Topology Exploration with Hierarchical Landscapes

Dogan Demir[1] Kenes Beketayev[2,4*] Gunther H. Weber[2] Peer-Timo Bremer[3] Valerio Pascucci[1]
Bernd Hamann[4]
[1]Scientific Computing and Imaging Institute, School of Computing, University of Utah
[2]Computational Research Divisition, Lawrence Berkeley National Laboratory
[3]Center for Applied Scientific Computing, Lawrence Livermore National Laboratory
[4]Institute for Data Analysis and Visualization, Computer Science Department, University of California, Davis

Abstract

Topological landscapes have been proposed as a visual metaphor for contour trees that does not require an understanding of the theory involved in defining contour trees. The idea is to create a representative terrain with the same topological structure as a given contour tree. This representation exploits the natural human ability to interpret topography and results in an intuitive visualization of otherwise abstract information. However, topological landscapes still suffer from some of the same limitations as traditional contour tree visualization. Most notable is the fact that for complex functions landscapes can quickly become highly complex, exceeding the limits of human comprehension as well as the available computing resources.

To address this challenge, we propose a new framework for the dynamic creation and visualization of hierarchical topological landscapes. Our system provides an interactive visualization of complex functions by utilizing a hierarchical decomposition of the contour tree as well as focus+context type interactions. For three dimensional data, we link the landscape display to flexible isosurface extraction in order to correlate terrain features with their corresponding three dimensional counterparts. We demonstrate the utility and versatility of our approach on a variety of both low and high dimensional data sets.

CR Categories: I.3.M [Computer Graphics]: Miscellaneous—Scalar Field Visualization H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical User Interfaces (GUI)

Keywords: scalar field visualization, topological landscapes, graphics user interface

1 Introduction

Topological information has proven useful in a wide variety of applications ranging from volume rendering [Weber et al. 2007b] to combustion analysis [Bremer et al. 2011]. The contour tree, in particular, has been used extensively in data analysis and visualization [Carr et al. 2003; Bajaj et al. 1998; van Kreveld et al. 1997; Mizuta et al. 2004] as it encodes the nesting behavior of all contours of a scalar function. However, interpreting contour trees requires some intermediate level understanding of Morse theory and related concepts. As a result they are not well suited for the majority of users, and topological landscapes [Weber et al. 2007a] have been developed to convey the same information in a more intuitive manner. A topological landscape is a two dimensional terrain with the same level set structure as a given contour tree and harnesses the human ability to interpret topographic information.

In the original algorithm [Weber et al. 2007a] the layout of the terrain is directly coupled to the 4-8 style subdivision of the SOAR terrain rendering scheme [Lindstrom and Pascucci 2002], which imposes significant constraints. The resulting terrain requires a large number of triangles for even moderately sized contour trees and provides no ability to selectively refine particular areas. Furthermore, to match the human intuition, it uses an expensive reparameterization step to correlate the area a feature covers with an importance metric, i.e., its volume.

Subsequent work has focused on applying these concepts to high-dimensional data [Harvey and Wang 2010; Oesterling et al. 2010], and proposes an alternate means of landscape layout schemes via tree maps [Harvey and Wang 2010]. The tree map based layout is more flexible than the original technique and directly incorporates any given area assignment. However, it can create triangles with extreme aspect ratios making the resulting landscape difficult to interpret.

Instead, we propose a new dynamic layout scheme that enables us to render deep hierarchies with a large number of critical points interactively. Our approach is based on converting a contour tree into a hierarchical branch decomposition and representing each branch as a rectangular box in the landscape. Boxes are placed recursively, according to a first fit packing approach that preserves the topological correctness, avoids unnecessarily fine triangulations, and scales the boxes according to the given area constraints. Subsequently, all boxes are seamlessly triangulated using a modification of the SOAR algorithm [Lindstrom and Pascucci 2002]. Both the layout and the triangulation are performed on-the-fly, which enables dynamic changes to the landscape. We utilize this additional flexibility to provide a focus+context type interaction, as well as a multiresolution adaptation.

Our contributions in detail are:

- Introducing a layout method that dynamically places and sizes boxes eliminating the need for re-parametrization;
- Extending the SOAR framework to triangulate various sized boxes with fewer elements and without the need to constrain their placement;
- Enabling dynamic and interactive changes to the terrain to support focus+context style zooming;
- Linking topological landscapes to isosurface extraction to drive a traditional scalar field exploration; and
- Testing the new technique on a variety of real-world application data sets.

