Topological Landscapes: A Terrain Metaphor for Scientific Data

Gunther H. Weber, *Member, IEEE Computer Society,* Peer-Timo Bremer, *Member, IEEE*
and Valerio Pascucci, *Member, IEEE*

Abstract—Scientific visualization and illustration tools are designed to help people understand the structure and complexity of scientific data with images that are as informative and intuitive as possible. In this context the use of metaphors plays an important role since they make complex information easily accessible by using commonly known concepts. In this paper we propose a new metaphor, called "Topological Landscapes," which facilitates understanding the topological structure of scalar functions. The basic idea is to construct a terrain with the same topology as a given dataset and to display the terrain as an easily understood representation of the actual input data. In this projection from an n-dimensional scalar function to a two-dimensional (2D) model we preserve function values of critical points, the persistence (function span) of topological features, and one possible additional metric property (in our examples volume). By displaying this topologically equivalent landscape together with the original data we harness the natural human proficiency in understanding terrain topography and make complex topological information easily accessible.

Index Terms—*Feature Detection* (primary keyword), User Interfaces, Visual Analytics, Contour Tree, Terrain, Topology, SOAR

1 Introduction

Scalar functions are an ubiquitous type of scientific data used to represent a wide variety of phenomena in application areas including chemistry, climate modeling, medicine, and physics. One of the fundamental tasks in scientific visualization is to provide users with a correct general understanding of data complexity and structure and to determine (i) if any major unexpected features may be present, and (ii) at which locations there may be a greater need for interactive exploration and detailed data analysis.

Classical approaches providing a quick overview of data include displaying multiple isosurfaces [1] or the use of direct volume rendering with varying transfer functions [2]. However, such techniques only show a small sub-set of the data in any one image and depend heavily on parameter selection. This requires users to combine information from a potentially large number of images. Furthermore, it is usually impractical to interactively explore the complete parameter space.

To complement direct visualizations, histograms of various type and dimension [3, 4, 5, 6, 7] have been proposed to reduce the time spent on trial and error data exploration. Such methods are most successful when used in conjunction with direct visualization. Direct visualization is very intuitive and familiar to users while histograms provide an overview of the data with respect to one or multiple parameters. However, the effectiveness of histograms is often limited as they only provide aggregate global information, are not localized spatially, and often machine learning techniques [8, 9] are necessary to utilize their full potential.

An alternate approach to understanding the structure of a function is to compute its topology, for example, by extracting its contour tree [10, 11, 12]. This type of analysis provides a complete study of the variation of a function and describes all its features and their location in space either in terms of critical points [12] or in terms of entire regions [13]. Moreover, one can compute additional properties, such as area/volume [14] or genus [15], that further quantify the shape of isosurfaces in a particular region. Knowledge of this complete and accurate information can enhance and guide standard visualization techniques [16, 14].

- *G.H. Weber is with the Computational Research Division, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA-94720. E-mail: ghweber@lbl.gov.*
- *P.-T. Bremer and V. Pascucci are with the Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Box 808, L-551, Livermore, CA 94551 E-mail: {ptbremer,pascucci}@llnl.gov.*

Manuscript received 31 March 2007; accepted 1 August 2007; posted online 27 October 2007.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

Topological information is very abstract and difficult to expose to a user. Direct visualization of a contour tree in a linked view system has been tried [14, 17] and proven to be a powerful interface for data exploration. However, its usage is still hampered by presenting the topological information as a graph, which is difficult for a user to relate to the structure of a scalar field.

In this paper, we take a classical approach to facilitate the understanding of complex scientific information by developing metaphors that leverage human intuition [18]. In particular, we propose a terrain metaphor for presenting the topological information provided by contour trees together with additional metric information associated with its branches, e.g., the volume associated with each topological feature. Given a contour tree, possibly with metric information, we create a terrain that we call *topological landscape* with the same topology as the original data. This process can be thought of as the inverse operation of the classical construction of a contour tree from a scalar field. In this projection from the topology to a terrain we preserve function values, persistence and nesting of critical points, as well as an additional metric property attached to each arc that we map to the areas of the corresponding regions

This new visual metaphor takes advantage of the fact that humans are naturally trained and very effective in understanding the structure of a terrain. Furthermore, one can combine topological information with additional properties at the level of individual components, something not easily achieved in a graph model. In fact, Carr et al. [14, 17] compute the metric information at the level of individual components, simplify the tree based on this individual information, yet still present it to the user clustered in a spectrum-like interface [7]. The structure of terrains, on the other hand, is easy to grasp and has long been used to illustrate topological concepts. In this paper we show how to harness this intuition to illustrate the behavior of general scalar functions.

Naturally, topological landscapes are not meant to replace the detailed representation of direct rendering methods, other histograms, or more advanced graph models. We believe, instead, that their ability to provide a complete and yet intuitive overview of the data fills an important gap in data exploration process and constitutes a valuable addition to an advanced used interface. In particular, the main contributions of this work include: (i) Introducing the concept of *topological landscapes* as a new metaphor for presenting the structure of a given scalar function f; (ii) Providing a method to construct a terrain whose contour tree is identical to that of f; (iii) Demonstrating how an adaptive refinement of the terrain based solely on 4-8 subdivision rules reproduces a correct topological approximation of f; (iv) Re-balancing contour trees to reduce their height; (v) Providing a multi-scale method for optimizing the mesh such that the area of each feature in the topological landscape is approximately proportional to the desired metric property; and (vi) Presenting examples that demonstrate the expressive power of topological landscapes.

2 Related Work

Even though topological methods have been used by the graphics and visualization community for more than fifteen years [19, 20] and were developed much earlier [21, 22, 23] few techniques aim to improve the presentation of topological information. Some topological structures, such as the Morse-Smale complex and the topological segmentations of vector fields, have natural embeddings [24, 25, 26, 27]. However, they are difficult to compute and in three dimensions (3D) suffer from significant occlusion problems. In scalar field visualization the most widely used topological structure is the contour tree [10, 12], which is an abstract graph without inherent embedding. Pascucci et al. [17] proposed the toporrery to layout and render hierarchical contour trees. The toporrery improves rendering times and overall layout significantly, yet the contour tree remains an abstract graph that is difficult to comprehend. Mizuta et al. [28] represent the nesting structure of a contour tree using a color image of nested rectangles. This method is effective at representing the nesting of features, yet gives no indication of the function values involved and/or the size and importance of the features.

A more common approach to using structural information in visualization is to enhance standard techniques such as isosurfaces and volume rendering [1, 2] with, for example, topological information. Fujishiro et al. [29] rendered translucent isosurfaces representing all possible isosurface topologies and used the result to automate transfer function design [30]. Similar approaches that use topological information to quickly find regions of interested have been proposed by Pascucci and Cole-McLaughlin [11] and Weber at al. [16]. Takahashi et al. [31] proposed using topology based interval volumes to explore datasets.

