re-enter on the opposite side. In order to generate merge and split events, randomly selected seed points are moved in addition to the overall movement of all seed points. Merge events are generated by moving a randomly selected seed point and its nearest neighbor towards each other and merging them into a single seed point once their distance falls below a certain threshold. Analogously, split events are inserted by duplicating a seed point on its position and moving the original and the newly seeded point in opposite directions in the following few time steps in addition to their overall movement. Split and merge events are restricted to $1:2$ and $2:1$ assignments. Note that the Voronoi ground truth data set does not include birth and death events. The rationale behind this assumption is that due to the space-filling setting there is no background in the data set, which is a necessary precondition for births and deaths. Thus, every newly formed feature covers parts of the volume of feature objects in the preceding time step and, hence, has to be part of a split event. Analogously, the volume of disappearing features is covered by other feature objects in the following time step. Therefore, the feature has to be part of a merge. This issue was brought up by our collaborators from fluid mechanics who initially suggested this specific interpretation. Table 1 (right) shows an overview of a single time step and the characteristics of this data set. We followed the same procedure as discussed in Section 4.1.1, tracking 200 time steps and comparing the outcome for both algorithms to the ground truth. The results are listed in Table 4.

Initial experiments suggested that the performance of the volume tracking approach for the Voronoi data set is—in contrast to the sphere data set—quite sensitive to the similarity threshold setting. Hence, we report results for the volume tracking for the thresholds 0.55 and 0.3, where the second value was chosen to specifically enhance the result for the merge detection in a space-filling setting, despite the value not being in the suggested range of 0.5 to 0.6. The three columns on the right of Table 4 show the results for a threshold of 0.3.

Again, both approaches—ours and the volume tracking with a threshold of 0.55—provide correct detections of more than 90 percent for continuations, splits, and the overall number of events. Furthermore, the number of erroneously detected birth and death events is low. Similar to the sphere data set, the major difference is in the correct detection of merge events. While the rate of correctly detected merges is only about 25 percent for the volume tracking, our approach correctly detects approximately 95 percent.

Due to the fact that the ground truth is known for this data set, we were able to hand-tune the similarity threshold of the baseline algorithm in order to facilitate a more meaningful comparison of both approaches. The initial experiments revealed that a threshold of 0.3 provides a notably higher percentage of correctly detected merges. This, however, comes at the cost of a notable drop in the overall

TABLE 4
Ground Truth and Tracking Results for our Method and the Volume Tracking Approach [7] for the Voronoi Data Set

Event Type	#Events Ground Truth	Our two-stage tracking			Volume Tracking [7]; $T = 0.55$			Volume Tracking [7]; $T = 0.30$		
		#Detected Events	#Correctly Detected	% Correctly Detected	#Detected Events	#Correctly Detected	% Correctly Detected	#Detected Events	#Correctly Detected	% Correctly Detected
Continuation	27,304	27,336	27,300	**99.99**	27,344	27,210	99.66	23,796	23,788	87.12
Split	293	279	275	93.86	322	284	**96.93**	247	230	78.50
Merge	186	177	177	**95.16**	47	47	25.27	162	160	86.02
Birth	0	3	0	-	40	0	-	3498	0	-
Death	0	0	0	-	195	0	-	3602	0	-
Sum	27,783	27,795	27,752	**99.89**	27,948	27,541	99.13	31,305	24,178	87.02

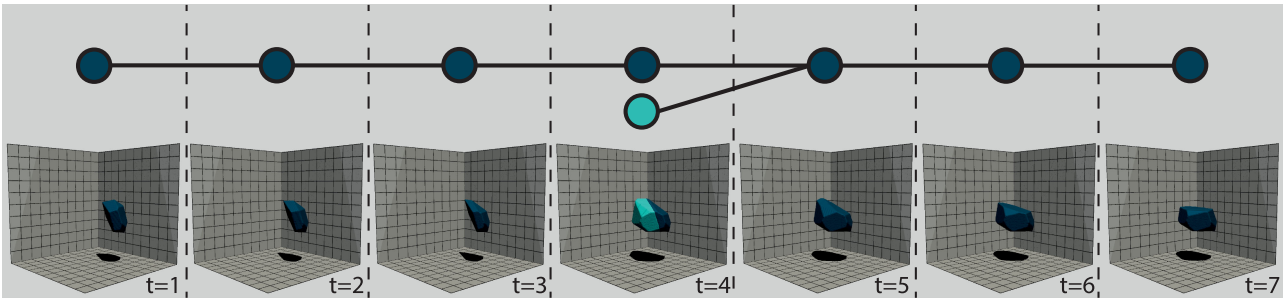


Fig. 3. Depiction of a merge event within the Voronoi case and the corresponding tracking graph. The dark blue feature is tracked over several time steps. Just before the neighboring Voronoi cell (light blue) is about to merge in the fourth time step, it is shown in addition. The final merged feature is shown in dark blue. Note that volume and shape of the merged feature are similar to the two Voronoi cells in the former time step.

tracking accuracy. Furthermore, the lowering of the threshold causes a substantial number of erroneous birth and death events. Despite the merge detection rate for the volume tracking being notably higher, our algorithm outperforms the baseline for the lower threshold regarding all types of evolutions.

Similar to the sphere data set, we inspected a couple of resulting temporal evolutions. Fig. 3 shows an example tracking result of our approach for a single Voronoi cell and the corresponding tracking graph which is in line with the ground truth information: the dark blue feature continues in the first three time steps; in the fourth time step the neighboring, light blue cell, which is about to merge with the dark blue one, is depicted in addition to the dark one; in the fifth time step the objects are merged and continue as a single feature in the following time steps. In summary, this evaluation demonstrates that our algorithm also performs well in a space-filling setting.