*e-mail:kbeketayev@lbl.gov

2 Related Work

2.1 Contour Trees

Contour trees capture the topological evolution of an isosurface for varying isovalue of a scalar function. Nodes correspond to critical points where the number of contours, i.e., connected components of the isosurface, changes [Carr et al. 2003].

Contour trees of even moderately-sized data sets can be quite complex. The branch decomposition of the contour tree [Pascucci et al. 2009] is an efficient data structure for encoding a multi-resolution representation of the contour tree. It decomposes the contour tree into a set of branches, each being a extremum-saddle pair. Using this data structure, it is possible to traverse the contour tree efficiently up to a desired level of detail.

Traditionally, the contour tree is visualized as a graph [Heine et al. 2011; Pascucci et al. 2009]. This graph-based visualization easily becomes cluttered and often is hard to understand for novices. Mizuta et al. [2006] introduced the contour nest, which focuses on the nesting properties of isosurfaces. The topological landscapes [Weber et al. 2007a; Harvey and Wang 2010] metaphor is another intuitive contour tree representation that has been applied to high-dimensional data [Harvey and Wang 2010; Oesterling et al. 2010].

Flexible isosurfaces [Carr and Snoeyink 2003] utilize the correspondence between connected isosurface components and contour tree arcs to increase the expressiveness of isosurface visualizations. Utilizing this method, it is possible color contours distinctively, show only a subset of contours for a given isovalue, or use different isovalues for individual contour components. Weber et al. [2007b] apply similar concepts to transfer function design in volume rendering.

2.2 Terrain Rendering

Edge bisection [Lindstrom et al. 1996] is a widely adopted approach for generating landscapes from hierarchical structures. Previous work has focused on generating a valid, crack-free triangulation, while adjusting resolution dynamically for performance. ROAM [Duchaineau et al. 1997] improved this method by using a top-down instead of a bottom-up approach. SOAR [Lindstrom and Pascucci 2002] builds on a variety of improvements [Röttger et al. ; Blow 2000] to the ROAM algorithm making it possible to use a dynamic error threshold based on virtually any type of error metric, while generating a continuos, crack-free terrain.

2.3 Box layout

The problem of finding the most efficient box layout within a bounded region arises in several application domains. For our application, we need to layout child boxes of varying sized corresponding to child branches within a rectangular region of a parent branch. In the original topological landscapes method [Weber et al. 2007a], all boxes have the same size (before re-parametrization), arranged in a spiral around the center of parent. Sorting child branches by their saddle value ensures preservation of the topology of the underlying dataset. Harvey and Wang [2010] use treemaps to generate the layout from the hierarchy. However in their paper they explore only layouts with hierarchy nodes with two children (since they use the contour tree and not the branch decomposition). We generalize this layout by considering a basic bin packing algorithm, which is a well-studied packing solution approximation [Coffman et al. 1997].

3 Contour Tree Computation

We construct dynamical landscapes from the branch decomposition of a contour tree. We compute the contour tree using the algorithm by Carr et al. [2003] and subsequently convert it into a branch decomposition [Pascucci et al. 2009].

When computing join and split tree (as a part of the contour tree computation), we need to enumerate neighbors for each point of the input data. For two or three dimensional data on a rectilinear grid, we use minimal tetrahedral subdivision [Carr et al. 2001] and an alternating 4-/8 or 6-/18-neighborhood to triangulate the domain.

Higher dimensional data is usually not specified on a regular grid, since with increasing dimensionality, the number of samples would quickly exceed practical bounds. Instead, it is given as an arbitrary point cloud. In that case, edges between vertices are defined by computing a set of neighbors for each vertex of a given point cloud [Harvey and Wang 2010; Oesterling et al. 2010]. In our implementation, we use the k-nearest-neighbors for high-dimensional point could data. We choose the parameter k by careful examination of the values between $2 * d$ and $3^d - 1$, where d is a number of dimensions. Correa and Lindstrom [2011] provide a more in-depth discussion of alternatives.

We approximate the volume of each branch by counting all regular points corresponding to it. During contour tree calculation, we keep track of all regular points getting merged into a particular arc, and we transfer this information to the branch decomposition. We also compute a volume of the leaf part of each branch by counting the regular points of the branch that have function values between those of the extremum and the closest saddle. For example, if an extremum is a maximum, we take a saddle with the highest function value, see Figure 1.