Another set of methods expose structural information more directly to the user, either in form of histograms or as abstract graphs. Bajaj et al. [7] introduced the contour spectrum, which aids users in the selection of appropriate isovalues by providing a histogram showing isosurface properties, such as area and enclosed volume as a function of the isovalue. Kettner et al. [32] extended this interface to higher dimensions and further considered properties, such as number of contours that are not decomposable. Carr et al. [14, 33] computed similar properties, but do so on a per contour bases. The results are used to guide the simplification of the contour tree, which in turn is used to simplify isosurfaces and provide a more flexible, contour centric interface.

Topological landscapes can been seen as a more powerful version of such interfaces that map topological properties of higher-dimensional datasets to topological properties of two-dimensional terrains [34, 35, 36]. Rather than showing the abstract contour tree (albeit simplified by geometric measures) as proposed by Carr et al. [14] or showing only metric properties like Bajaj et al. [7] topological landscapes combine this information into single intuitive interface.

To reflect a metric property, e.g., the enclosed volume, in the topological landscape we adapt the initial terrain, see Section 4.1, such that the areas of a feature become proportional to the enclosed volume of their corresponding isosurfaces, see Section 4.3. This is related to the construction of continuous cartograms [37, 38, 39] where the goal is to adjust the layout of a given map according to a set of weights with minimal shape distortion. Depending on additional constraints, such as shape or neighborhood preservation, there exist different flavors of the problem and we refer the reader to the paper by Keim et al. [38] for complete definitions and a list of additional references. In general, solutions are based on formulating a non-linear energy function that incorporates the shape and weight error and optimizing this energy. House and Kocmoud [37] used a large spring system for optimization that produces good results but is computationally expensive. Keim et al. [38] divided their entire map into sections between a set of scanlines and scaled each section either along or perpendicular of each scanline. The restriction to small localized updates makes this algorithm comparatively fast. However, the quality is strongly dependent on the placement of the scanlines and the best results are achieved with manual placement. Finally, Heilmann et al. [39] reduced the dimensionality of the problem by restricting all regions to be rectangular. The resulting reduced problem is solved using genetic programming.

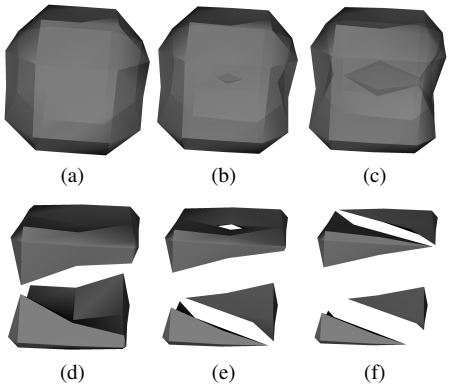

Fig. 1. The number of contours of an isosurface changes as the isovalue is swept thought the function range: After hitting a minimum, one contour is created (a). A new contour inside the existing contour is created around a second minimum. For a higher value, these two contours merge into a single contour (c). As the isovalues increases, this contour splits in two independent contours (d), which in turn both split (e), (f) until there are four contours. As the isovalue is further increased, these four contours are destroyed. We do not track genus changes that occur between split events, e.g., the hole forming in the upper component (d), (e). (Dataset courtesy of Hamish Carr, University College Dublin.)

While all these solutions are related, and in particular the RecMap approach seems applicable (since all our initial regions are squares), there exist important details that make them impractical in our case. To preserve contour tree topology, our neighborhood constraints are absolute. As shown in Section 4, the initial layout is carefully chosen to preserve topology and two regions changing neighborhood status (either becoming neighbors or no longer being neighbors) is likely to change topology. Second, our regions are defined hierarchically and therefore even more inter-dependent than in the standard cartography problem. Finally, we aim to preserve the initial triangulation (to ensure fast rendering) without fold-overs or T-junctions. This adds a large number of additional constraints to the interior of each region that do not exist in the case of traditional continuous cartographs.

Instead, we use a multi-scale approach similar to the graph drawing techniques developed by Harel and others [40, 41, 42]. Their method creates a graph layout by first coarsening a graph, solving the layout for the simplified graph and then progressively re-refiniing and re-optimizing the layout at each resolution level. As discussed in Section 4.3, our problem is particularly suited for such a multi-scale approach since our constraints as well as the mesh are by construction organized hierarchically. Furthermore, for each atomic step one can calculate the optimal position of a vertex making complicated energy minimizations unnecessary.

To represent a topological landscape we are using an adaptive 4-8 mesh [43, 44, 45]. In particular, we chose SOAR [45], because its simplification metrics are compatible with persistence, its recursive refinement scheme implies a recursive method for translating the branch decomposition into a terrain, and its simplicity.

3 Background

In this section we define contour trees, show how they are simplified, and discuss branch decompositions as one method of representing contour trees. For a more detailed discussion of the underlying theory we refer the reader to [23].

Let $\mathbb{M}$ be a manifold with boundary and $f : \mathbb{M} \to \mathbb{R}$ a scalar function on $\mathbb{M}$. One of the fundamental techniques to understand the behavior of f is to extract its *isosurfaces*, see Figure 1. Given an *isovalue* v the isosurface of f at v is defined as the pre-image of v on $\mathbb{M}$: $f^{-1}(v)$. The connected components of an isosurface are called *contours*. The contour tree of f is the graph obtained by continuously contracting every contour of f into a point, see Figure 2(a). Contour trees encode

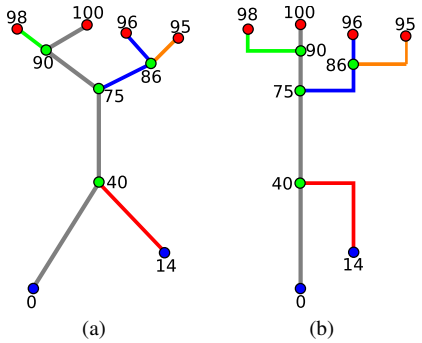

Fig. 2. The contour tree describing the evolution of contours in Figure 1 (a) and its multi-resolution representation as branch decomposition (b): (a) Two nodes at the bottom represent the two minima around which the initial two surface components (Figure 1 (a) and (b)) are created. The four degree three nodes represent the merge and split events of Figures 1 (c), (d), (e) and (f). The four nodes at the top correspond to the maxima, where the four contours of Figure 1 (f) are destroyed. (b) The branch decomposition is a multi-resolution representation of that same contour tree. It is obtained by selecting a root branch connecting maximum-minimum pair of the contour tree that maximizes a metric (here persistence). Branches connecting an extremum (degree one nodes) to a saddle (degree three and higher nodes) are added in an order based on some importance measure (here persistence). The further down a branch is located in the parent-child hierarchy, the less important its contribution to the topological description of the dataset is deemed.

the evolution of contours as the isovalue is swept through the range of f. Bottom leaf nodes of the contour tree correspond to minima of f (where contours are created), internal nodes to saddles (where contours merge or split), and top leaves correspond to maxima (where contours disappear). Because of this relation between the contour tree and the global structure of the isosurfaces of f, contour trees have been used extensively in visualization.

