Topological Cacti: Visualizing Contour-based Statistics

Gunther H. Weber, Peer-Timo Bremer and Valerio Pascucci

Abstract Contours, the connected components of level sets, play an important role in understanding the global structure of a scalar field. In particular their nesting behavior and topology—often represented in form of a contour tree—have been used extensively for visualization and analysis. However, traditional contour trees only encode structural properties like number of contours or the nesting of contours, but little quantitative information such as volume or other statistics. Here we use the segmentation implied by a contour tree to compute a large number of per-contour (interval) based statistics of both the function defining the contour tree as well as other co-located functions. We introduce a new visual metaphor for contour trees, called topological cacti, that extends the traditional toporrery display of a contour tree to display additional quantitative information as width of the cactus trunk and length of its spikes. We apply the new technique to scalar fields of varying dimension and different measures to demonstrate the effectiveness of the approach.

1 Introduction

Isosurfaces [10, 13, 14] are an extremely versatile and ubiquitous component of data analysis since distinct isosurfaces often have a useful physical interpretation directly related to the application domain. In premixed combustion simulations, for

Gunther H. Weber
Computational Research Division, Lawrence Berkeley National Laboratory, One Cyclotron Road, Berkeley, CA 94720 e-mail: GHWeber@lbl.gov

Peer-Timo Bremer
Center of Applied Scientific Computing, Lawrence Livermore National Laboratory, Box 808, L-560, Livermore, CA 94551 e-mail: bremer5@llnl.gov

Valerio Pascucci
Scientific Computing and Imaging Institute, University of Utah, 72 South Central Campus Drive, Salt Lake City, UT 84112 e-mail: pascucci@sci.utah.edu

example, an isotherm (isosurfaces of temperature) of the appropriate temperature is often associated with the location of the flame; in molecular simulations, isosurfaces of a certain charge density indicate molecular boundaries; and in fluid simulations, isosurfaces can be used to identify an interface between mixing fluids. Furthermore, isosurfaces provide a geometric representation of scientific data that supports the definition of further derived quantities (such as surface area) that are useful for an in-depth analysis.

Due to their importance, isosurfaces and in particular their topology have been the focus of an extensive body of research. In particular, the dependence of isosurface properties as a function of the isovalue has been examined in detail. Traditionally, there are two complementary types of information derived to aid in identifying interesting and relevant isosurfaces. Structural approaches, often based on the contour tree, provide information about individual contours, the connected components of isosurfaces, and how their number changes as the isovalue varies. The other class of approaches, including the contour spectrum, derive quantitative measurements, such as surface area from the isosurfaces, and plot these quantities as a function of isovalue. While providing quantitative information about the isosurface, information about individual connected components is lost, and these approaches usually result in a single one-dimensional function plot.

It is possible to combine these two complementary techniques by considering the segmentation implied by the contour tree. Each arc in the contour tree corresponds to a region swept by the contours represented by the arc. Using this segmentation, it is possible to calculate measurements on a per-contour basis. In this paper, we introduce *topological cacti*, a new visualization metaphor for the contour tree that incorporates these segmentation-based quantitative measurements into the graph display. Like the toporrery [15], topological cacti utilize a radial graph layout in three dimensions to avoid self-intersections. However, topological cacti also support the display of quantitative measures along with the graph. We draw graph edges of the toporerry as cylindrical extrusions and map a selected quantity to graph edge radius. We augment this visualization by adding spikes to display a second quantity, thus completing the cactus metaphor.

We will now describe how to display topological cacti. As a first step, we calculate measurements based on the segmentation implied by the contour tree (Sect. 3). In particular, we compute the contour spectrum for individual contours to provide per-contour area and volume measures (Sect. 3.1). Subsequently, we describe how to use this information in layout computation of topological cacti (Sect. 4). We illustrate the concept of topological cacti on example data sets using the per-contour representation of the contour spectrum (Sect. 5). We also demonstrate the display of other derived quantities, such as those associated with a join/merge tree resulting from the analysis of combustion simulations, as well as merge trees derived from higher-dimensional data.

2 Background and Related Work

2.1 Contour Trees

The contour tree [2] is a special case of the more generally defined Reeb graph [16] and captures the topological evolution of an isosurface as the isovalue varies. Critical points, where the number of contours changes, appear as nodes in the contour tree. Nodes of degree one (leaves of the tree) correspond to extrema, where contours are created or destroyed. Interior nodes of degree three or higher correspond to "saddles," where two or more contours merge or split. Arcs of the contour tree represent contours between critical points, i.e., contours that do not change topology (with the exception of genus changes) as the isovalue varies between critical values.

