Feature Tracking Using Reeb Graphs

Gunther Weber[1], Peer-Timo Bremer[2], Marcus Day[1], John Bell[1], and Valerio Pascucci[3]

[1] Lawrence Berkeley National Laboratory, {GHWeber|MSDay|JBBell}@lbl.gov
[2] Lawrence Livermore National Laboratory, ptbremer@llnl.gov
[3] University of Utah, pascucci@sci.utah.edu

Abstract. Tracking features and exploring their temporal dynamics can aid scientists in identifying interesting time intervals in a simulation and serve as basis for performing quantitative analyses of temporal phenomena. In this paper, we develop a novel approach for tracking subsets of isosurfaces, such as burning regions in simulated flames, which are defined as areas of high fuel consumption on a temperature isosurface. Tracking such regions as they merge and split over time can provide important insights into the impact of turbulence on the combustion process. However, the convoluted nature of the temperature isosurface and its rapid movement make this analysis particularly challenging.
Our approach tracks burning regions by extracting a temperature isovolume from the four-dimensional space-time temperature field. It then obtains isosurfaces for the original simulation time steps and labels individual connected "burning" regions based on the local fuel consumption value. Based on this information, a boundary surface between burning and non-burning regions is constructed. The Reeb graph of this boundary surface is the tracking graph for burning regions.

Key words: Topological data analysis, Feature tracking, Combustion simulation, Reeb graph, Tracking graph, Tracking accuracy

1 Introduction

Understanding combustion processes is a fundamental problem impacting areas such as engine and stationary power plant design, both in terms of production efficiency and pollutant emission. Fuel-lean flame configurations are of particular interest since such flames generically produce far lower pollutants than comparable fuel-rich or stoichiometrically mixed flames. However, such flames are difficult to stabilize in the sort of quasi-steady robust configurations necessary for practical applications, particularly when using advanced fuel mixtures, such as hydrogen-air and hydrogen-seeded methane-air. These advanced fuel mixtures, selected to reduce the use of carbon-based fuels and subsequent emissions, often burn in cellular patterns of intense chemical reaction, separated by regions of local extinction. A broad range of classical flame propagation models used in analysis and engineering design of practical combustion systems are based on the notion that a flame is a thin continuous interface separating cold reactants from hot products. Such models are not suitable for modelling cellular flames. It therefore is of great practical interest to understand this mode of combustion, with the ultimate goal of incorporating the cellular burning behavior into revised engineering design models.

In the present study, detailed numerical simulation is used to evolve a turbulent reacting hydrogen-air mixture in an idealized configuration. Characteristics of the flow

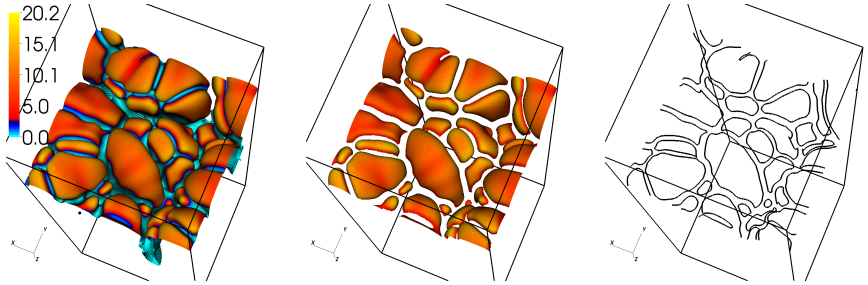

Fig. 1. Burning regions on an isotherm of a lean flame. (Left) Isotherm (T=1225K) extracted from a single time step and colored by fuel consumption rate. Blue signifies low fuel consumption, and orange signifies high fuel consumption. (Center) Burning regions are those portions of the isotherm where fuel consumption exceeds a given threshold (here: $2.6\frac{\mathrm{kg_{H_2}}}{\mathrm{m^3 s}}$ of H_2 consumption). (Right) Boundaries of the burning regions identified by contouring.

and turbulence used in the simulation represent conditions similar to those found in a practical turbulent combustor. The fields computed include the velocity and a set of scalar quantities representing the temperature and mass densities of a large number of molecular species. In the simulation these quantities are evolved in detail, naturally forming into locally disconnected (cellular) burning structures. It has long been known that the cellular hydrogen flame patterns correlate with a narrow band of intermediate temperature. Because of this correlation, we can build a simplified analysis of the detailed simulation data by looking at the variable rate of combustion along a representative isotherm (see Fig. 1, left illustration, for a representative example of such an analysis). Thresholding the rate of fuel consumption on the isotherm yields a geometric representation of the burning cells (Fig. 1, center illustration), and the time-evolution of these structures provides an important basic characterization of the flame.