The contour tree of a complicated and/or noisy function can become too large to be studied (or even displayed) directly [14, 17]. A contour tree is simplified by successively removing leaves and their corresponding branches, which is equivalent to a cancellation [24] of the two critical points corresponding to the end points of the deleted branch. Another view is that a cancellation represents removing the mountain top (or filling in the valley) that the leaf symbolizes. To rank cancellations, different metrics are suitable. We choose persistence [46]—the absolute difference in function value spanned by a branch—since it corresponds to topological simplification. This correspondence is crucial in creating a landscape whose topology reflects that of a given contour tree.

We also use the sequence of simplifications to build a branch decomposition [17] of the contour tree, see 2(b). Each time a branch $B0$ attached to a branch $B1$ is canceled we say that $B1$ is a parent of $B0$ and $B0$ is a child of $B1$. This process provides us with a hierarchical representation of a contour tree that can be directly translated into a SOAR mesh hierarchy.

4 Algorithm

Given a scalar function f, we first extract its contour tree using any of the standard techniques [12, 11] and construct its branch decomposition [17]. Following the hierarchy of the branch decomposition, we recursively create a terrain with the same topological structure as f.

4.1 Terrain construction

Starting with the root, each branch is mapped to a region of the terrain, which is subsequently refined to provide space for all children of the branch. This process is applied recursively to each child branch resulting in the final terrain. Each branch creates either a peak (for a maximum) or a valley (for a minimum) surrounded by a ring of vertices at the function value of the saddle. The ring ensures that the saddles of the terrain have the same value as their corresponding saddles of f.

Initially, the root branch is modeled as a single peak, see Figure 3(a). The vertex at the center (marked red) represents the maximum and the ring of vertices comprising the boundary (marked blue) represents the minimum of the root branch. While one level of the SOAR hierarchy could be used to represent this peak, we use two levels to simplify the recursive construction. If the root branch has no children the terrain is complete. Otherwise, we refine the mesh to create sufficient space to fit the same 3-by-3 vertex configuration for each child. As shown in Figure 3(b), three additional refinement steps create sufficient space to place four children. However, regions share vertices (marked yellow) which interferes with assigning appropriate function values (see below). Therefore, we refine one more level, see Figure 3(c) creating space for four isolated child branches. If more than four children must be placed, we further subdivide the mesh two levels at a time, quadrupling the number of available spots each time, see Figure 3(d).

Assignment of child branches to the available spots depends on the function values of their nodes. Each branch (except for the root) connects an extremum to a saddle. We assign the function value of the extremum to the center vertex of the corresponding region and the function value of the saddle to the eight remaining vertices. This scheme automatically creates peaks for maxima and valleys for minima. Children are sorted according to the function value of their saddles and placed in decreasing order in a spiral layout, see Figure 3(e). Finally, all vertices between children except the center and boundary of the parent are assigned the average function value of their neighbors. We apply this scheme recursively until all branches of the contour tree have been placed.

As children are refined, vertices may need to be added to their parent region to avoid T-junctions. We determine these additional vertices by temporarily considering a mesh uniformly subdivided to its highest resolution (as required to place recursively all branches of the hierarchy) and using the SOAR algorithm to determine the fewest number of vertices that must be added to create a consistent mesh. Note that for two neighboring child branches with the same saddle value special care must be taken to avoid under-refining the mesh. Conceptually, all new vertices are the result of an edge bi-section and we use linear interpolation along this edge to assign their function values.

This hierarchical construction scheme ensures that the isolines at the level of the parent-saddle enclose all isolines of its children. Therefore, the nesting structure of all isolines in the terrain is identical to that of f's isosurfaces. Furthermore, the layout guarantees that there are no extraneous extrema since all vertices not specifically designed as critical points have at least one higher and one lower neighbor. It follows that there exist no extraneous saddles and the contour tree of the terrain is identical to that of f.

The topological landscape not only preserves the topological structure of f but the function values are carefully chosen to also preserve the persistence of the features. Finally, by mapping the hierarchical branch decomposition directly to the SOAR hierarchy one can perform a correct topological simplification of the landscape by simply rendering a coarser mesh. As the center vertex of each branch is removed, the branch is rendered as a flat region at the height of the saddle. This corresponds to removing a mountain top or filling in a valley canceling the feature.

4.2 Contour tree re-balancing for better space utilization

Using a recursive terrain construction scheme exposes a problem of the branch decomposition that is less obvious in graph-based representations, such as the toporrery. A branch decomposition often results in a very "deep" tree with many hierarchy levels. While some of these levels are a result of contour nesting properties, the use of symbolic perturbation and subsequent simplification based on persistence can contribute significantly to the depth of the hierarchy. To alleviate this problem we employ a recursive branch re-balancing step. Starting with one hierarchy level above the leaves we check if the saddle value of a child node is close to the saddle value of its parent branch. All

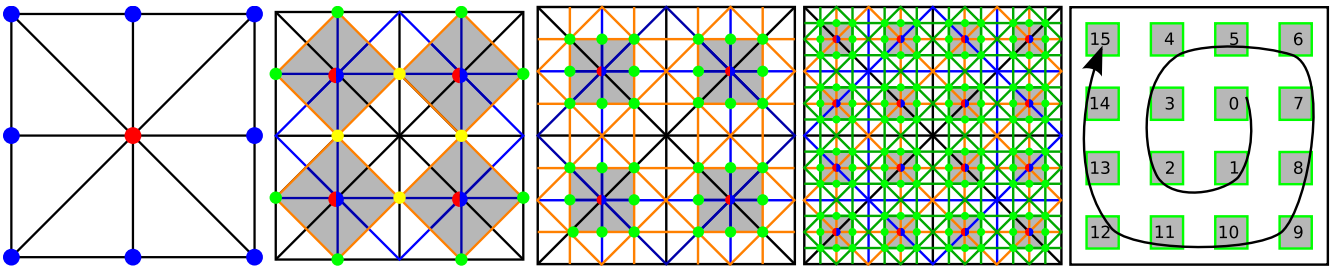

Fig. 3. Constructing a terrain from the branch decomposition compatible with the SOAR hierarchy. (a) The root branch is mapped to a SOAR hierarchy consisting of two levels with the maximum at the center and the minimum at the boundary of the terrain. The resulting mesh is refined further to create space for placing the children of the current branch. Adding three levels of refinement (b) adds sufficient space to place the extrema. However, locations for the saddles of a branch are ambiguous, as indicated by the yellow vertices shared between two children. Adding four levels of extra refinement (c) creates sufficient space for placing four child branches (highlighted gray in the image) and leaving a "buffer" zone between them. Furthermore, regions for placing the child branches are identical to the set-up for the initial branch. The vertex corresponding to the extremum (marked red/blue) is surrounded by eight vertices linked to the saddle (marked green). (d) Every two levels of extra refinement quadruple the number of available spots for children. (e) Children are placed in order of deceasing saddle value in a spiral layout.