Conceptually, one can compute the contour tree by considering an isosurface for a particular isovalue and collapsing each contour (connected component) into a single point. Performing this process for each isovalue yields the graph structure described in the previous paragraph. However, currently most algorithms for computing the contour tree are based on the work by Carr et al. [5] and take a different, computationally more efficient approach. Instead of computing the contour tree directly, this algorithm starts by computing two simpler graphs, the join (or merge) tree and the split tree. The join tree tracks how connected regions above a threshold join as the threshold is decreased, and it can be computed efficiently using a union-find approach. The split tree analogously traces the behavior of regions below a threshold as that threshold is increased. A second phase of the algorithm uses join and split tree to compute the contour tree using graph-based operations. While join trees and split trees have been introduced in the context of contour tree calculation, they are also useful as structures in their own right. For example, they represent all possible segmentations based on thresholding and enables extensive parameter studies based on those segmentations [3].

Contour trees of even moderately-sized data sets can be quite complex. This complexity can be due to either noise or the presence of features at various scales. To remove noise, or features at smaller scale, it is necessary to simplify the contour tree. Carr et al. [6] simplified the contour tree using two operations: an arc-pruning operation to remove an extremum by "chopping" off a branch of the contour tree, and a vertex collapse to remove the resulting degree two node. Local geometric measures, like persistence (difference in function value) or volume, define an order in which arcs are pruned, i.e., what isosurface features are deemed less important and are removed first. Takahashi et al. [20, 19] use the related volume skeleton tree to simplify topology in a similar fashion. In subsequent work, they utilized the results, e.g., for the design of transfer functions that emphasize topology and take contour nesting properties into account [21].

Pascucci et al. [15] proposed the branch decomposition of the contour tree as an efficient data structure for encoding a multi-resolution representation of the contour tree. This representation decomposes the contour tree into a set of branches stored as a rooted tree. A root branch connects a minimum-maximum pair in the contour

tree. Other extrema are ordered by an importance metric (e.g., persistence), and child branches connect them to a saddle in an existing branch. In the resulting tree, each extremum appears as part of a branch, and the level in which a branch appear corresponds to its importance. The less important a branch is, the further down in the tree it appears. Using this data structure, it is possible to traverse the contour tree efficiently up to a desired level of detail (as specified by the importance metric).

2.2 Contour Tree Visualization

The contour tree is a graph and it is possible to display it using standard graph drawing algorithms like those provided by the GraphViz package [8]. Traditional layouts of the contour tree usually constrain the height of nodes to correspond to function value. This layout has the advantage that a horizontal line drawn at the height for a given isovalue intersects as many edges (or nodes) in the contour tree as there are contours for that value. This property significantly increases readability of the graph. More recently, Heine et al. [9] presented a fast and effective technique for contour tree layout in the plane.

One disadvantage of the height constraint is that it is not always possible to layout the contour tree without self-intersection. To avoid this problem, Pascucci et al. [15] proposed an alternative three-dimensional radial layout for the contour tree. This layout is motivated by using an orrery, i.e., a mechanical device that illustrates relative positions of planets and moons in the solar system, as a metaphor. Extrema take the place of planets and moons, and the toporrery shows the hierarchical relationship between the branches connecting them.

The toporrery preserves the contour tree as a graph, and its layout is related to traditional radial graphic layouts. As such, it does not provide information beyond the graph structure of the contour tree. An alternate approach uses structural information to illustrate the nesting behavior of contours [18]. Mizuta et al. [12, 11] computed contour trees on digital image data and visualized the resulting trees using the contour nest. The contour nest focuses on the nesting properties of isosurfaces and represents the contour tree as a set of nested rectangles. In this visualization, rectangle size corresponds to feature size (area in 2D and volume in 3D). However, the contour nest displays only a global value for each branch and does not show how measures vary as the isovalue of a contour changes.

Topological landscapes [22] utilize another metaphor to visualize a contour tree. Observing that the contour tree was originally introduced in the context of two-dimensional terrains and that a terrain can have the topology implied by a contour tree, this metaphor constructs a terrain that has the same topology as the contour tree. In this way, the terrain serves as a proxy representation for a higher-dimensional (three dimensions or more) data set. To improve the expressiveness of the terrain metaphor, it is possible to re-parametrize the terrain in such a way that the area of features corresponds to a specified metric (e.g., volume associated with a branch of

the branch decomposition). Like the contour nest, the construction scheme is limited to visualizing on the branch level, i.e., variations within a branch are ignored.