Our goal is to compute a tracking graph whose nodes represent creation/destruction and split/merge events of burning cells and whose arcs represent evolution of cells over time. Tracking burning cells over time presents two main challenges. First, the aim is not to track surfaces per se, but features embedded in time-dependent surfaces. Second, to utilize the tracking graph fully, a correlation between a specific burning cell in one time step and a node/arc on the tracking graph must be maintained.

We derive this information by considering the boundaries of burning regions (Fig. 1, right illustration), obtained via a second contouring operation based on the fuel consumption rate on the triangle mesh comprising the isotherm, and determining when these boundaries split and merge over time. Our approach extracts the boundary of the space-time volume that each of these contours sweeps out over the course of the simulation and reduces the four-dimensional tracking problem to the computation of the *Reeb graph* of the resulting surface (using time as a Morse function), which encodes merge and split events. We also provide an in-depth analysis of tracking accuracy by comparing simulations performed under varying conditions and with different temporal and spatial resolutions. While the examples provided in this paper focus on the analysis of combustion simulations, the underlying framework for topology-based tracking obviously applies to a wide range of application areas.

2 Related Work

Time-dependent Isosurface Extraction. Given a manifold $\mathbb{M}$ and a scalar function $f : \mathbb{M} \to \mathbb{R}$, an *isosurface* or *level set* of f at isovalue s is defined as all points on $\mathbb{M}$ with function value s. If $\mathbb{M}$ is described by a three-dimensional rectilinear grid, isosurfaces can be efficiently constructed using the Marching Cubes algorithm [1, 2]. More recent work [3] has extended this approach using convex hull computations to define triangulations within a grid cell. This method eliminates the need for extensive case tables, ensures consistency, and easily generalizes to higher dimensions.

One can exploit this generality to visualize time-varying isosurfaces by treating time as an additional dimension creating the space-time manifold $\mathbb{M} \times \mathbb{R}$. Extracting a level set of $\mathbb{M} \times \mathbb{R}$ results in a three-dimensional hypersurface (embedded in four-dimensional space-time) comprised of tetrahedra. Intersecting this volumetric space-time mesh with planes of constant time values results in traditional two-dimensional isosurfaces at a given time. Therefore, the hypersurface is a comprehensive description of the temporal evolution of the corresponding two-dimensional isosurfaces.

Topological Analysis and Reeb Graphs. By construction, a level set of f provides only limited information on f as a whole, but very detailed information about a specific function value. To understand the global behavior of f, it is useful to analyze its *contours*, which are the connected components of all level sets of f. Contracting the contours of a *Morse function* f into points creates the *Reeb graph*, which encodes the topology of all level sets of f. *Nodes* of the Reeb graph are formed by the contours passing through critical points of f, i.e., points where contour topology changes. Its *arcs* are formed by the remaining contours, i.e., by the family of contours that do not change topology. For more details on Reeb graphs and algorithms to construct them efficiently, we refer the reader to the description by Pascucci et al. [4].

In our application, the boundaries of burning regions sweep out a two-dimensional space-time surface S (see Fig. 2). Using the time-coordinate of all vertices of S as the function f, the level set of f at value t describes the cell boundaries at time t. Thus, the Reeb graph of f encodes the temporal evolution of the boundaries and can be used as the tracking graph describing how cells merge and split over time.

Feature Tracking. Defining and tracking features has long been of interest to the visualization community. Most relevant to our work is feature tracking in the context of scalar field visualization. Here, one usually is interested in tracking features defined by thresholding or isosurface extraction [5].

Tracking algorithms can roughly be divided into two categories: tracking by geometry and tracking by topology. Methods in the former category use various forms of overlap and/or distance between geometric attributes, e.g., the center of gravity [6] or volume overlap [7, 8] for tracking. Laney et al. [9] use a similar approach to track bubble structures in turbulent mixing. Ji et al. [10, 11] track the evolution of isosurfaces in a time-dependent volume by extracting the 3D space-time isosurface in a 4D space.

Methods in the latter category [12, 13] compute tracking information topologically using, for example, Jacobi sets [14], which describe the paths all critical points take over time. Sohn and Bajaj [15] introduce a hybrid approach using volume matching similar to Silver and Wang [7, 8] instead of topological information [13, 14] to define correspondences between contour trees.

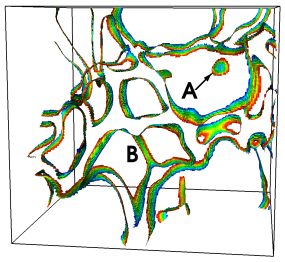

Fig. 2. Traced over time, the boundaries of the burning regions (Fig. 1, right illustration) sweep out surfaces. In this figure, the swept out surfaces are colored according to time (ranging from blue for early time steps to red for later time steps), which we use as the Morse function in a Reeb graph construction step.