4.2 Simulation Data Sets

To evaluate our approach in a real world setting, we analyze data sets from turbulence research by manual visual inspection w.r.t. the plausibility of the tracking results achieved by our algorithm. Analogously to the synthetic data sets, we consider examples for sparse as well as space-filling features. They all result from DNS with a high spatio-temporal resolution; all are simulations of homogeneous, isotropic turbulence inside a periodic box. In the first data set, we analyze the temporal evolution of vortices. The two other data sets both contain extracted DEs as features, but they differ in data size and complexity.

Table 2 (left) shows an overview of a single time step of the vortex data set. Vortices have been extracted by thresholding for high vorticity magnitude. They are sparse with respect to the data domain and thus most of the domain is classified as background. Similar settings are used for evaluating tracking algorithms, e.g., [2], [6], [7], [22]. Consequently, we include this data set to enable comparison with established techniques.

The two other simulation data sets containing DEs directly result from our collaborators' research. They are included to assess the tracking capabilities of our algorithm for space-filling features. DEs describe the geometric structure of small-scale turbulence. They arise as natural geometries in turbulent scalar fields [3]. They are extracted from an instantaneous scalar field, which is locally smoothed by diffusion, e.g. temperature, instantaneous kinetic energy, or its dissipation. The extraction is performed with a validated code provided by our collaborators [39]. As stated before, DEs are closely related to 3D Morse Smale cells [16]. We note that our tracking is oblivious to the method of extraction.

The data sets have a spatial resolution of 256^3 voxels and 512^3 voxels, respectively. For both data sets, every tenth simulation time step has been stored to disk, which is a relatively high temporal resolution given today's I/O constraints. Table 2 shows an overview of a single time step and the characteristics of these data sets. Except for a small boundary region, every data point is assigned to one dissipation element. Thus, the features are actually space-filling as required by definition.

In order to demonstrate the increased complexity of feature tracking in a space-filling setting, we analyze the structure of the matching graphs over the entire temporal domain. Because this graph contains an edge for every overlapping pair of feature objects, its density serves as an indicator for the difficulty of the underlying correspondence

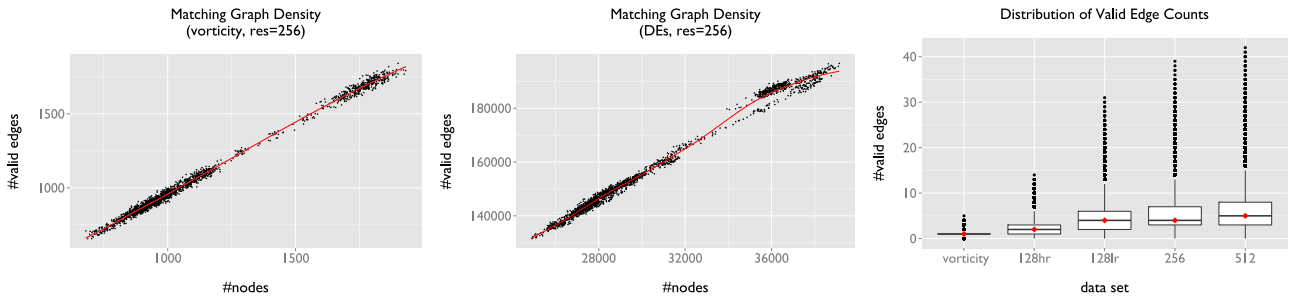


Fig. 4. Graph density for the 256^3 vortex (left) and the 256^3 DE data set (middle) and the distribution of valid edges per feature object for all simulation data sets (right). All plots cover the distributions over the entire temporal domain. In the two density plots, each point represents a single matching instance, i.e., the connectivity for one pair of consecutive time steps. The box plot aggregates the number of overlapping candidates over all feature objects in all time steps. The two graph density plots including a smoothed trend line in red suggest that the number of edges is roughly proportional to the number of nodes. Note the different scaling of the axes and the different proportionality factor for the graph density. Despite the apparently linear relationship, the bar chart on the right shows that the number of valid edges for each feature as well as the number of outliers and their connectivity is substantially higher in the space-filling DE data sets.

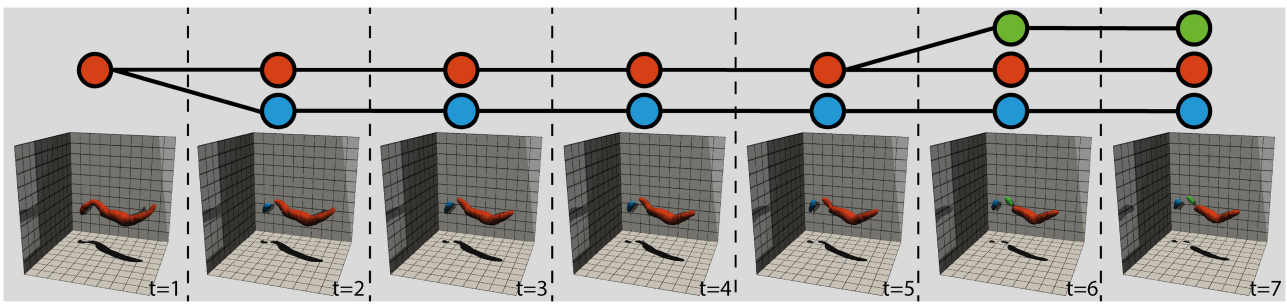


Fig. 5. Closeup of the evolution of a single, connected vortex in the isotropic turbulence data set. The vortex splits twice into a larger and a smaller component over time.

problem. The number of possible explanations—including continuations, splits, and merges—includes the power set of all overlapping candidates: a node u in the matching graph with a degree of d_u gives rise to 2^{d_u} possible explanations.