2.3 Contour Spectrum

While the contour tree provides structural information about an isosurface and the behavior of individual components, it does not provide quantitative information. The contour spectrum [1] on the other hand provides global measures for an isosurface as a function of isovalue. Observing that the surface area of an isosurface in a tetrahedral mesh can be expressed as a B-Spline, the authors develop a framework that encodes total surface area, exterior and interior volume, as well as gradient magnitude.

Fundamentally, the authors use the connection between isosurfaces on a tetrahedral mesh and simplex splines. Based on this relationship, they derive a B-Spline representation of isosurface area as a function of isovalue within an individual tetrahedron of the data set. Combining the B-Spline functions of all individual tetrahedra of a given mesh results in a B-Spline representation of the entire data set. The authors subsequently derive a second B-Spline describing the volume enclosed in the isosurface by integrating the area B-Spline.

3 Computing Per-Component Measures

To display per-component information about an isosurface, we need to compute it and associate it with the contour tree. For the purpose of computing these measures, we extend the contour spectrum to provide per-component measures (Sect. 3.1) or calculate measures based on the segmentation implied by the contour tree (Sect. 3.3). While prior work uses, e.g., interval volumes [7], to derive measures such as area and volume for contours (i.e., connected isosurface components), our approach provides these measures as a function of isovalue that is associated with contour tree edges. This function representation makes it possible to compute an individual, contour-specific area or volume for any isovalue along an edge, as opposed to approximating a single volume for an entire edge like prior work on topology simplification [20, 21].

In this context, it is beneficial to use the branch decomposition of the contour tree. (i) Branches in this decomposition may combine several arcs of the contour tree and therefore the segmentation consists of fewer regions. Nevertheless, each contour maps uniquely to a unique branch and a value along the branch. (ii) The hierarchical nature of the branch decomposition simplifies finding paths in the contour tree. (iii) The topological cactus layout is based on the toporrery layout, which is inherently based on the branch decomposition.

3.1 Computing the Contour Spectrum

We compute per-component measures of area and volume using the contour spectrum. Computing the area is relatively straightforward, as there are no ambiguities. Associating volume with the contour tree is more difficult, since for nested contours the definition of enclosed volume is ambiguous. For both measurements, we associate either a B-Spline or a piecewise polynomial function with each branch of the branch decomposition.

3.2 Computing per-contour Area

Calculating the contour spectrum globally relies on the fact, that within a tetrahedral cell surface area can be expressed as a function of isovalue as a B-Spline. In the original contour spectrum algorithm, contributions of all tetrahedra are accumulated into a global B-Spline. To compute area on a per-component basis, we associate a B-Spline (or alternatively a piecewise polynomial function) with each branch of the branch decomposition, and accumulate contributions per-component.

For accumulating per-component area, we observe that linear tetrahedra do not contain any critical points in their interior. In particular, they do not contain any saddles. Consequently, the "contour tree" for the individual tetrahedra is a straight line, ranging from its minimum vertex to its maximum vertex. Thus, in the contour tree a tetrahedron corresponds to a single path between two vertices (not necessarily nodes), along which the function value uniquely identifies the contour containing this tetrahedron. As a result, we can consider each tetrahedron and accumulate its contributions to the branch decomposition.

We perform this accumulation as follows. First, we compute contour tree and branch decomposition. We also compute the segmentation of the data set implied by the contour tree. This segmentation is stored as an array with an entry for each sample of the data set that contains a reference to the single branch within the decomposition that corresponds to the sample.

Subsequently, we accumulate the contributions of all tetrahedra to the contour spectrum. For each tetrahedron, we first determine its minimum and its maximum vertex. These vertices define the path in the contour tree along which contributions have to be accumulated, as shown in Fig. 1. We use the segmentation information to identify the branches corresponding to the minimum and maximum value.

We then find the path in the branch decomposition that connects the minimum and the maximum. Since the branch decomposition is a rooted tree (of branches), we simply find the common ancestor branch of the minimum and maximum vertex to construct their connecting path, see Fig. 1. In the figure, the root branch is the common branch, the path from the maximum to the root traverses two branches and the path from the minimum traverses one branch. Therefore, two lists of branches descending from the minimum and the maximum, as well as the common branch

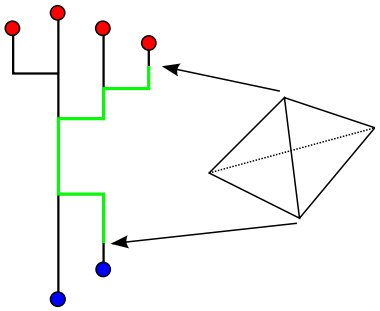

Fig. 1 Computing the contour spectrum utilizing the branch decomposition. A linear tetrahedron corresponds to a path in the branch decomposition. After identifying this path connecting the tetrahedron's maximum and minimum vertex, we add appropriate contributions to the contour spectrum associated with each branch.

connecting the last branches in these lists, completely define the path in the branch decomposition.