Geometric tracking, in general, is ill-suited for the flame surfaces of interest here. As illustrated by Fig. 1, flame surfaces may be convoluted and contain many densely packed burning cells. As a result, geometric distance is not a good predictor for tracking burning cells, as flame sheets may fold on themselves, creating cells in close proximity yet far apart relative to the flame surface. Current algorithms for topological tracking using Jacobi sets are not capable of dealing with large embedded surfaces.

3 Feature Tracking Algorithm

An important aspect of the combustion process is how burning cells evolve over time. In particular, scientists are interested in determining how and when cells are created/destroyed and merge/split. To obtain this information, we track the boundaries of burning cells over time. For each time step, these boundaries are a set of curves, as shown in Fig. 1, right illustration.

If we follow these boundary lines over time, they sweep out surfaces; see Fig. 2. Using a 3D time surface in four dimensions, it is possible to extract this swept surface directly. In a first step, one views the data set as a 4D data set, with time as the fourth dimension, by concatenating all available time steps. The resulting data set serves as input for Marching Cubes to extract a 3D time surface comprised of tetrahedra. Furthermore, one also calculates the fuel consumption rate at the vertex positions and associates the rates with the individual vertices. A second step computes an isosurface of fuel consumption rate on this tetrahedral mesh, resulting in the swept boundary surface as shown in Fig. 2.

Based on this swept surface, it then is possible to track how burning regions change. If we take this swept boundary surface as a manifold, then the elapsed simulation time is a Morse function on it, and level sets correspond to boundaries at a given time. Critical points, where the number of contours of the level set changes, correspond to the changes of the number of burning cells. For example, in Fig. 2, in the area marked with the letter "A" a new burning region is created, and in the area around the letter "B," a burning region splits into two separate burning cells. Consequently, the Reeb graph of the swept boundary surface with time as a Morse function also is a tracking graph for individual burning cells (after simplification).

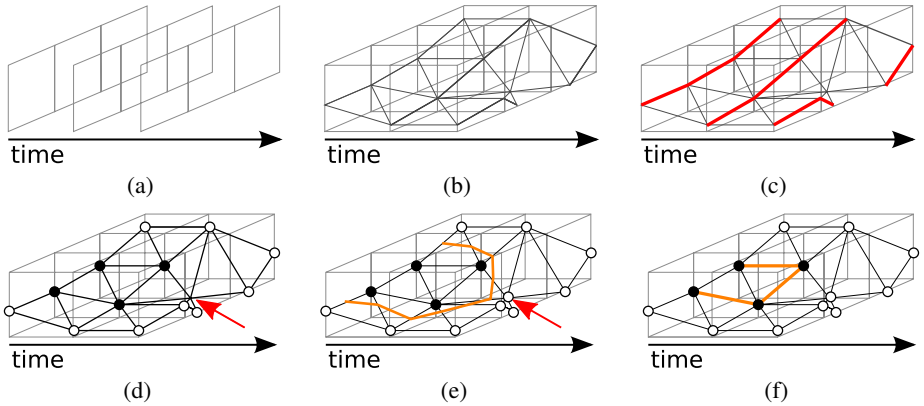

Fig. 3. Our method traces the evolution of burning regions and extinction pockets by tracking their boundaries. This figure illustrates the underlying concepts for the 2D case; the 3D case is analogous. (a) Input data comes as a set of discrete time slices. (b) We treat time as an additional, third dimension and extract an isosurface. The resulting time surface makes it possible to correlate isotherms from different time steps with each other. (c) We then extract isotherms (contour lines in 2D) for all original time steps by filtering all lines that have only vertices in the time step of interest (bold red lines in the figure). (d) We classify isotherm vertices at original time steps as either burning (solid black discs in the figure) or non-burning (empty circles in the figure) based on the local fuel consumption rates and simplify the segmentation using a Morse complex-based method. We note that vertices between time steps (arrowed vertex in figure) are not classified, yet. (e) We classify vertices between time steps (arrowed vertex in figure) by thresholding, and extract boundaries separating burning regions and extinction pockets (bold orange lines). (f) To simplify this step, we snap intersection points on edges that connect burning and non-burning vertices to the burning vertex (bold orange lines). We do so because it simplifies data processing and does not change the topology of the boundary surface. The Reeb graph (not shown) of the resulting surface is the desired tracking graph.

Our actual implementation is based on this fundamental concept with some additional refinements that we describe in the following. Due to data set size, we pipeline individual processing steps and stream data sets through this pipeline. While our pipeline performs all these steps on 3D data sets, we use the 2D case shown in Fig. 3 to illustrate the underlying concepts.