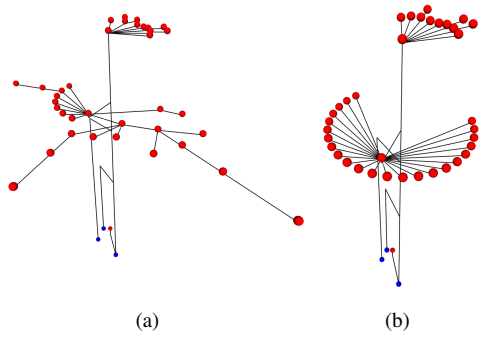

Fig. 4. To achieve better space utilization in the recursive terrain generation scheme, we re-balance the branch decomposition. Child branches whose saddles are close (in function value) to that of its parent become its siblings. This reduces the depth of the hierarchy and in particular removes artifacts due to symbolic perturbation. The image shows the branch decomposition of the nucleon dataset in toporrery layout (a) and the re-balanced branch decomposition, where all branches with zero persistence are moved to their parent branch (b).

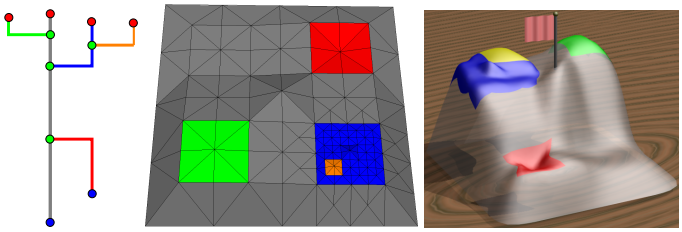

Fig. 5. Constructing a terrain (b) form a branch decomposition (a) of the model shown in Fig. 1 and distributing volumes associated with the branches (c). Given a root branch (gray) the volume of its children is assigned to their respective patches (see Section4.1) and its own volume to the surrounding region. This scheme is followed recursively within each sub-tree. (c) Final terrain after adjusting areas according to the given volumes.

children whose saddle value is within a user-specified threshold become siblings of their parent (i.e., are moved up one hierarchy level). Subsequently, this process is continued for the next higher hierarchy levels until the root branch is reached. Potentially, a child branch can move up several levels in the hierarchy, if saddle values are very close. Figure 4 illustrates the effect of removing artifacts due to symbolic perturbation. All children with identical saddle values as their parent branch are moved up in the hierarchy. Clearly, the topological structure becomes much more obvious. While this approach is particularly useful for removing symbolic perturbation artifacts, higher re-balancing thresholds can be appropriate for some datasets. However, the resulting branch decomposition no longer reflects the exact nesting properties of contours.

4.3 Metric-based distortion

The previous sections describe how to create a terrain whose topology reflects that of a given contour tree. Furthermore, the persistence metric is preserved by choosing appropriate height values. Nevertheless, there exists a perceptual problem, namely that the area (in the xy-plane) corresponding to a branch only depends on its level in the hierarchy. This can result in small patches with high persistence creating thin spikes in the terrain. Not only can such features be difficult to see, but they are traditionally associated with being noise or outliers. In this context, however, this association might be misleading as the hierarchy level, in general, is not necessarily related to the significance of a topological feature (especially when considering relative significance between branches on the same level).

To alleviate this problem and to further increase the expressive power of topological landscapes we treat the area assigned to a branch as an additional variable to be optimized. This allows us to preserve not only topology and persistence, but also another metric property chosen by the user. In our examples, we use the volume associated with a branch, as this is another well studied metric to decide the significance of a contour [14]. As discussed above, each branch of a contour tree corresponds to a single contour being swept through the volume. Integrating the area of the contour through its life time results in the volume we associate with the corresponding contour tree branch. We approximate this volume by the number of vertices passed over by the contour. The volume is distributed to the terrain as shown in Figure 5(b).

Each branch is assigned a square patch. Wihtin this patch, the volume of its children is assigned to their respective sub-patches and the volume of the root branch of this (sub-)tree to the triangles covering the remaining area. There are two important details for practical application: First, within a given patch the volume is distributed according to the initial area of each triangle. The original triangle area is tied into the valences of surrounding vertices and observing the local area ratios prevents problems in the later optimization. Second, instead of the original volume V we distribute $V^{2/n}$, where n is the dimension of the function domain. This compensates for the well known perceptual difference in judging relative areas compared to relative (hyper-)volumes.

Once the volume has been distributed, we create a standard quad-mesh hierarchy in which leaf nodes correspond to triangles (or triangle-halves) and are assigned the corresponding volume. Internal nodes are assigned the sum of the volume of all their children, see Figure 6(a). Subsequently, we iteratively adapt vertex positions (but not

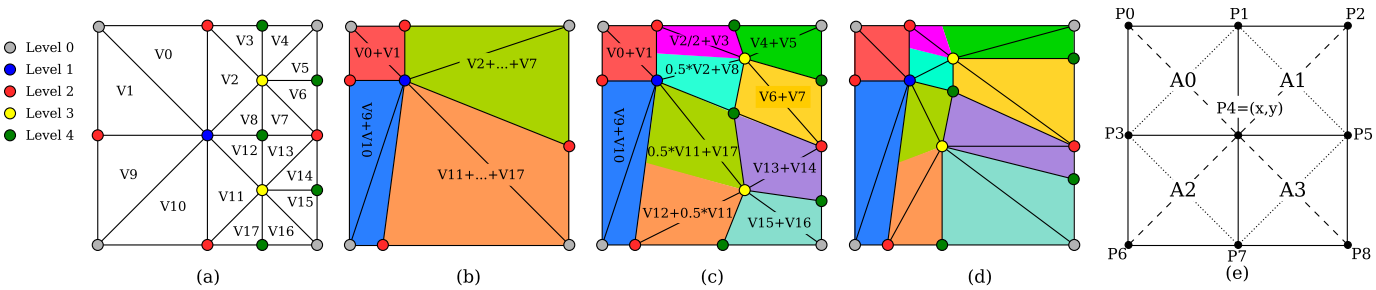

Fig. 6. (a) Adaptive mesh with volumes assigned to its triangles; (b) The mesh of (a) after the optimization of level one and two vertices with the colors indicating the appropriate quad-tree cells and their associated target volumes; (c) The mesh of (b) after the nodes on level three and four have been introduced as the mid points of their respective edges. Note, that not all boundaries between quadtree cells are edges in the mesh; (d) The final mesh after optimization; (e) Prototypical vertex neighborhood for vertices on even (dotted) and odd levels (dashed).