Subsequently, we split the B-Spline or piecewise polynomial function into segments corresponding to the branches along the path. For the first branch from the maximum, this segment corresponds to the range from the maximum tetrahedron value to its saddle. For all subsequent branches along the path, the segment corresponds to the range from the saddle value of the previous branch along the path to the saddle value of the current branch. Segments for the minimum path are defined analogously. Finally, the segment along the branch connecting the two paths corresponds to the range between the two saddle values.

3.2.1 Computing per-contour Volume

Computing per-contour volume is more difficult. If a contour contains another contour, it is ambiguous, whether the volume of the nested contour should be subtracted or not. To avoid ambiguities, we follow the convention that we calculate volume from the last saddle. Thus, the volume associated with a contour corresponds to the volume swept by that contour as long as it "maintained its identity," i.e. since its last "interaction" with another contour. Adding the contribution to the branch decomposition is similar to adding the contribution for the area. There are two key differences: Instead of stopping at the maximum value of the tetrahedron, the contribution on the maximum branch reaches until the next saddle value. This difference is due to the fact that once the maximum value of the tetrahedra is passed the volume function is a constant corresponding to the volume of the tetrahedron. Furthermore, following our convention, for each segment, we subtract the volume at the saddle from the entire function, such that for each segment only the volume swept since the saddle is considered.

Unlike the original contour spectrum approach, we also observe that obtaining the volume by integrating the area yields incorrect results. Expressing the volume as integral of area corresponds to a substitution of the integration variable, and we need to multiply the result by the derivative, which corresponds to multiplication with the inverse gradient magnitude. This is analogous to observations on computing other isosurface statistics [17].

3.3 Segmentation-based Measures

Visualizing volume and area already enhances contour tree display and provides valuable information about a function. However, in many application areas, it is desirable to display additional statistics, such as the mean value of a second variable on the isosurface. We compute these values at discrete locations of the contour tree or the join tree. To compute these values, we consider the segmentation implied by the contour tree [23] or the join tree [3] and accumulate these statistics for all samples corresponding to a branch or arc. This calculation would yield a single value per branch. To approximate function behavior along a branch or arc, we split it at regular intervals and insert degree two nodes into the join tree. When computing the branch decomposition, we associate these function values with the branch. This approach basically approximate calculating statistics with a segmentation defined in interval volumes [20].

4 Topological Cacti Layout

For generating topological cacti, we use an radial graph layout identical to that of the toporrery [15]. Extrema are ordered in concentric discs, where the disc is chosen based on the depth of an extremum in the branch decomposition. Like in the original toporrery, the radius ratio is constant in our layout, and each branch is assigned a radial wedge with an angle proportional to the number of child branches. To display metrics, we draw each branch as cylindrical extrusion and chose the radii according to a first selected measure (clamping however at a minimum radius to ensure that branches do not disappear). We draw connectors as sphere sweeps to ensure smooth connection of the cylindrical extrusion corresponding to parent and child branch. Before mapping a metric such as volume to radius, we support choosing a variety of transformation functions (square root, cube root, logarithm) to ensure that higher-dimensional properties are appropriately scaled to the one dimensional radius. This is analogous to a similar transformation before mapping volume to area in the topological landscapes [22].

For display of a second metric, we augment the cylindrical extrusions with spikes. We start by placing circles along regular intervals of the cylindrical extrusion and choosing angles in regular intervals along this circle. We then choose spike

location by jittering these regular samples, i.e., we add random offset angles and heights to the originally chosen spike locations. We then calculate the metric value for the function value corresponding to the (jittered) height. Spikes are drawn as cones with a fixed (user defined) base radius and a length depending on the metrics. Similar to the extrusion radius, we support user chosen mappings such as square root, cube root or logarithm.