Extracting the Time Surface and Isosurfaces at the Original Time Steps. To extract the boundary of burning cells over time, we need a means to correlate isosurfaces in different time steps to each other. For this purpose, we add time as an additional dimension to the original 3D grid, resulting in a (virtual) 4D hyper-grid containing all time steps of the simulation. We then extract a 3D isovolume that encodes the time evolution of the flame surface, see Fig. 3(b). Intersecting this 3D (tetrahedral) isovolume with a plane of constant time produces the flame surface for that particular time step. We use the algorithm of Bhaniramka et al. to compute the isovolume and refer the reader to [3] for a more detailed description.

Here we are interested only in isosurfaces at times that correspond to original time steps in the simulation. In this special case, intersecting with a plane of constant time

can be reduced to a filter operation. Starting from the space-time tetrahedra, we obtain the isosurface as the set of all tetrahedra-faces (triangles) whose vertices all lie in the time step of interest. We perform this filter operation for all time steps of the original data set and save the corresponding flame surfaces. In addition to vertex positions, we also interpolate the fuel consumption scalar field and associate these values with the appropriate vertices.

Classification of Burning and Non-Burning Regions. Once all isosurfaces are extracted, we segment the surface for each time step into burning and non-burning regions. For this purpose, we do not apply simple thresholding to identify burning cells. Instead, we calculate the Morse complex of the isotherm for each individual time step, with fuel consumption as the associated Morse function [16]. This approach supports labeling each region with a unique identifier and performing persistence-based [17] simplification on the number of burning regions, merging small burning regions with nearby larger burning regions. While the hierarchy produced by this approach allows free selection of the fuel consumption threshold and the simplification persistence, we compute the segmentation for tracking using a persistence of 0.1 and a fuel consumption threshold of 2.6. These values are based on parameter studies performed in [16]. The result is a segmentation of the flame surfaces into individual burning cells that assigns a unique (within the time step) identifier to each cell. We propagate this identifier to all interior vertices of each cell.

Remember that the vertices of the flame surfaces are vertices of the space-time isovolume as well. Therefore, once we have computed the individual segmentations, only vertices in the regular time steps of the isovolume are classified as burning/non-burning, and the burning ones have a additional identifier attached. (We note that even though Fig. 2, shows a projection of the swept surface in 3D space, the triangle mesh actually is embedded in four-dimensional space, where each vertex also has a time coordinate. It is possible that the time coordinate of a vertex lies between two time steps of the simulation.) The remaining non-classified vertices are those falling between time steps. We classify these as burning/non-burning based on their fuel consumption value, but assign no identifier. This value of fuel consumption must be interpolated from the data available at adjacent time steps.

Extracting the Swept Boundary Surface. Based on the classification results, we could use a standard marching tetrahedra algorithm to extract the boundary between burning cells and extinction regions, shown as the bold orange line in Fig. 3(e). Each vertex of this space-time surface lies on an edge of the isovolume that connects a burning to a non-burning vertex (of the isovolume). As will be discussed later, we only are interested in the connectivity of the space-time surface and not its geometry. Therefore, we can simplify the extraction by snapping the intersection points on the tetrahedra edges to the burning vertices of the isovolume as shown in Fig. 3(f). This snapping reduces the extraction of the space-time surface to another filter operation. Starting from the isovolume, we discard all tetrahedra whose vertices are either all burning or all non-burning. We also discard tetrahedra with only a single burning vertex since the snapping operation will reduce the iso-triangle to a single point that does not contribute to the surface connectivity. Tetrahedra that have two burning and two non-burning vertices require

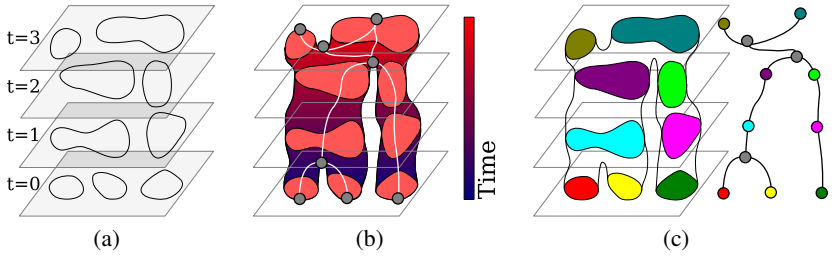

Fig. 4. Using the Reeb graph to compute a tracking graph. (a) Several time steps of a 2D data set with the boundaries of several cells indicated. (b) The space-time boundary created by the boundaries of the cells in (a) as they are interpolated over time. The surface is color coded using time as the Morse function, and the Reeb graph for the surface is shown in white and grey. (c) Each cell within a time step is assigned a unique identifier, which we use to augment the Reeb graph. Note that the final tracking graph still contains nodes between time steps without any identifier (shown in grey).

special attention. Here, snapping intersection points to the burning vertices results in two degenerate triangles, coinciding with the edge connecting the burning vertices. To maintain proper connectivity, we must add at least one of these triangles to the surface. Finally, tetrahedra with three burning vertices produce exactly one triangle identical to one of its faces that we add to the surface.