Let $TL = (V,T)$ be a topological landscape
lvl = 1;
$stepSize$ = 0.25;
$epsilon$ = convergence threshold;
`while` lvl < MAXLEVEL(TL)
  $delta$ = 2*$epsilon$;
  `while` $delta > epsilon$
    `forall` i in $\{0,1\}$
      `forall` v in VERTICESONLEVELS($TL, \{lvl+i\}$)
        $optimal[v]$ = GETOPTIMALPOSITION(v);
      `forall` v in VERTICESONLEVELS($TL, \{lvl+i\}$)
        v = STEPTOWARDS($optimal[v]$,$stepSize$);
    $delta$ = MAXSTEPLENGTH($optimal$);
  `endwhile`
  `forall` v in VERTICESONLEVELS($TL, \{lvl+2, lvl+3\}$)
    v = AFFINEPROJECTION(v);
  $lvl = lvl + 2$;
`endwhile`

Fig. 7. Pseudo-code to adjust the area rations of a topological landscape to the metric properties of its branches.

the connectivity) in a coarse-to-fine manner aiming to achieve area ratios corresponding to the given volumes. Figure 7 shows the complete algorithm as pseudo-code using the nomenclature of Figure 6(a). In particular, we assume all vertices of the terrain to be assigned levels according to the SOAR hierarchy.

We adapt the mesh two hierarchy levels at a time considering only vertices of the current level or coarser. Each vertex independently computes its optimal position (see below), and then all vertices on a given level take a step towards their optimal position. Steps are constrained to maintain the current triangulation considering only vertices on their own level or above. Once optimization has converged for a given level, we lift the vertices of the next two finer levels to their appropriate affine position, placing them at the midpoints of the appropriate edges (given by the SOAR hierarchy). Figure 6(a)-(d) show an example of an adaptive mesh being optimized by our algorithm. While we have no formal proof of convergence, the algorithm quickly converged in all our tests using a step size of 0.25.

To compute the optimal position of a vertex, we consider it as the center of four quad-tree cells, each with its target volume attached. There exist two cases: moving vertices on even or on odd levels as their connectivity differs, see Figure 6(e). In the following we derive the formula for odd levels following the nomenclature of Figure 6(e). First, we re-scale the volumes corresponding to the four quadrants $A0-A3$ by the sum of their current areas to obtain the target areas t_0-t_3 we aim for. The area of, for example, the left upper quadrant can be calculated as.

$$\begin{aligned} 2*area_0 &= (p_{1x}-x)*(p_{0y}-y)-(p_{1y}-y)*(p_{0x}-x) \\ &\quad +(p_{0x}-x)*(p_{3y}-y)-(p_{0x}-x)*(p_{3y}-y) \\ &= 2a_0x+2b_0y+2c_0\,. \end{aligned}$$

With the corresponding calculations for the other three quadrants the squared area defect with respect to $p_4 = (x,y)$ is given by

$$\text{areaDefect}(x,y)^2 = \sum_{i=0}^{3}(a_ix+b_iy+c_i-t_i)^2.$$

Finally, we solve for $\nabla\text{areaDefect}(x,y)^2 = (0,0)$ to find the optimal position of the center vertex. We found stepping toward the optimal solution to perform significantly better than the more traditional step in gradient direction. Two special cases exist: boundary vertices, and vertices for which p_1, p_3, p_5, and/or p_7 do not exist. In the former case, optimization only considers those quadrants that are part of the mesh. In the later case, missing vertices are assumed to lie at the midpoint of their respective edges. Note that the optimal position of a boundary vertex always lies on the boundary as well. Therefore, optimization preserves the quadratic foot-print of the terrain. Figure 5(c) shows the resulting terrain for the example dataset of Figure 1.

Since we do not change connectivity, the optimized mesh can still be rendered using the SOAR algorithm. Nevertheless, for high quality rendering one must also guarantee that locally coarsening the mesh does not create inverted normals: Assume that an adaptively refined mesh contains a triangle with an inverted normal. Within this triangle consider the vertex on the highest SOAR level. The connectivity of this vertex is identical to that seen during its optimization. Since at the optimization stage triangles were explicitly maintained to preserve their orientation, this is a contradiction. If adaptive rendering is unimportant, we add an additional smoothing stage, aiming to improve triangle quality within each patch while maintaining patch boundaries.

We measure our error per branch as the difference in percent between final area and target area. The maximal error of a single branch in all examples is 8.8% for the engine dataset and the largest average error (error weighted by size of the area) is 1.4% for the hydrogen atom dataset.

4.4 Discussion

The algorithm described above constructs a topological landscape for any given contour tree. We can illustrate any scalar function defined on a simply connected domain using this new metaphor. In particular, contour trees can be computed from n-dimensional functions [12], allowing us to illustrate topology of functions of arbitrary dimension. However, the main source of datasets with dimension larger than three are time-dependent fields, which require special treatment. While it is possible to treat a three-dimensional time-dependent function as being four-dimensional, current experience suggests that results are difficult to understand and that constructing a time series of three-dimensional fields is more advisable. Constructing animated landscapes to handle these cases will be one focus of future research. A limitation of our approach is that the topological landscape of a two-dimensional function (i.e. a terrain), in general, will look very different from the original. However, since terrain topology is readily apparent in standard renderings we find this to be a small limitation.

As with many non-linear and iterative optimization schemes the run time complexity is data and parameter dependent. Therefore, we cannot provide a formal bound on the overall complexity and using unfa-

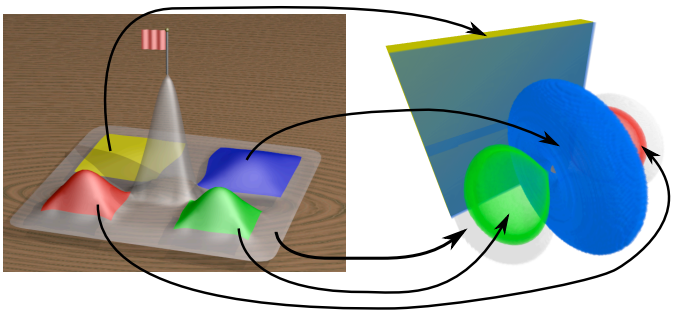

Fig. 8. Hydrogen atom datasets with average volume-to-area projection error of 1.1%. (Left) The topological landscape showing, in addition to the main mountain, four structures of large area. Two have high persistence (red, green), one has medium persistence (blue) and one is nearly flat (yellow). (Right) Volume rendering using a similar color scheme to increase correlation between the two views.

vorable parameters (e.g., an overly large step-size) the algorithm might not converge. Memory requirements depend hierarchy depth and corresponding SOAR mesh resolution. Currently, our implementation is not optimized and requires a temporary mesh of uniform resolution. We are working on an optimized implementation that eliminates this constraint.

5 Results

We have tested our technique on a set of classical datasets widely used by the visualization community. In general, we believe that the metaphor proposed here constitutes a good complement to classical techniques such as volume rendering or isocontouring. The topological landscape provides an overview of the structures present in the data showing their relationships in terms of function value, topological nesting, persistence, and amount of space occupied.

We stress that this concept is meant to be a complement to existing approaches since important properties, such as the distance between features, are obviously lost in the projection from the 3D volume to the terrain.