5 Results

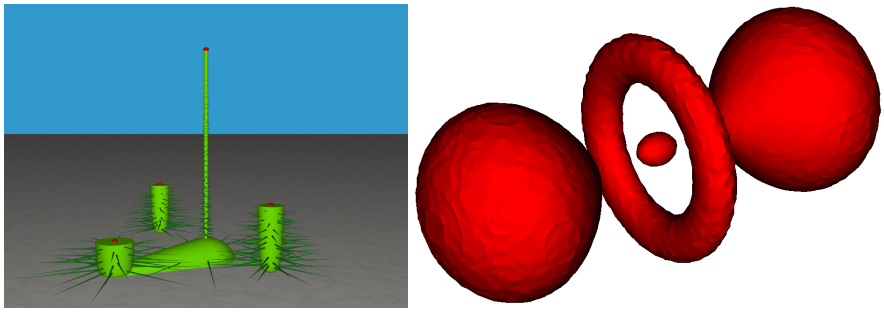

Fig. 2 Topological cactus for the hydrogen atom (left) and isosurface rendering of the corresponding data set (right). The root branch in the center corresponds to the contour in the center of the ring. The two symmetric branches correspond to the two lobes, and the lowest branch to the ring. The branch width corresponds to the cube root of enclosed volume, and spike length to the square root of surface area.

Figure 2 shows an example of a topological cactus for the hydrogen atom data set (simplified to a persistence of ten). Branch width corresponds to the cube root of volume and spike length to the square root surface area. The root branch corresponds to the evolution of the contour in the center. It is immediately obvious that despite its very high persistence (resulting in it being chosen as root branch), both its volume and surface area are very small. The two lobes, corresponding to the two symmetric branches are much larger in both volume and surface area. Finally, the ring, corresponding to the lowest branch, has the largest volume and area.

Figure 3 shows the topological cactus for the methane data set. The image on the right hand side shows corresponding isosurfaces. The red isosurface corresponds to a level close to the maxima and consists of twelve contours that are created around the twelve maxima visible on the cactus. As the cactus view shows, the three contours of each “arm” merge first, and subsequently these four “arms” merge into a single contour. At a lower isovalue, this contour splits into two separate contours. The cactus shows, how the volume of these components varies with the isovalue.

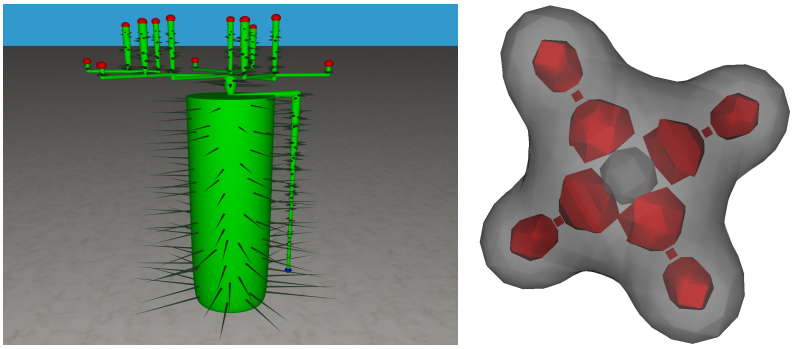

Fig. 3 Topological cactus for the methane molecule (left) and isosurface rendering of the corresponding data set (right). The right hand image shows isosurfaces for two isovalues. The red isosurface corresponds to isovalue close to the maximum value and shows contours created around the twelve maxima. The gray isosurface corresponds to a lower isovalue at a level where all twelve red contours have merged and then split into two separate components. In the cactus display, branch width corresponds to the cube root of volume and spike length to the square root of area.

For example, surface area and volume of the outer component remain relatively constant, while the inner component shrinks slowly and disappears.

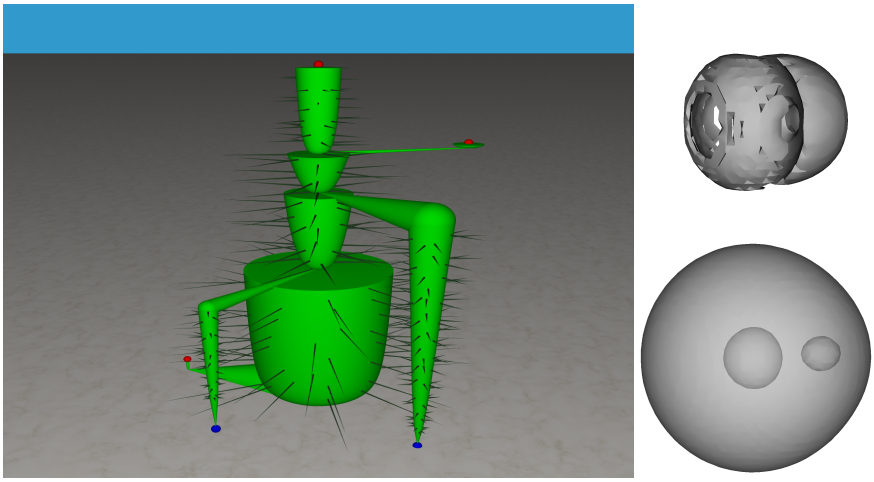

Fig. 4 Topological cactus for the nucleon (left) and isosurface rendering of the corresponding data set (right). Branch width corresponds to the cube root of volume, and spike length to the square root of surface area. The cactus only shows branches with a minimum persistence of three (data values are quantized to byte values).