Computing the Tracking Graph. Ignoring the snapping operation, the space-time surface is the surface defined by the boundaries of the burning cells as they evolve over time, see Fig. 4(b). Computing the Reeb graph of the time function on this surfaces yields the tracking graph of the cells, as shown in Fig. 4 (c). Since the Reeb graph ignores the embedding of the underlying manifold (the space-time surface) snapping the vertices to those of the isovolume leaves the Reeb graph unchanged. As discussed in Sect. 2, each arc of the Reeb graph corresponds to contours that do not change topology as the function value changes. The Morse function in this case is time and therefore, arcs describe boundaries of burning cells that, over time, neither change genus, nor interact with other cells. Furthermore, at each time value for which a flame surface has been extracted, all vertices forming a contour have the same identifier by construction. We augment the Reeb graph with this information if necessary by adding nodes of valence two. This approach allows us to correlate specific cells of the flame surface with points in the tracking graph. This correlation is crucial when analyzing the graph. Finally, the identifiers also enable us to compute the genus of the burning cells. Each hole in the interior of a cell creates its own boundary component and thus appears in the Reeb graph as a separate component. However, using the cell identifiers, we can collect easily all components belonging to a particular cell and thus compute its genus.

Once we have computed the Reeb graph, we simplify [18] all loops that span less than a full time step since they must represent artifacts of the construction (using linear interpolation between time steps no true feature should exist between time steps). For display purposes we also merge all nodes representing the same cell while keeping track of its genus. Finally, we simplify all loops spanning exactly one time step

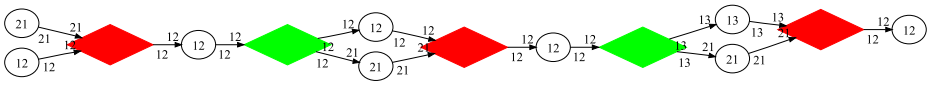

Fig. 5. An extended merge event: Two cells (initially labeled 12 and 21) merge and split twice before ultimately merging. These are inherent instabilities in segmenting by thresholding caused by saddles very close to the cut-off.

to streamline the graph. This simplification helps to resolve segmentation instabilities caused by saddles close to the threshold value. Using this information, we then construct a simplified graph representing the life of each component. Nodes of the graph indicate significant events: birth, death, splitting and merging. Diamond-shape nodes indicate events that occur between time steps. As shown in Fig. 5, one sometimes finds "extended" split/merge events in which two regions merge and split several time before finally merging/splitting. We also remove components of the graph that have an overall life-span of less than two time steps. We use "dot" [19] to layout the resulting graph.

4 Results

We have used our method to analyze simulations of a lean hydrogen flame burning under varying turbulence conditions labeled as "none," "weak" and "strong." With increasing levels of turbulence, we observe a qualitative change in the formation and propagation of burning cells. Our new tool can be used to quantify these changes. Fig. 6 shows a portion of the resulting tracking graph for the "none" turbulence case (the other two would produce a qualitatively similar result). The resulting graphs are very large, and evaluating differences between the different turbulence cases is the subject of ongoing research. However, the example clearly shows that the graph represents the split and merge events for the burning regions.

In the remainder of this section, we focus on evaluating tracking accuracy. Due to the size of the combustion simulations we are considering, we usually do not have every time step for the entire simulation available for analysis. Instead, the simulation code commonly saves only every fifth time step, and it is possible that features move several cells between subsequent saved time steps. Our tracking accuracy study was motivated by the fact that on first examination of the tracking graphs produced by our method, we noticed artifacts that indicated an occasional loss of some of the tracked components.

Fig. 7 shows an example of an artifact that arises due to limited temporal resolution. The first graph in Fig. 7, labeled "coarse," shows the tracking results obtained by using every fifth time step. Instead of showing a single death event for a burning region, the graph shows the death of a burning region and the simultaneous birth of another region that dies in the subsequent time step. The diamond shape of the death and birth events indicates that both occur between two real time steps.