Fig. 4 shows the matching graph density for the 256^3 vortex and the 256^3 DE data set (left and center) as well as the distribution of valid edge counts (right), i.e., the aggregated node degrees in the matching graph. In addition to the vortex data set and the two DE data sets, which are discussed in detail, we include two DE data sets with a spatial resolution of 128^3. One case (128lr) has the same temporal resolution as the two larger cases, i.e., every tenth time step has been stored. For 128hr every simulation step has been written out. All given plots cover the entire temporal domain. The two density plots show the number of edges over the number of feature objects in the originating time step for all $T-1$ matching instances; each point represents a single matching instance and, hence, the connectivity for one pair of consecutive time steps. The box plot aggregates the number of overlapping candidates per feature object over all feature objects and all time steps.

The two density plots, which include a smoothed trend line shown in red, suggest a proportional relationship between the overall number of nodes per time step and the resulting number of edges, regardless of sparse or space-filling data. Thus, the average number of overlapping candidates per feature object is roughly constant across the entire temporal domain. Fig. 4 (right) shows that this number is substantially lower for the sparse setting compared to the space-filling setting. The median of the sparse vortex case is 1: on average, there is only a single candidate to choose from. In contrast, the medians for the four DE cases are 2, 4, 4, and 5 for the $128hr$, $128lr$, 256, and 512 cases, respectively. Hence, even in the average case, represented by the median, there are more candidates to choose from. More importantly, the number of outliers and the connectivity of these outliers increase for a space-filling setting and for increasing spatial resolution. A higher temporal sampling reduces the number of potential candidates, however, the median is still higher than in a sparse setting. An exhaustive search for the best event explanation requires an optimization over the power set of all candidates. With outliers as high as 42 in the 512^3 case, this quickly becomes prohibitive. We note that these observations are solely based on the underlying data and do not rely on any assumptions related to our algorithm. This supports our initial claim that feature tracking in a space-filling setting is significantly more complicated.

In addition, this investigation supports the chosen threshold on the number of possible event explanations which is currently set to $\tilde{n} = 2^8$ (see Section 3.2). While we prune the explanation set for a number of outliers, including only the $\tilde{n}$ best choices leads to good solutions in practice. Additionally, setting $\tilde{n} = 2^8$ is sufficient to cover the main body of the distribution of valid edges even in case of the 512^3 data set.

4.2.1 *Vortices in Isotropic Turbulence*

For our evaluation, we picked out a number of vortices and inspected their evolution for plausibility, which is in line with evaluations in previous work [2], [6], [23], [25]. In Fig. 5, we show an exemplary tracking result of our algorithm of a single vortex. In the first time step, a single,

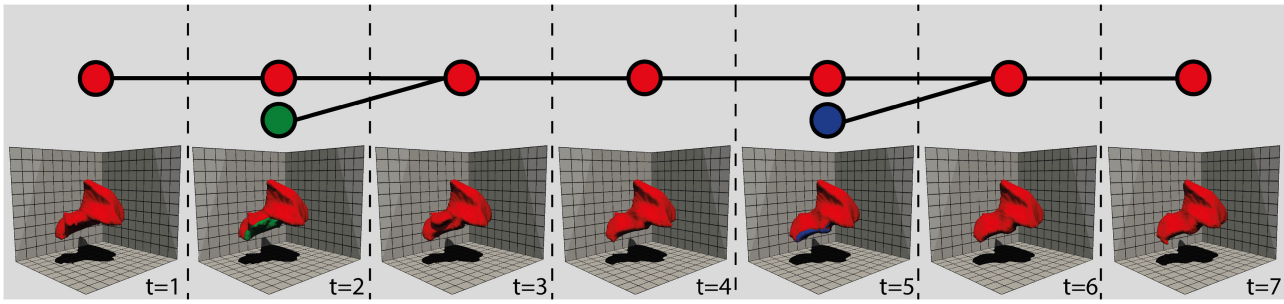


Fig. 6. Illustration of a larger DE (shown in red) which merges twice with smaller elements (green and blue) over time. The two features which are about to merge with the red one are shown in a different color in the time step before the merge; the final merged DE is shown in red.

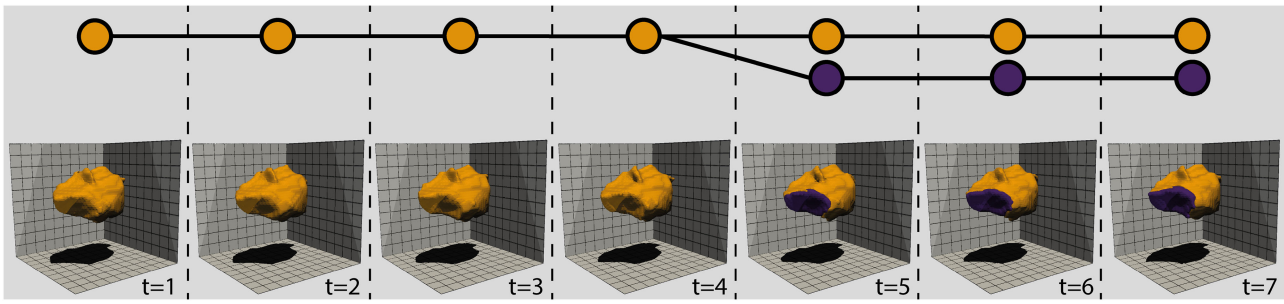


Fig. 7. Depiction of the evolution of a single DE which is involved in a split event in the fifth time step. Both resulting DEs evolve independently after the split, even though they do not move apart due to the space-filling property.