Below, we show a series of examples and comment on our perceptual experience with them. Naturally, this is just anecdotal evidence and by no means a substitute for a rigorous user study. For example, during development we arrived several times at the conclusion that a topological landscape was incorrect, probably due to an implementation error. Only after further exploration of the data we realized that our intuition regarding the overall structure of the dataset was inaccurate. The topological landscape was in fact correcting our misconception about the data. Overall, our initial practical experience has reinforced our belief that this new metaphor can be an effective addition to the general toolbox of scientific visualization components.

In all the examples below we have re-balanced the contour tree as described in the previous section to avoid nesting artifacts in the presentation and filtered low persistence topological noise. In all images we assign a different color to each hill and valley to visually separate features. The main mountain, not necessarily ranging from global minimum to global maximum, is presented in a semi-transparent gray to facilitate the visualization of deep valleys and its center is marked by a flag. In general, this metaphor has a bias in favor of maxima since the minima may be hidden at the bottom of a valley. In cases where this may be a problem, we provide a secondary view using the same main mountain but with nested hills and valleys flipped for visual inspection, see Figure 12. We note that this not the same as providing a completely inverted contour tree, but having an identical main mountain makes it easier to correlate information between two landscapes. For each dataset we report the average error that we commit when projecting the volume of a region onto the corresponding area in the topological landscape.

Our first example is the hydrogen atom dataset (data downloaded from `http://www.volvis.org/`). Figure 8 shows the topological landscape together with a volume rendered image of the

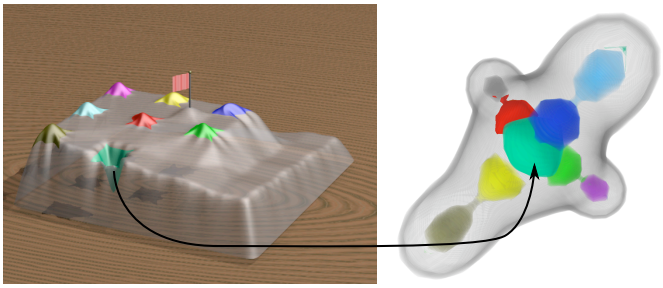

Fig. 9. Methane dataset: with average volume-to-area projection error of 1.5%. (Left) Topological landscape. (Right) Volume rendering using the same color scheme.

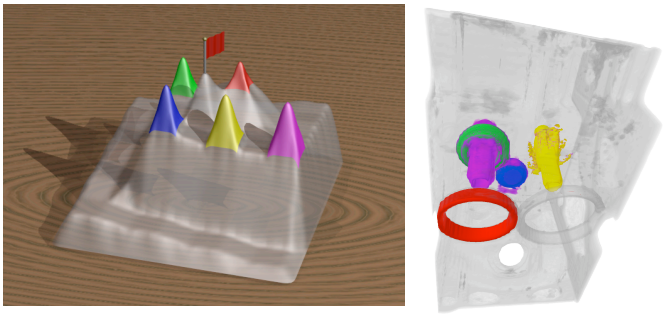

Fig. 10. Engine dataset with average volume-to-area projection error of 1.3%. (Left) Topological landscape. (Right) Volume rendering using the same color scheme.

data, assigning different transfer functions to separate topological features [47]. Notice how the landscape helps appreciate the fact that the two lobes in the hydrogen orbital (red, green) have larger function range than the toroidal ring in the middle (blue), although the ring takes a larger volume. One can also clearly see a large region at nearly zero persistence (yellow), which is probably an artifact from the construction process. In the past we completely missed this region due to its low function value even though it occupies a large portion of the volume. This is one case where we initially thought the landscape to be incorrect, while instead our standard exploration had simply missed this feature. A more important insight has been the size of the toroidal ring. From the 3D images it is not easy to realize that the ring becomes larger than the two lobes before merging with the rest of the volume. The explanation is that it merges from its interior and it is therefore difficult to capture this event using a 3D view. In the topological landscape we clearly see that the ring becomes larger than the two lobes before the actual merge event occurs.

The next example is the methane dataset shown again as topological landscape together with a volume rendering using a similar color scheme (see Figure 9). In this case it is interesting to see how the landscape immediately clarifies, which feature is built around a minimum/maximum, something not evident from the volume rendering. In particular, the main feature related to the carbon atom is associated with a large minimum, while each hydrogen atom is associated with two maxima. Clearly, the combination of these two images, even without interactive exploration, provides a better explanation than either of them independently.

Figures 10 and 11 show similar illustrations for the engine and nucleon datasets. In both cases considerations similar to those for the hydrogen atom and methane data sets apply. The topological landscape immediately relates which regions are associated with maxima and minima and their persistence. Moreover, one understands better, which features take a large portion of the volume even though the 3D rendering does not contain that information explicitly due to occlusion.

The last three examples are the silicium (Figure 12), neghip (Figure 13) and fuel (Figure 14) datasets. In these cases, we show topological landscapes together with a more traditional volume rendering using a global transfer function. This leads to a reduced correlation between the two visualizations and therefore hampers the benefit of the

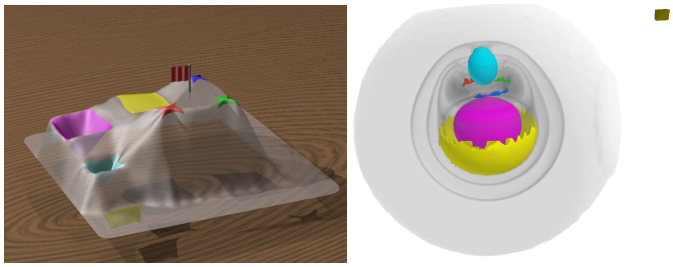

Fig. 11. Nucleon dataset with average volume-to-area projection error of 0.6%. (Left) Topological landscape. (Right) Volume rendering using the same color scheme

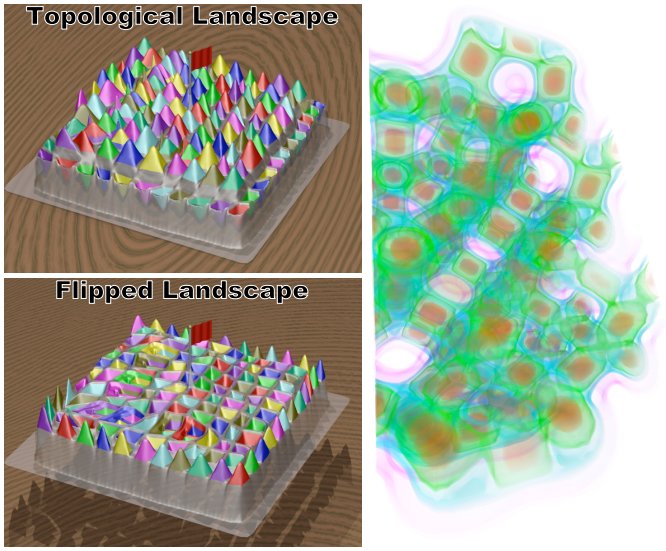

Fig. 12. Silicium dataset with average volume-to-area projection error of 1.0%. (Top left) Topological landscape. (Bottom left) Flipped landscape. (Right) traditional volume rendering.