We first assumed that such event sequences occur when our method failed to track a fast-moving component. In an attempt to remedy such errors, we inserted an interpolated time step between every two original simulation time steps. Adding interpolated time steps improved tracking performance in some instances, but not in the example shown here. In the second graph in Fig. 7, labeled "interpolated," the corresponding component dies off completely in the interpolated time step (452.5) and is "reborn"

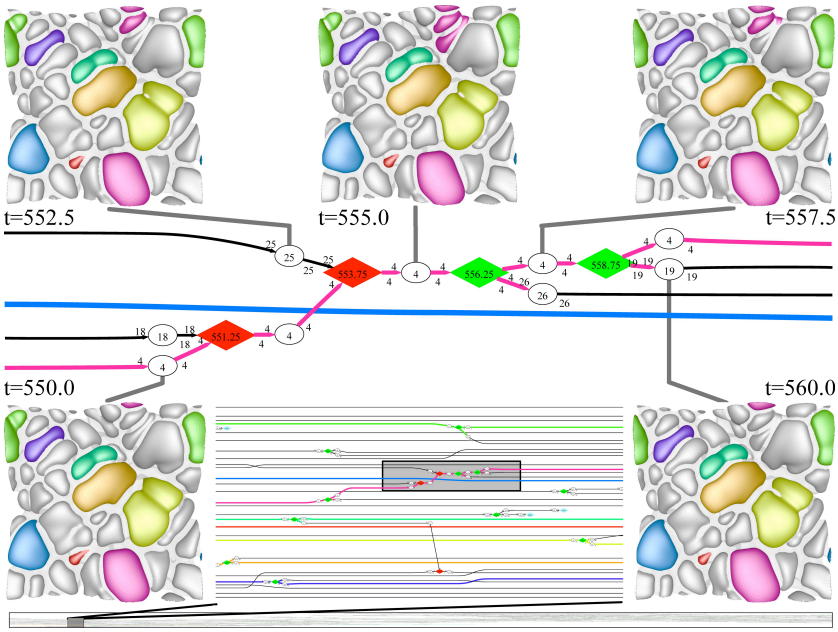

Fig. 6. Subsection of the tracking graph for the "no turbulence" case compressed to only show two merge and two split events. Arc color corresponds to region color in corresponding segmentations. Round nodes correspond to cells explicitly segmented by the Morse complex, diamonds to topological events between time steps. Red signifies a merge, green a split, and turquoise a birth/death event. The figure shows three zoom levels of the graph: graph for the entire simulation (bottom), a portion corresponding to several subsequent time steps (middle) and a zoom into the two merge and split events annotated with colored segmentations of the isotherm (top).

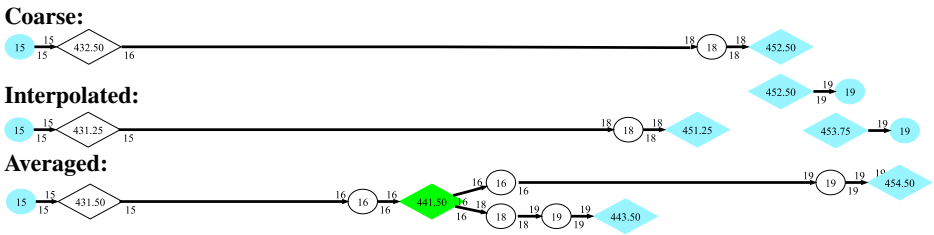

Fig. 7. "Lost" tracking of a burning region.

immediately before the next real simulation time step (455) and then immediately dies again. Closer examination of this "false death event" reveals that the burning region is moving several cells in a single time step so that there is no overlap of the cell in two subsequent time steps. Thus, whenever interpolated values are used, either by our classification between time steps in the coarse case, or when classifying the in-between time step in the interpolated case, interpolated values fall below the burning threshold and the region is classified as "extinguished."

For a small subrange of time (time step 431 to 511), we also had access to all time steps of a higher-resolution version of the simulation. We downsampled this higher-

resolution data by averaging in space to the same resolution as the original coarse case. With every time step available we used this version as a "gold standard," with the caveat that there may be differences due to performing the simulation at a higher resolution. The bottom graph in Fig. 7 shows the result of using this finer data, labeled "averaged". With all time steps available, the death event is captured properly. The graph also differs for earlier time steps in that an additional burning region splits off and dies. However, the split (time step 441) and subsequent death (time step 444) events occur entirely between time steps available in the coarse case; this event simply is missed by tracking with data at every fifth time step.

We compared tracking results for the three cases: (i) coarse data set with every fifth time step, (ii) coarse data set with interpolated time steps, and (iii) averaged time steps of the finer resolution simulation. During this period approximately 29 burning regions existed in the domain (at the beginning and the end of the time period there were 29 burning regions; in between, the number of burning regions varied). The tracking graphs for 16 of these 29 regions differed between the various analysis approaches.