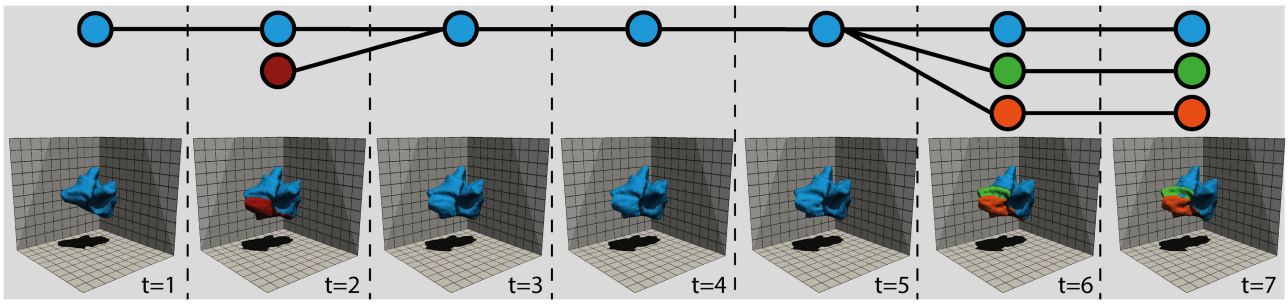


Fig. 8. Closeup of the evolution of a DE which is first involved in a merge event, followed by a three-way split into one larger and two smaller DEs.

connected vortex is extracted. As shown in the tracking graph, the vortex splits into two structures in the second time step—the larger red vortex and a smaller blue one in the left portion of the image. The two vortices evolve independently in the following time steps and the larger one splits again into two structures in the sixth time step. This setting is demonstrative for a number of similar situations that we evaluated. In conclusion, our algorithm provides plausible results for the temporal evolutions in this data set.

Prompted by the substantial differences observed in Section 4.1, we performed a similar comparison between our algorithm and the reference algorithm focusing on differences regarding the detection of merges. As stated above, we found evidence for a systematic issue and, hence, discuss this aspect in detail in Section 4.3.

4.2.2 Dissipation Elements

Similar to the vortex data set, we picked out a number of structures and their evolution for plausibility inspection for the two data sets containing DEs. In the following, three exemplary evolutions with different types of events identified by our algorithm are described. The first two examples are taken from the 512^3 case while the last one is taken from the smaller 256^3 case.

Fig. 6 illustrates the evolution of a larger (red) DE which merges with two smaller DEs in the depicted time steps, shown in green and blue in the time steps just prior to the merge with the red one. Note the slight growth in overall volume for the red feature resulting from the merge events.

In contrast, Fig. 7 shows the evolution and the corresponding tracking graph of a larger DE which undergoes a split event. While the element's overall shape remains stable for the first four time steps, the purple component splits apart in the fifth time step and both DEs evolve independently. Note that the elements do not move apart due to the space-filling setting.

Finally, Fig. 8 depicts the evolution of a feature involving a merge and a split event. In the second time step, the red feature object which is about to merge with the selected one (shown in blue) is depicted additionally. Note again the growth in volume after the merge. The feature splits up in the sixth time step into one larger and two smaller components. The dense packing of the resulting structures illustrates the challenges arising with the tracking of space-filling structures.

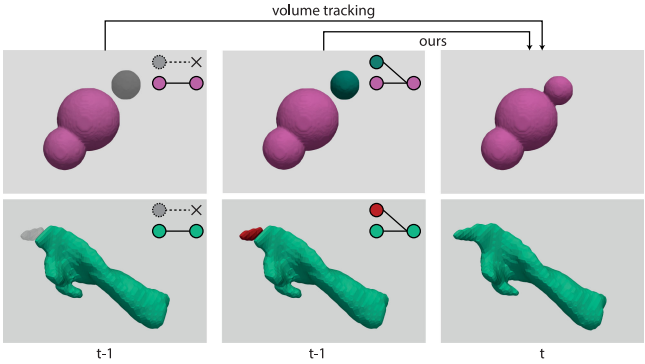


Fig. 9. Erroneous merge detection by the volume tracking approach in the sphere (top row) and the vortex data set (bottom row). Depiction of the features within the former time step for the volume tracking (left) and our approach (middle) and the subsequent time step (right). The volume tracking approach labels the event as a combination of a continuation and a death, while ours correctly detects a merge event.

The plausibility of these results has been backed by our collaborators. In conclusion, we argue that our approach covers all categories of feature evolutions within a space-filling setting.

4.3 Observations

During the comparison to the ground truth for the synthetic data sets and the visual inspection of the tracking results for the simulation data sets, we found substantial differences in the detection of merges between our approach and the volume tracking algorithm. As this suggests a systematic effect rather than a data-dependent observation, we further investigated the affected merges.

Fig. 9 shows two examples for such erroneous merge detections within the sphere data set (upper row) and the vortex data set (bottom row), respectively. The left image shows the two involved features in the time step just prior to the merge event for the volume tracking approach and the corresponding tracking graph. The middle image shows the same time step and the corresponding tracking graph for our approach. The right image depicts the subsequent time step where the two objects have merged.