Figure 4 shows the nucleon data set (simplified to a persistence of three). The right hand side shows isosurfaces at two levels (in separate images due to complex-

ity). The lower isosurface image shows contours just below the second lowest saddle resulting in three separate contour. We note that for any value, the contour branching off at the lowest saddle is too small to be visible in an rendered isosurface image. The cactus representation also clearly shows that volume and area for this contour are minute. The upper isosurface shows contours for an isovalue slightly above the higher saddle. The cactus shows clearly that even though one of the components has only small persistence, it has actual a volume comparable to the other component.

Figure 5 shows the topological cactus of the merge tree of fuel consumption rate in a combustion simulation. Branch width corresponds to the cube root of number of vertices, thus approximating volume. Spike length corresponds to the variance of temperature. Each branch corresponds to a burning region around a maximum of fuel consumption. The cactus shows that there are two types of regions: (i) Regions that are relatively wide close to the maximum and thus are comprised of a relatively large number of vertices and (ii) smaller regions with a thin region around the maximum. The latter occur at lower fuel-consumption levels. The spikes show that temperature variance is small for high fuel consumption, corresponding to intense burning and is (expectedly) larger for fuel consumption corresponding to non-burning regions.

Figure 6 shows the cactus of the merge tree of a 21 dimensional climate simulation ensemble and the more traditional graph layout of the branch decomposition of the merge tree. The data set consists of 1197 climate simulations performed with the Community Atmospheric Model (CAM) [4] each using a different combination of 21 input parameters. Such ensembles are used to study the sensitivity of the climate predictions to changes in input parameters. The output function shown in Fig. 6 is the average longwave energy output (FLUT) in the winter months (December, January, February). Unlike the traditional graph layout, the cactus shows that most of the "volume" of the parameter space is collected in several large but relatively low persistent branches of maxima. This fact is interesting as the observed energy output is expected to be closer to the global minimum. Thus, it appears that only a small portion of the input samples produces realistic outputs. Nevertheless, all persistences are fairy low indicating that the overall behavior is stable under perturbation of the input parameters.

6 Conclusions and Future Work

We have introduced topological cacti and demonstrated how they combine insights obtained from the contour tree with metrics derived on a per contour bases. In particular, where importance measured by persistence differs from other metrics such as area of volume, this combination can greatly enhance the usefulness of contour tree representations. Going forward, we plan to improve both contour spectrum and cactus representation. Our current approach of computing the contour spectrum in piecewise polynomial/B-Spline representation works well for quantized data sets. However, for real-valued data, each data value appears in the knot vector, making