Approximately 10% of the cases investigated showed inconsistencies between the coarse and fine analyses that could not be attributed directly to the interpolation errors discussed above. In these cases, the source of the discrepancies was traced to the data itself: the coarse and fine values of the threshold quantity did not provide a consistent segmentation, even though the actual solution used for the study was sufficiently grid-independent. The explanation for the discrepancy is related to the numerical integration procedures used to generate the threshold criteria itself. For our study, the combustion solutions were generated using a locally adaptive solution technique that has the property that proportionately small time steps are used near the highly reactive flame surface, which requires the smallest grid spacing in the domain (other regions in the solution were computed with cells that were 4-8 times larger). The thresholding quantity used for the study is derived from the state of the evolved system: the effective rate of fuel mass destruction over the local time step interval. However, this information is written to disk as snapshots at the larger time intervals related to the coarsest grid cells, so the finer diagnostic is undersampled by a factor of 4-8. Over this larger interval, flame features can translate several fine cells and generate substantial sampling errors. Unfortunately, solutions to this problem in our analyses would require a modification to the on-the-fly diagnostics routines in the flow solver, and therefore, is not feasible for the present study. As an aside, we can reasonably expect that thresholds based on evolved quantities, rather than derived ones, would be substantially more robust to this mode of failure.

Three of the 16 tracking differences were due to these differences in the data, with the difference between coarse and fine simulation being mainly the exact time when a merge took place. We verified that these paths and the tracking in the coarse case are correct for the supplied data. Consequently, tracking in the coarse case was correct in 16 cases. Furthermore, the discrepancy in two additional paths is due to two burning regions briefly merging between time steps of the coarse simulation, as shown in Fig. 8. Thus, these events do not appear in the coarse data, and we count only the loss of tracking for region 25 as error. The tracking differences for one region consists of two burning regions splitting off and dying between time steps available in the coarse

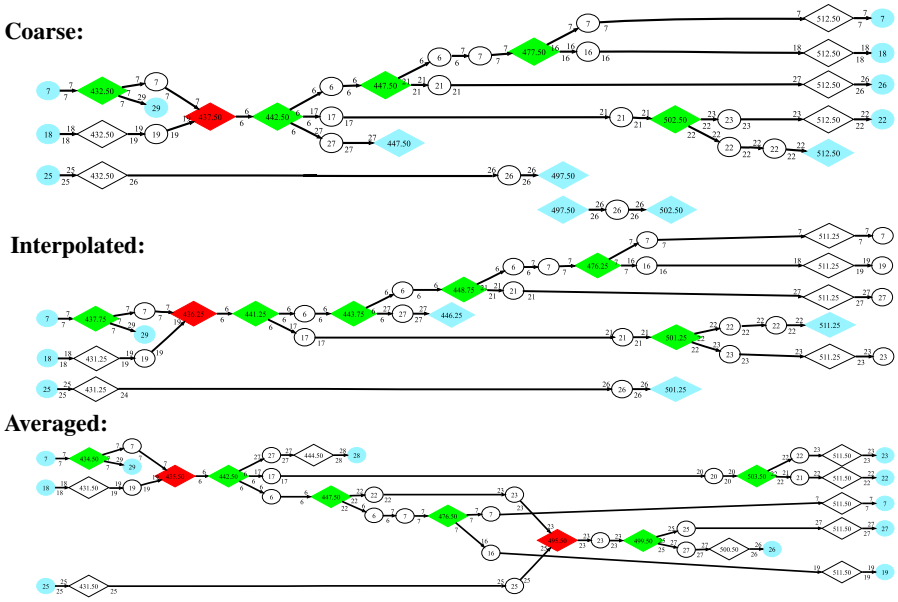

Fig. 8. A more complicated difference in tracking. In the averaged case the traces originating from regions 7 and 18 briefly merge with the trace of region 25. This merge is followed by an immediate split before the next available coarse time step, resulting in coarse and interpolated case missing it. The trace of region 25 is an example where interpolation eliminates a "false death event."

simulation. (Interestingly, the interpolated case detects one of these split/death events.) Discounting these differences, the coarse data allows correct tracking of 19 out of 29 regions. Most of the other differences involve erroneous merge events in the coarse case between available time steps, with the general evolution of burning regions being captured satisfactorily.