For the case of the sphere data set, the volume tracking identifies a continuation for the purple feature; the evolution of the grayish feature is classified as death. Our approach establishes the $1:1$ assignment of the purple elements during the matching and then extends this solution into a merge by adding the dark green sphere to the event which matches the ground truth information. Differences like this offer an explanation for the observed merge detection rates of both algorithms given in Tables 3 and 4.

Since we found this phenomenon in both synthetic data sets, we examined the underlying scenarios, for which the two algorithms disagree, in more detail. We found that mostly asymmetrical merges are affected, i.e., merges where the participating objects notably differ in terms of their volume. Hence, we searched for cases that fit this description in the vortex data set. To this end, we compared the tracking results of both approaches similar to the comparison to the ground truth and inspected the different evolutions found by the approaches regarding their plausibility.

An example is shown in Fig. 9 (bottom row). As in the images for the sphere data set, the left image shows the

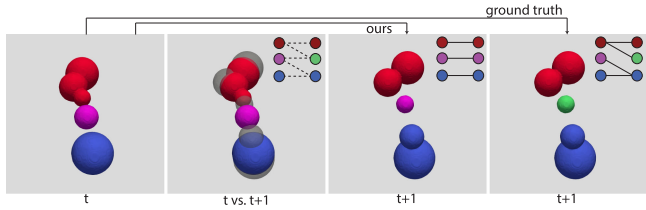


Fig. 10. Erroneous detection of three continuations instead of a merge and a split event close to each other in the sphere data set by our algorithm. The images show the features within the former time step, the overlap of the features of both time steps, the features in the subsequent time step for our approach, and the ground truth result for the subsequent time step (left to right).

feature objects in the former time step and the corresponding tracking graph for the volume tracking approach, the middle image shows the feature objects in the same time step for our approach, and the right image shows the merged features in the subsequent time step. The volume tracking establishes a continuation for the green feature in the former time step resulting in the green feature in the subsequent one and a death event for the small gray feature object. Our approach augments the $1:1$ assignment of the green objects to a merge of the green and the red object. This explanation is arguably more plausible w.r.t. shape, volume, and position of the features which are involved.

We hypothesize that the issue of missing parts in asymmetric merge events results from an order dependence within the volume tracking approach. It first searches for continuations and splits, and removes all objects involved in these events from the search space. The search for merges is executed afterwards, but is limited to the remaining feature objects. Due to the greedy assignment, the larger component of the merge event alone creates a seemingly valid continuation which fulfills the threshold and is, hence, established in the first step; the corresponding feature objects are no longer investigated in the subsequent merge detection.

In contrast, our approach first focuses on $1:1$ assignments by the matching in the first phase because of the assumption that continuations make up the vast majority of evolutions. As said in Section 3.2, these $1:1$ assignments either represent a continuation or the largest part of a split or merge event, which are established by augmenting the $1:1$ assignments. Subsequently, our algorithm looks for splits and merges symmetrically and performs a global optimization. Thus, the algorithm is not biased towards either splits or merges.

However, our approach remains predisposed in favor of continuations because of the matching in the first phase (cf. Section 3). If an erroneous continuation is established once, the associated features are no longer available for augmenting other matching edges.

An example for such a case is depicted in Fig. 10. The leftmost image shows the features in the former time step. The middle left image shows the overlap of the two investigated time steps and the corresponding bi-partite overlap graph, which contains all valid edges for the considered feature objects indicated by dashed lines. The middle right image shows the tracking outcome, which is achieved by our tracking approach, and the image on the far right shows the ground truth result.

The ground truth result in this case is a merge event of the pink and the blue spheres in the first time step, which then

form the blue one in the second time step, and a simultaneous split of the red object into the red and the light green ones. In contrast, our approach establishes three continuations, since the two pink-colored feature objects overlap and, hence, increase the matching weight in the first phase of the tracking. Due to the fact that matching edges are not reconsidered during event detection, the two other matching edges between the red objects and the blue objects cannot be augmented to the actual events because the other objects are no longer unmatched. We note, however, that such cases are caused by a very specific spatial configuration and, hence, rather rare. A split and a merge event have to happen in immediate vicinity of each other. In addition, one of the objects involved in the split and one involved in the merge event have to overlap each other. This is a pathological configuration with respect to our algorithm, because adding the third matching edge will always enhance the matching weight and will lead the algorithm to erroneously assume a continuation.

However, as the results for the synthetic data sets revealed, the amount of falsely detected continuations is arguably rather small. Furthermore, referring to previous work by experts [7] and domain scientists [3], the same can be expected to hold for the simulation data sets as in these continuations make up the majority of explanations. In summary, our algorithm provides reliable results in most instances. Finally, we would like to point out that this sort of structured reasoning about falsely detected events was only enabled by the availability of ground truth data.

4.4 Runtime Performance

In addition to the tracking performance, we investigated the runtime behavior of our approach for all aforementioned data sets. To this end, we measured the runtimes for the problem setup and the graph optimization problems for all time steps that were tracked.

All measurements have been performed on a dual socket server featuring two IntelXEON E5-2695 v3 CPUs (28 cores @ 2.3 GHz each) and 512 GB of RAM. Since the algorithm does not (yet) run in parallel, it does not benefit from the available cores.