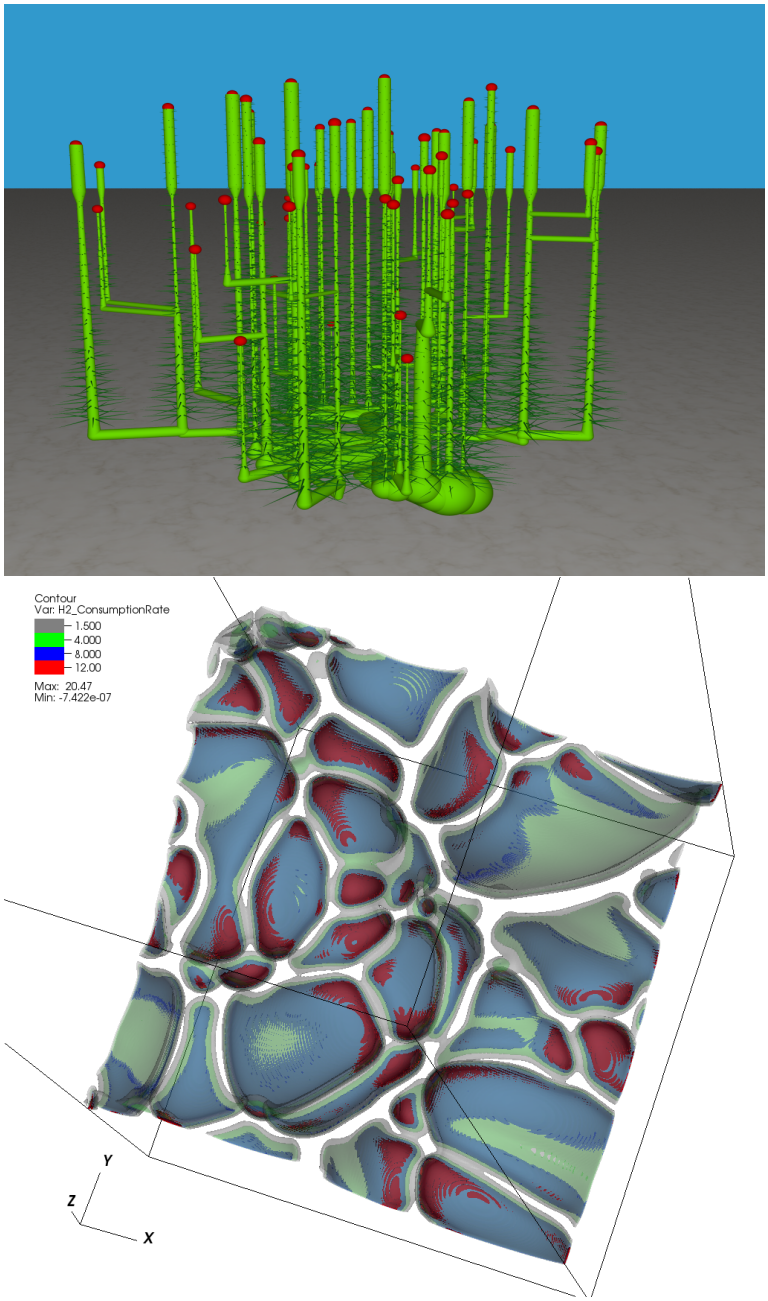

Fig. 5 Topological cactus of the merge tree of fuel consumption rate in a combustion simulation (top). The width corresponds to the cube root of number of vertices in a region. Spike length shows the variance of temperature. Isosurfaces for four different fuel consumption rates illustrating burning regions for these thresholds (bottom).

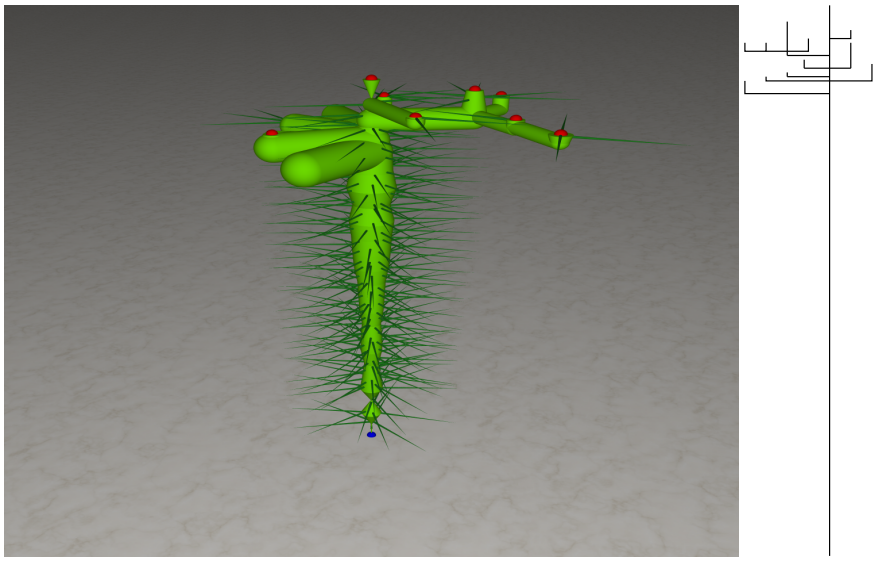

Fig. 6 Topological cactus of a climate simulation showing the merge tree of long wave energy output in the winter months (left) and graph layout of the branch decomposition (right). The width corresponds to cube root of vertex count. The spike length shows the mean value of FLUT.

this representation inefficient. To counteract this problem, we need to devise a simplification scheme that can be applied during the calculation and still support error bounds.

For the cactus representation, we plan to improve the way connectors are drawn. Drawing them as sphere sweeps sometimes hides details about the width of the parent branch. Furthermore, we plan to improve the placement of spikes to make their distribution more even, e.g., by increasing the number of spikes per angle for very wide branches (i.e., where the first metric is large).

Acknowledgements

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) and the use of resources of the National Energy Research Scientific Computing Center (NERSC).