5 Conclusions and Future Work

We have presented a method to track burning regions in combustion simulations that uses very general, topology-based techniques and that can be adapted for other application areas. Our biggest problem is the lack of temporal resolutions since not all time steps are available for analysis. In some instances, creating interpolated time steps helps to improve tracking accuracy. However, in other cases interpolation aggravates problems due to fast moving burning regions. While improved interpolation techniques, e.g., Lagrangian-based techniques, that take additional velocity information available from the simulation into account, may improve our diagnostic, it is likely that we cannot avoid interpolation related problems completely. One possible solution is to integrate tracking and topological analysis into the simulation code, giving it access to all time steps. Going forward, we will pursue a tighter coupling between simulation and analysis. We further plan to perform tracking of burning regions in a full three-dimensional setting to avoid the issues associated with a sampling surface. For example, extracting an isotherm first adds an additional parameter (value of temperature); it is desirable to eliminate completely the influence of this parameter on analysis results.

6 Acknowledgments

This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract Nos. DE-AC02-05CH11231 (Lawrence Berkeley National Laboratory), DE-AC52-07NA27344 (Lawrence Livermore National Laboratory) and DE-FC02-06ER25781 (University of Utah) through the Scientific Discovery through Advanced Computing (SciDAC) program's Visualization and Analytics Center for Enabling Technologies (VACET) and the use of resources of the National Energy Research Scientific Computing Center (NERSC).

References

1. Lorensen, W., Cline, H.: Marching cubes: A high resolution 3d surface construction algorithm. SIGGRAPH Comp. Graph. **21**(4) (1987) 163–169
2. Nielson, G.: On marching cubes. IEEE Trans. Vis. Comp. Graph. **9**(3) (2003) 341–351
3. Bhaniramka, P., Wenger, R., Crawfis, R.: Isosurface construction in any dimension using convex hulls. IEEE Trans. Vis. Comp. Graph. **10**(2) (2004) 130–141
4. Pascucci, V., Scorzelli, G., Bremer, P.T., Mascarenhas, A.: Robust on-line computation of Reeb graphs: Simplicity and speed. ACM Trans. Graph. **26**(3) (2007) 58.1–58.9
5. Mascarenhas, A., Snoeyink, J. In: Isocontour based Visualization of Time-varying Scalar Fields. Springer Verlag (2009) 41–68 ISBN 978-3-540-25076-0.
6. Samtaney, R., Silver, D., Zabusky, N., Cao, J.: Visualizing features and tracking their evolution. IEEE Computer **27**(7) (1994) 20–27
7. Silver, D., Wang, X.: Tracking and visualizing turbulent 3d features. IEEE Trans. Vis. Comp. Graph. **3**(2) (1997) 129–141
8. Silver, D., Wang, X.: Tracking scalar features in unstructured datasets. In: Proc. IEEE Visualization '98. (1998) 79–86
9. Laney, D., Bremer, P.T., Mascarenhas, A., Miller, P., Pascucci, V.: Understanding the structure of the turbulent mixing layer in hydrodynamic instabilities. IEEE Trans. Vis. Comp. Graph. **12**(5) (2006) 1052–1060
10. Ji, G., Shen, H.W., Wegner, R.: Volume tracking using higher dimensional isocontouring. In: Proc. IEEE Visualization '03. (2003) 209–216
11. Ji, G., Shen, H.W.: Efficient isosurface tracking using precomputed correspondence table. In: Proc. IEEE/Eurographics Symposium Visualization '04. (2004) 283–292
12. Edelsbrunner, H., Harer, J., Mascarenhas, A., Pascuccii, V., Snoeyink, J.: Time-varying Reeb graphs for continuous space-time data. Comp. Geom. **41**(3) (2008) 149–166
13. Szymczak, A.: Subdomain-aware contour trees and contour tree evolution in time-dependent scalar fields. In: Proc. Shape Modeling International '05. (2005) 136–144
14. Edelsbrunner, H., Harer, J.: Jacobi sets of multiple Morse functions. In: Found. of Comput. Math., Minneapolis 2002. Cambridge Univ. Press, England (2002) 37–57
15. Sohn, B.S., Bajaj, C.: Time-varying contour topology. IEEE Trans. Vis. Comp. Graph. **12**(1) (2006) 14–25
16. Bremer, P.T., Weber, G., Pascucci, V., Day, M., Bell, J.: Analyzing and tracking burning structures in lean premixed hydrogen flames. IEEE Trans. Vis. Comp. Graph. **16**(2) (2010)
17. Edelsbrunner, H., Letscher, D., Zomorodian., A.: Topological persistence and simplification. Disc. Comput. Geom. **28**(4) (2002) 511–533
18. Agarwal, P., Edelsbrunner, H., Harer, J., Wang, Y.: Extreme elevation on a 2-manifold. Disc. Comput. Geom. **36**(4) (2006) 553–572
19. Koutsofios, E., North, S.: Drawing graphs with dot. Technical Report 910904-59113-08TM, AT&T Bell Laboratories, Murray Hill, NJ (1991)