Results for the synthetic data are included in Table 1. On the given hardware, a full tracking step—including graph setup, matching and event detection—for the sphere data set takes on average $0.5\,s$. The matching step takes about 0.02 percent and the independent set 4.99 percent of the time on average for this data set. For the Voronoi data set a full step takes $0.7\,s$. The matching step takes only about 0.01 percent on average and the independent set accounts 4.02 percent of the runtime. In both cases, most time is consumed by the overlap tests, which takes 94.99 and 95.97 percent of the time on average, respectively.

All average runtimes for the simulation data sets are listed in Table 2. A full tracking step for the vortex data sets takes on average $0.1\,s$. As in the synthetic data set, most of this time is consumed by the overlap tests (84.74 percent) whereas the matching step accounts for 1.48 percent and the independent set for 13.78 percent of the time for this data set. In contrast to that, the overall runtimes for the space-filling DE data sets are $98.54\,s$ and $1{,}176.47\,s$ for the 256 and 512 case, respectively. In both cases, most of the time is consumed by solving the independent set problem, which is about 93.03 and 86.80 percent. This obvious increase in runtime for the independent set stage compared to the other steps results from the exponential growth of the graph, which is considered for the independent set, w.r.t. the number of features.

We include these numbers mainly as an indicator for the absolute overall runtime and the relative cost of the individual steps. Moreover, they serve to illustrate that the independent set stage of our algorithm—while accounting for an increasing portion of the overall runtime—still remains within feasible bounds even for the 512^3 data set. As the number of candidate features—and with it the number of potential explanations—grows rather gently (cf. Fig. 4), we believe that this trend will hold for even larger data sets. Finally, we note that the current implementation has not yet been optimized for runtime efficiency. In particular, it is not running in parallel, a step that we plan for future work.

5 Discussion

The main goal of the method introduced in this paper has been to enable the tracking of space-filling structures. This has been motivated by the analysis of dissipation elements [3]. Overall, the results presented in the previous section demonstrate that our algorithm is able to track the temporal evolution of features in general and to identify the canonical classes of events, i.e., splits, merges, births, and deaths. In this section, we briefly discuss a few limitations of our approach and its current implementation.

One central aspect for tracking in general is the choice of a similarity criterion. In our approach, in both phases of tracking, i.e., matching and event detection, we currently use normalized volume overlap: we normalize the common volume of two feature objects from neighboring time steps by the volume of the larger feature object. Despite the normalization, this results in a bias towards larger feature objects, which may introduce a systematic error. For example, assume a situation where a small feature object u_1 at time t is completely included in a larger feature object v in $t+1$ and a large feature object u_2 from t overlaps partly with v. If $c(u_2, v) > c(u_1, v)$, i.e., u_2 overlaps a larger fraction of v than u_1, the matching is likely to choose (u_2, v) to maximize the overall weight regardless of the fact that v might be regarded as a natural explanation for u_1. These issues notwithstanding, we note that our algorithm is oblivious to a switch of the underlying similarity metric. We thus plan to investigate different similarity criteria in future work, e.g., based on different attributes of the features of interest [22] or on graph similarity [40].

Another focus for future work relates to a detailed investigation of *temporal stability*, i.e., the dependency of tracking performance on the temporal sampling rate of the data. For an overlap-based tracking approach, the temporal resolution has to be rather high in order to ensure that the feature objects do not move too far between time steps. If no overlap is generated, existing correspondences will be missed. Increased temporal resolution generally implies smaller changes between subsequent time steps, which in turn drastically reduces the search space during event detection, as indicated by the graph connectivity numbers for the two 128^3 DE cases shown in Fig. 4 which differ in their temporal

resolution. Integrating the feature extraction and the tracking stages with the simulation in order to run both in an in situ environment may be the only feasible option to provide high enough temporal resolution in some cases. A resulting reduction in candidate feature objects would also allow us to ease the artificial cutoff of our approach or omit it entirely. We note, however, that the question of sufficient temporal resolution is intrinsically tied to tracking approaches in general and not specific to our method. In particular, an insufficient temporal resolution will render the tracking problem ill-posed and thus not solvable at all.

Another issue, which generally applies to all automatic feature tracking approaches, is systematic evaluation. Tracking correctness is very hard to measure, especially in a setting where even small data sets contain tens of thousands of features. To aggravate the situation, for real-world data there is generally no ground truth information with which to compare tracking outputs automatically. At the same time, tests on synthetic data have only limited expressiveness since it is hard to cover all aspects of the realistic behavior of fluid dynamics. Hence, we believe that the best effort to date remains a thorough visual inspection of the resulting temporal evolutions of tracked features by domain scientists in addition to a test on synthetic data sets with a known temporal evolution which exhibits as similar dynamics as possible. In this regard, we plan to investigate additional options for a formal evaluation of complex tracking approaches.

Finally, we would like to increase the workflow's overall performance. Currently, the matching algorithm as well as the independent set solver are not running in parallel. Moreover, a custom solver for the independent set problem which is particularly designed to resolve the independent set on a graph containing large numbers of cliques like in our approach's event detection should be able to speed up the tracking. We will address these aspects in order to scale the tracking to larger cases, particularly large-scale, DNS-based analyses of dissipation element dynamics.

6 Conclusion and Future Work

In this work, we have presented a novel approach for feature tracking which generalizes to space-filling features. Our approach formalizes the tracking problem by means of two successively solved graph-theoretical optimization problems based on feature similarity. Continuations, i.e., one-to-one assignments, are found using a weighted, bi-partite matching. This matching is subsequently augmented by events, i.e., splits and merges, by computing a maximum-weight independent set. This graph-theoretical formulation allows us to drastically reduce the search space and thus to extend our algorithm from sparse features to densely packed or space-filling ones. We have demonstrated this capability by testing our approach on a variety of simulation data sets containing sparse features—vortices—as well as space-filling structures—dissipation elements. Visual inspection supported by our collaborators confirmed the plausibility of our tracking results. Additionally, in an effort to enable a more formal evaluation, we have tested our algorithm with synthetic data sets containing features with known spatio-temporal evolution. Results were on par with or better than the reproduced reference algorithm.