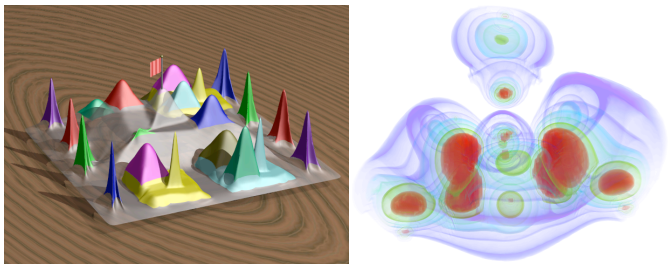

Fig. 13. Neghip dataset with average volume-to-area projection error of 1.9%.

simultaneous presentation. Nevertheless, one can derive information that would not be obvious otherwise. In the case of the silicium data set, for example, one notices that the structures forming the lattice of the crystal are a set of maxima and minima all of similar persistence and all occupying similar volumes. This is easy to see in the topological landscape and its flipped counterpart. The volume rendering complements this information with a sense of the geometric shape of the actual crystal. For the neghip dataset the topological landscape reveals that many features that are small in terms of volume but span a large function range. Their geometric distribution is highlighted by the volume rendering.

In general, we find that this new metaphor has the potential to greatly improve the effectiveness of classical visualizations, such as volume rendering or isocontouring, since it conveys complementary information increasing the ability of users to fully understand the inner structure of the data with much less exploration. Furthermore, we believe that this would be very effective in static illustrations where any information about the global structure of the data becomes extremely valuable.

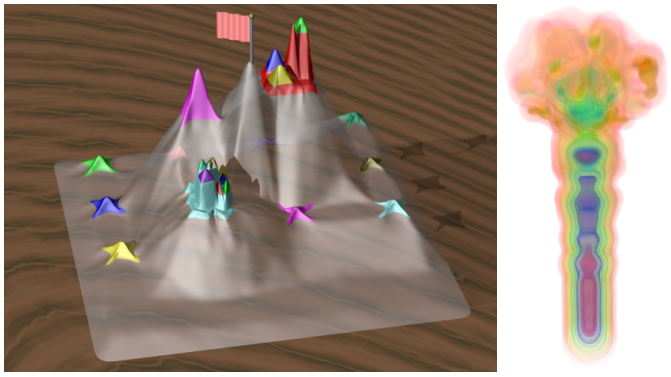

Fig. 14. Fuel dataset with average volume-to-area projection error of 4.1%.

6 Conclusions and Future Work

We have introduced topological landscapes as a new metaphor to present topological information. Starting from a scalar function f we show how to project f to a terrain preserving the topological structure, function values, and persistence, as well as the volume associated with each feature. Furthermore, the landscape is encoded as a hierarchical mesh which can be efficiently rendered and whose hierarchy corresponds to the persistence based simplification of f. Overall, topological landscapes, though currently limited to scalar functions that have an associated contour tree, i.e., which are defined on a simply connected domain, provide a powerful new interface to complement existing techniques.

Going forward, we are experimenting with metrics [48, 49, 14] beside the enclosed volume to further increase the flexibility of topological landscapes. Furthermore, we are investigating different layouts to reduce the amount of "empty" space and are considering changing the volume assignment within each branch to reflect the volume above/below its highest/lowest saddle in separate regions.

We further plan to explore linking multiple representations of a dataset, such as contour tree display, a topological landscape and a volume rendered image and/or an isosurface display. This combination should be very effective when exploring the structure of a dataset. Adding the ability to extract contour/height lines on a topological landscape, along with linking this contour with the corresponding isosurface of the volumetric dataset will facilitate the ability to detect higher level topological features.

Acknowledgements

We would like to thank Hamish Carr and Scott Dillard for making available the code used to compute contour trees and the branch decomposition and Peter Lindstrom for making available the SOAR engine publicly. We would also like to thank Hamish Carr, Scott Dillard, Bernd Hamann and Peter Lindstrom for fruitful discussions. Finally, we thank the VisIt (`http://www.llnl.gov/visit/`) development team. This work was supported by the LDRD "Efficient and Reliable Data Exploration Via Multi-Scale Morse Analysis and Combinatorial Information Visualization" awarded by the Lawrence Livermore National Laboratory and the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 through the Scientific Discovery through Advanced Computing (SciDAC) program's Visualization and Analytics Center for Enabling Technologies (VACET). This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

[1] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, vol. 21, no. 4, pp. 163–169, Jul. 1987.

[2] M. Levoy, "Display of surfaces from volume data," *IEEE Computer Graphics and Applications*, vol. 8, no. 3, pp. 29–37, May 1988.

[3] W. Hibbard, C. R. Dyer, and B. Paul, "Display of scientific data structures for algorithm visualization," in *VIS '92: Proceedings of the 3rd conference on Visualization '92*, 1992, pp. 139–146.

[4] H. Akiba, N. Fout, and K.-L. Ma, "Simultaneous classification of time-varying volume data based on the time histogram," in *Proceedings of Eurographics Visualization Symposium*, May 2006, pp. 1–8.

[5] K. Stockinger, E. W. Bethel, S. Campbell, E. Dart, , and K. Wu, "Detecting Distributed Scans Using High-Performance Query-Driven Visualization," in *SC '06: Proceedings of the 2006 ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis*, October 2006.

[6] J. Kniss, G. Kindlmann, and C. Hansen, "Multidimensional transfer functions for interactive volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 270–285, 2002.

[7] C. L. Bajaj, V. Pascucci, and D. R. Schikore, "The contour spectrum," in *Proc. IEEE Visualization '97*, Oct. 19–24 1997, pp. 167–173.

[8] F.-Y. Tzeng, E. Lum, and K.-L. Ma, "An intelligent system approach to higher-dimensional classification of volume data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 3, pp. 273–284, 2005.

[9] F.-Y. Tzeng and K.-L. Ma, "Inelligent feature extraction and tracking for large-scale 4d flow simulations," in *Proceedings of Supercomputing 2005*, 2005.

[10] M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. R. Schikore, "Contour trees and small seed sets for isosurface traversal," in *Proceedings of the 13th Annual ACM Symposium on Computational Geometry (SoCG)*, 1997, pp. 212–220.

[11] V. Pascucci and K. Cole-McLaughlin, "Parallel computation of the topology of level sets," *Algorithmica*, vol. 38, no. 2, pp. 249–268, Oct. 2003.

[12] H. Carr, J. Snoeyink, and U. Axen, "Computing contour trees in all dimensions," *Computational Geometry – Theory and Applications*, vol. 24, no. 2, pp. 75–94, Feb. 2003.

[13] G. H. Weber, G. Scheuermann, and B. Hamann, "Detecting critical regions in scalar fields," in *Data Visualization 2003 (Proceedings of VisSym 2003)*, 2003, pp. 85–94.

[14] H. Carr, J. Snoeyink, and M. van de Panne, "Simplifying flexible isosurfaces using local geometric measures," in *Proc. IEEE Visualization 2004*, Oct. 2004, pp. 497–504.