References

1. Bajaj, C., Pascucci, V., Schikore, D.R.: The contour spectrum. In: Proc. IEEE Visualization, pp. 167–175 (1997)
2. Boyell, R.L., Ruston, H.: Hybrid techniques for real-time radar simulation. In: Proc. 1963 Fall Joint Computer Conference, pp. 445–458. IEEE (1963)
3. Bremer, P.T., Weber, G.H., Tierny, J., Pascucci, V., Day, M.S., Bell, J.B.: A topological framework for the interactive exploration of large scale turbulent combustion. In: Proc. 5th IEEE International Conference on e-Science, pp. 247–254. Oxford, UK (2009)
4. Community atmosphere model (CAM). See http://www.ccsm.ucar.edu/models/atm-cam/
5. Carr, H., Snoeyink, J., Axen, U.: Computing contour trees in all dimensions. Computational Geometry—Theory and Applications **24**(2), 75–94 (2003)
6. Carr, H., Snoeyink, J., van de Panne, M.: Simplifying flexible isosurfaces using local geometric measures. In: Proc. IEEE Visualization 2004, pp. 497–504 (2004)
7. Fujishiro, I., Maeda, Y., Sato, H., Takeshima, Y.: Volumetric data exploration using interval volume. IEEE Trans. Vis. Comp. Graph. **2**(2), 144–155 (1996)
8. Gansner, E.R., North, S.C.: An open graph visualization system and its applications to software engineering. Software—Practice and Experience **30**, 1203–1233 (1999)
9. Heine, C., Schneider, D., Carr, H., Scheuermann, G.: Drawing contour trees in the plane. IEEE Trans. on Vis. Comp. Graph. (2011). PrePrint doi: 10.1109/TVCG.2010.270. In press.
10. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3D surface construction algorithm. Computer Graphics (Proc. ACM SIGGRAPH 87) **21**(4), 163–169 (1987)
11. Mizuta, S., Ono, T., Matsuda, T.: Contour nest: A two-dimensional representation for three-dimensional isosurfaces. In: Proc. Volume Graphics, pp. 67–70 (2006)
12. Mizuta, S., Suwa, Y., Ono, T., Matsuda, T.: Description of the topological structure of digital images by contour tree and its application. Tech. rep., Institute of Electronics, Information and Communication Engineers (2004)
13. Montani, C., Scateni, R., Scopigno, R.: A modified look-up table for implicit disambiguation of marching cubes. The Visual Computer **10**(6), 353–355 (1994)
14. Nielson, G.M.: On marching cubes. IEEE Trans. Vis. Comp. Graph. **9**(3), 341–351 (2003)
15. Pascucci, V., Cole-McLaughlin, K., Scorzelli, G.: The Toporrery: Computation and Presentation of Multi-Resolution Topology, pp. 19–40. Springer-Verlag (2009)
16. Reeb, G.: Sur les points singuliers d'une forme de pfaff complètement intégrable ou d'une fonction numérique. Comptes Rendus de l'Acadèmie des Sciences de Paris **222**, 847–849 (1946)
17. Scheidegger, C.E., Schreiner, J.M., Duffy, B., Carr, H., Silva, C.T.: Revisiting histograms and isosurface statistics. IEEE Trans. Vis. Comp. Graph. **14**, 1659–1666 (2008)
18. Shinagawa, Y., Kunii, T.L., Kergosien, Y.L.: Surface coding based on morse theory. IEEE Computer Graphics & Applications **11**(5), 66–78 (1991)
19. Takahashi, S., Nielson, G.M., Takeshima, Y., Fujishiro, I.: Topological volume skeletonization using adaptive tetrahedralization. In: Proc. Geometric Modeling and Processing, pp. 227–236 (2004)
20. Takahashi, S., Takeshima, Y., Fujishiro, I.: Topological volume skeletonization and its application to transfer function design. Graphical Models **66**(1), 24–49 (2004)
21. Takahashi, S., Takeshima, Y., Fujishiro, I., Nielson, G.M.: Emphasizing isosurface embeddings in direct volume rendering. In: Scientific Visualization: The Visual Extraction of Knowledge from Data, pp. 185–206. Springer-Verlag (2005)
22. Weber, G.H., Bremer, P.T., Pascucci, V.: Topological landscapes: A terrain metaphor for scientific data. IEEE Trans. Vis. Comp. Graph. (Proc. Visualization 2007) **13**(6), 1416–1423 (2007)
23. Weber, G.H., Dillard, S.E., Carr, H., Pascucci, V., Hamann, B.: Topology-controlled volume rendering. IEEE Trans. Vis. Comp. Graph. **13**(2), 330–341 (2007)

Disclaimer