Several aspects for future improvement and extension have been discussed in the previous section. In particular, we plan to implement and assess more sophisticated similarity metrics for tracking dissipation elements as well as other space-filling features more robustly. In addition, we intend to increase the tracking algorithm's efficiency and practical usability by parallelization. Finally, we strive to integrate the simulation, the feature extraction, and the feature tracking phases to operate in situ in order to address memory and I/O constraints.

In summary, the feature tracking approach proposed in this work enables fluid mechanics researchers to monitor and assess the temporal evolution and thus the time-dependent characteristics of arbitrary volume features, even when they are space-filling.

Acknowledgments

This work received support from the DFG under KU 1132/10-1 and the German Excellence Initiative via JARA–HPC. In addition, we thank Sebastian Freitag, Claudia Hänel, Ingrid Hotz, Heike Leitte, Tom Vierjahn, and Benni Weyers for their valuable input to this work. Finally, we thank the supervising members of the 2017 IEEE VIS Doctoral Colloquium, Hank Childs, Kelly Gaither, and Ken Moreland for their constructive feedback.

References

[1] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch, "The state of the art in flow visualisation: Feature extraction and tracking," *Comput. Graph. Forum*, vol. 22, no. 4, pp. 775–792, 2003.

[2] R. Samtaney, D. Silver, N. Zabusky, and J. Cao, "Visualizing features and tracking their evolution," *IEEE Comput.*, vol. 27, no. 7, pp. 20–27, Jul. 1994.

[3] L. Wang and N. Peters, "The length-scale distribution function of the distance between extremal points in passive scalar turbulence," *Fluid Mech.*, vol. 554, pp. 457–475, 2006.

[4] L. Wang and N. Peters, "Length-scale distribution functions and conditional means for various fields in turbulence," *J. Fluid Mech.*, vol. 608, pp. 113–138, 2008.

[5] L. Schäfer, J. H. Göbbert, and W. Schröder, "Dissipation element analysis in experimental and numerical shear flow," *Eur. J. Mech. - B/Fluids*, vol. 38, pp. 85–92, 2013.

[6] D. Silver and X. Wang, "Tracking and visualizing turbulent 3D features," *IEEE Trans. Vis. Comput. Graph.*, vol. 3, no. 2, pp. 129–141, Apr. 1997.

[7] D. Silver and X. Wang, "Volume tracking," in *Proc. IEEE Vis.*, 1996, pp. 157–164.

[8] J. Jeong and F. Hussain, "On the identification of a vortex," *J. Fluid Mech.*, vol. 285, pp. 69–94, 1995.

[9] S. Stegmaier, U. Rist, and T. Ertl, "Opening the can of worms: An exploration tool for vortical flows," in *Proc. IEEE Vis.*, 2005, pp. 463–470.

[10] M. Jiang, R. Machiraju, and D. Thompson, "Detection and visualization of vortices," in *Visualization Handbook*, C. R. Johnson and C. D. Hansen, Eds. Amsterdam, The Netherlands: Elsevier, 2004, pp. 295–309.

[11] K.-L. Ma, J. V. Rosendale, and W. Vermeer, "3D shock wave visualization on unstructured grids," in *Proc. Symp. Vol. Vis.*, 1996, pp. 87–94.

[12] A. Gyulassy, P.-T. Bremer, B. Hamann, and V. Pascucci, "A practical approach to morse-smale complex computation: Scalability and generality," *IEEE Trans. Vis. Comput. Graphi.*, vol. 14, no. 6, pp. 1619–1626, Nov. 2008.

[13] J. L. Helman and L. Hesselink, "Visualizing vector field topology in fluid flows," *IEEE Comput. Graph. Appl.*, vol. 11, no. 3, pp. 36–46, May 1991.

[14] G. Scheuermann and X. Tricoche, "Topological methods for flow visualization," in *Visualization Handbook*, C. R. Johnson and C. D. Hansen, Eds. Amsterdam, The Netherlands: Elsevier, 2004, pp. 341–356.