[15] V. Pascucci, "On the topology of the level sets of a scalar field," in *"Proceedings of the 13th Canadian Conference on Computational Geometry"*, August 2001, pp. 141–144.

[16] G. H. Weber, G. Scheuermann, H. Hagen, and B. Hamann, "Exploring scalar fields using critical isovalues," in *Proc. IEEE Visualization 2002*, 2002, pp. 171–178.

[17] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli, "Multi-resolution computation and presentation of contour trees," Lawrence Livermore National Laboratory, Tech. Rep. UCRL-PROC-208680, 2005, preliminary version appeared in the proceedings of the IASTED conference on Visualization, Imaging, and Image Processing (VIIP 2004), 2004, pp.452-290.

[18] J. Cat, "On understanding: Maxwell on the methods of illustration and scientific metaphor," *Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics September 2001*, vol. 32, no. 3, pp. 395–441, 2001.

[19] Y. Shinagawa, T. L. Kunii, and Y. L. Kergosien, "Surface coding based on Morse theory," *IEEE Computer Graphics and Applications*, vol. 11, no. 5, pp. 66–78, 1991.

[20] A. V. Gelder and J. Wilhelms, "Topological considerations in isosurface generation," *ACM Transactions on Graphics*, vol. 13, no. 4, pp. 337–375, Oct. 1994.

[21] M. Morse, "Relations between the critical points of a real functions of n independent variables," *Transactions of the American Mathematical Society*, vol. 27, pp. 345–396, July 1925.

[22] G. Reeb, "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*, vol. 222, pp. 847–849, 1946.

[23] J. W. Milnor, *Morse Theory*. Princeton University Press, May 1963.

[24] H. Edelsbrunner, J. Harer, and A. Zomorodian, "Hierarchical Morse-Smale complexes for piecewise linear 2-manifolds," *Discrete Comput. Geom.*, vol. 30, pp. 87–107, 2003.

[25] A. Gyulassy, V. Natarajan, V. Pascucci, P.-T. Bremer, and B. Hamann, "Topology-based simplification for feature extraction from 3D scalar fields," *IEEE Transactions on Computer Graphics and Visualization (TVCG)*, vol. 12, no. 4, pp. 474–484, 2006.

[26] J. L. Helman and L. Hesselink, "Visualizing vector field topology in fluid flows," *IEEE Computer Graphics and Applications*, vol. 11, no. 3, pp. 36–46, May/Jun. 1991.

[27] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P.Seidel, "Saddle connectors - An approach to visualizing the topological skeleton of complex 3d vector fields," in *Proc. IEEE Visualization '03*, 2003, pp. 225–232.

[28] S. Mizuta, Y. Suwa, T. Ono, and T. Matsuda, "Description of the topological structure of digital images by contour tree and its application," Institute of Electronics, Information and Communication Engineers, Tech. Rep., 2004.

[29] I. Fujishiro, Y. Takeshima, T. Azuma, and S. Takahashi, "Volume data mining using 3D field topology analysis," *IEEE Computer Graphics and Applications*, vol. 20, no. 5, pp. 46–51, Sep./Oct. 2000.

[30] I. Fujishiro, T. Azuma, and Y. Takeshima, "Automating transfer function design for comprehensible volume rendering based on 3D field topology analysis," in *Proc. IEEE Visualization '99*, Oct. 25–29, 1999, pp. 467–470.

[31] S. Takahashi, I. Fujishiro, and Y. Takeshima, "Interval volume decomposer: A topological approach to volume traversal," in *Visualization and Data Analysis 2005 (Proceedings of the SPIE)*, 2005.

[32] L. Kettner, J. Rossignac, and J. Snoeyink, "The safari interface for visualizing time-dependent volume data using iso-surfaces and contour spectra," *Computational Geometry: Theory and Applications*, vol. 25, no. 1-2, pp. 97–116, 2003.

[33] H. Carr, "Topological manipulation of isosurfaces," Ph.D. dissertation, University of British Columbia, Apr. 2004.

[34] J. L. Pfaltz, "Surface networks," *Geographical Analysis*, vol. 8, pp. 77–93, 1976.

[35] L. R. Nackman, "Two-dimensional critical point configuration graphs," *IEEE Transcactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 4, pp. 442–450, 1984.

[36] S. Takahashi, T. Ikeda, Y. Shinagawa, T. L. Kunii, and M. Ueda, "Algorithms for extracting correct critical points and constructing topological graphs from discrete geographical elevation data," *Computer Graphics Forum*, vol. 14, no. 3, pp. 181–192, 1995.

[37] D. H. House and C. J. Kocmoud, "Continuous cartogram construction," in *IEEE Visualization*, 1998, pp. 197–204.

[38] D. A. Keim, S. C. North, and C. Panse, "Cartodraw: A fast algorithm for generating contiguous cartograms," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, no. 1, pp. 95–110, 2004.

[39] R. Heilmann, D. A. Keim, C. Panse, and M. Sips, "Recmap: Rectangular map approximations," in *IEEE Symp. on Information Visualization*, 2004, pp. 33–40.

[40] R. Hadany and D. Hardel, "A multi-scale method for drawing graphs nicely," *Discrete Applied Mathematics*, vol. 113, no. 3-21, 2001.

[41] D. Harel and Y. Koren, "A fast multi-scale method for drawing large graphs," in *Graph Drawing: 8th International Symposium (GD'00)*, 2000, pp. 183–196.

[42] Y. Koren, L. Carmel, and D. Harel, "Ace: A fast multiscale eigenvectors computation for drawing huge graphs," in *IEEE Symposium on Information Visualization*, 2002, pp. 137–144.

[43] M. A. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein, "Roaming terrain: Real-time optimally adapting meshes," in *Proc. IEEE Visualization '97*. IEEE, Nov. 1997, pp. 81–88.

[44] L. Velho and D. Zorin, "4–8 subdivision," *Computer-Aided Geometric Design*, vol. 18, no. 5, pp. 397–427, 2001, special Issue on Subdivision Techniques.

[45] P. Lindstrom and V. Pascucci, "Terrain simplification simplified: A general framework for view-dependent out-of-core visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 3, pp. 239–254, 2002.

[46] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," *Discrete Comput. Geom.*, vol. 28, pp. 511–533, 2002.

[47] G. H. Weber, S. Dillard, H. Carr, V. Pascucci, and B. Hamann, "Topology-controlled volume rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 2, pp. 330–341, 2007.

[48] S. Takahashi, Y. Takeshima, and I. Fujishiro, "Topological volume skeletonization and its application to transfer function design," *Graphical Models*, vol. 66, no. 1, pp. 24 – 49, Jan. 2004.

[49] S. Takahashi, G. M. Nielson, Y. Takeshima, and I. Fujishiro, "Topological volume skeletonization using adaptive tetrahedrization," in *Proceedings of Geometric Modeling and Processing 2004*, 2004, pp. 227–236.