[15] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci, "Morse-smale complexes for piecewise linear 3-manifolds," in *Proc. Annu. Symp. Comput. Geometry*, 2003, pp. 361–370.
[16] A. Gyulassy, P. Bremer, R. Grout, H. Kolla, J. Chen, and V. Pascucci, "Stability of dissipation elements: A case study in combustion," *Comput. Graph Forum*, vol. 33, no. 3, pp. 51–60, 2014.
[17] R. M. Karp, " Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, Eds., New York, NY, USA: Springer, 1972, pp. 85–103.
[18] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ, USA: Prentice Hall, Feb. 1982.
[19] D. Silver and X. Wang, "Tracking scalar features in unstructured data sets," in *Proc. IEEE Vis.*, 1998, pp. 79–86.
[20] J. Chen, D. Silver, and L. Jiang, "The feature tree: Visualizing feature tracking in distributed AMR datasets," in *Proc. IEEE Symp Parallel Large Data Vis. Graph.*, 2003, pp. 103–110.
[21] I. K. Sethi, N. V. Patel, and J. H. Yoo, "A general approach for token correspondence," *Pattern Recognit.*, vol. 27, no. 12, pp. 1775–1786, 1994.
[22] F. Reinders, F. H. Post, and H. J. Spoelder, "Visualization of time-dependent data with feature tracking and event detection," *Visual Comput.*, vol. 17, no. 1, pp. 55–71, 2001.
[23] C. Muelder and K.-L. Ma, "Interactive feature extraction and tracking by utilizing region coherency," in *Proc. IEEE Pacific Vis.*, 2009, pp. 17–24.
[24] J. Clyne and P. M. A. Norton, "Physically-based feature tracking for CFD data," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 6, pp. 1020–1033, Jun. 2013.
[25] F. Sauer, H. Yu, and K.-L. Ma, "Trajectory-based flow feature tracking in joint particle/volume datasets," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 12, pp. 2565–2574, Dec. 2014.
[26] H. Theisel and H.-P. Seidel, "Feature flow fields," in *Proc. Joint EG/IEEE TCVG Symp. Vis.*, 2003, pp. 141–148.
[27] T. Weinkauf, H. Theisel, A. V. Gelder, and A. Pang, "Stable feature flow fields," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 6, pp. 770–780, Jun. 2011.
[28] C. Garth, X. Tricoche, and G. Scheuermann, "Tracking of vector field singularities in unstructured 3D time-dependent datasets," in *Proc. IEEE Vis.*, 2004, pp. 329–336.
[29] S. Ozer, J. Wei, D. E. Silver, K.-L. Ma, and P. Martin, "Group dynamics in scientific visualization," in *Proc. IEEE Symp. Large Data Anal. Vis.*, 2012, pp. 97–104.
[30] S. Ozer, D. E. Silver, K. Bemis, and P. Martin, "Activity detection in scientific visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 3, pp. 377–390, Mar. 2014.
[31] E. J. Griffith, F. H. Post, M. Koutek, T. Heus, and H. J. J. Jonker, "Feature tracking in VR for cumulus cloud life-cycle studies," in *Proc. EG Workshop Virtual Environments*, 2005, pp. 121–128.
[32] H. Doraiswamy, V. Natarajan, and R. S. Nanjundiah, "An exploration framework to identify and track movement of cloud systems," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 12, pp. 2896–2905, Dec. 2013.
[33] D. Laney, P.-T. Bremer, A. Mascarenhas, P. Miller, and V. Pascucci, "Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, pp. 1053–1060, Sep. 2006.
[34] P.-T. Bremer, G. H. Weber, V. Pascucci, M. Day, and J. B. Bell, "Analyzing and tracking burning structures in lean premixed hydrogen flames," *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 2, pp. 248–260, Mar. 2010.
[35] H. Saikia and T. Weinkauf, "Global feature tracking and similarity estimation in time-dependent scalar fields," *Comput. Graph. Forum*, vol. 36, no. 3, pp. 1–11, 2017.
[36] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Philadelphia, PA, USA: Society for Industrial Mathematics, 2009.
[37] A. V. Goldberg and R. Kennedy, " An efficient cost scaling algorithm for the assignment problem," *Math. Program.*, vol. 71, no. 2, pp. 153–177, 1995.
[38] CBC: An lp-based branch-and-cut library. [Online]. Available: https://projects.coin-or.org/Cbc, Last Access: May 15, 2015.
[39] N. Berr, D. Schmidl, J. H. Göbbert, S. Lankes, D. an Mey, T. Bemmerl, and C. H. Bischof, "Trajectory-search on scaleMP's vSMP architecture," in *Proc. PARCO*, 2011, pp. 227–234.
[40] L. A. Zager and G. C. Verghese, "Graph similarity scoring and matching," *Appl. Math. Lett.*, vol. 21, no. 1, pp. 86–94, Jan. 2008.

Andrea Schnorr received the master's degree in technomathematics from the FH Aachen University of Applied Sciences. She is a research assistant with the Virtual Reality and Immersive Visualization Group, RWTH Aachen University, Germany. Her research interests include the development and evaluation of feature tracking algorithms with a focus on space-filling structures in turbulent flow data sets.

Dirk N. Helmrich received the BSc degree in scientific programming from the FH Aachen University of Applied Sciences. He is a programmer with the Virtual Reality and Immersive Visualization Group, RWTH Aachen University, Germany. His work focuses on the evaluation of feature tracking algorithms and implementation of techniques for result analysis. He implements techniques from related works for comparison.

Dominik Denker received the master's degree in mechanical engineering from the RWTH Aachen University, in 2014. Since then, he has been working as a research assistant with the Institute for Combustion Technology in Aachen. His research focuses on numerical investigation and modeling of turbulence—flame interaction as well as turbulence theory.

Torsten W. Kuhlen is a full professor and the head of the Virtual Reality and Immersive Visualization Group, RWTH Aachen University, Germany. His research interests include all areas of virtual reality with a focus on immersive data analysis in scientific and technical applications as well as the design and evaluation of 3D, multimodal human computer interfaces. He has co-authored about 250 research papers and served as program chair, program committee member, and general chair for various renowned conferences on Virtual Reality, Computer Graphics, and Visualization.

Bernd Hentschel received the Dipl.-Inform. (MSc) and Dr. rer. nat. (PhD) degrees in computer science, both from RWTH. He is a senior researcher with the Virtual Reality and Immersive Visualization Group, RWTH Aachen University, Germany. His research interests include the analysis of domain-specific features in large simulation data, parallel visualization algorithms, and immersive visualization. He closely collaborates with domain scientists in order help them solve complex visualization problems.