4 Dynamical Landscape Creation

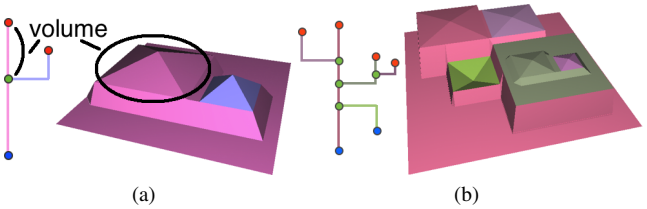
Given a contour tree, our approach dynamically creates the corresponding landscape in a hierarchical fashion proceeding in three basic steps: First, we convert the tree into a hierarchical branch decomposition, assign an importance metric to each branch, and create a corresponding hierarchy of boxes. Second, we create a topology preserving layout of the box hierarchy using a modified first-fit box packing scheme. Finally, we use an augmented SOAR algorithm to create a crack-free triangulation for interactive rendering.

4.1 Nested Box Hierarchy

Given a fully augmented contour tree we use the algorithm of Pascucci et al. [2009] to create a hierarchical branch decomposition. As shown in Figure 1, this decomposition results in a hierarchical set of branches in which each branch (except the root) attaches to its parent at its saddle value. Each branch represents a single mountain or valley with its children creating smaller nested terrain features. In the landscape each branch will be represented as a box with child boxes nested within. Thus, the hierarchy determines the global structure of the terrain. As in the original algorithm we typically use persistence as metric to create the decomposition as it corresponds to an L_∞-optimal simplification. However, similar to Harvey and Wang [2010] we could use other metrics or branch orderings to construct landscapes with different nestings.

Given a decomposition we compute four parameters for each branch: its peak/valley value; its saddle value; its volume above/below the highest/lowest saddle; and its volume below/above the highest/lowest value. The first two parameters determine the

generic shape used for each box. More specifically, a branch without children is represented as a square box with a single interior vertex where the edge of the box is drawn at the height of the saddle value and the center vertex at the peak/valley height. Similarly, a branch containing children will draw its edge at the saddle value but its peak as a smaller box on the interior, see Figure 1. The exception to this rule is the root branch that typically represents its maximum as a peak and the global minimum by the landscape edge.



(a) (b)

Figure 1: *Two brach decompositions with the corresponding landscapes. (a) The two base elements of our landscapes are a leaf box shown in blue and a box with children shown in pink. In the second case the area of the box vs. the area of the peak is determined by the volume of the corresponding branch below and above the saddle. (b) A slightly more complex example of a branch decomposition with multiple children.*

The second two parameters describing the corresponding volume in the original domain are used to determine the area a given box should cover. Modulating the areas is important as users implicitly connect the overall size of a terrain feature with its importance. We use the volume as the natural measure to correspond to the area but any other property such as hypervolume or surface area can be used. Given a branch b with a total volume v, its target area V_b is:

$$V_b = v^{d/2} + \sum_{c \in C} V_c,$$

where d is the dimension of the domain and C is the set of children of b. The factor $v^{d/2}$ compensates for the perceptual difference in judging relative areas compared to relative (hyper-) volumes. For a branch with children, the target area of its peak/valley box is determined by the third parameter: the (scaled) volume above/below the highest/lowest saddle. In this manner we ensure that a parent box is always large enough to contain all children and that the relative sizes among branches are preserved.

To simplify the subsequent layout and rendering, we subdivide the parent box into a power of two grid and restrict child boxes to conform to this grid, see Figure 2. While this quantization introduces a certain error with respect to the ideal target area it significantly reduces the complexity of the box placement and allows for a simple triangulation scheme. The area error can be reduced by increasing the grid resolution, see Figure 2, at the cost of more triangles to display.

4.2 Box Packing

Given the box hierarchy with the corresponding target areas the algorithm proceeds hierarchically starting with an arbitrary sized box for the root branch. Given a box, we first determine the grid resolution and box sizes necessary to approximate the area of all children up to a user defined threshold. To this end, we compute the edge length of a child box with respect to its parents box as:

$$L_{child} = \left\lceil \sqrt{G_{parent} \frac{V_{child}}{V_{parent}}} \right\rceil,$$

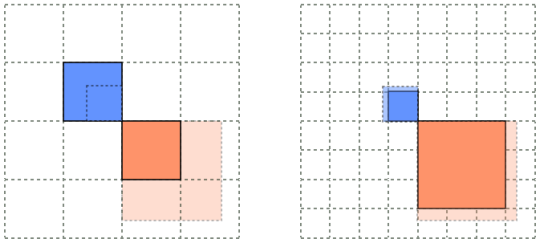


Figure 2: *The children of a box are placed on a grid within the parent area. The resolution of the grid determines how closely the final area of the children approximate their target area values. In this figure we show the parent grid in the background. As the resolution of grid is increased by the algorithm, box sizes (shown in solid outline) approximate better their target area values (shown in dotted outline).*

where G_{parent} denotes the total number of cells of the parent. We then compute the corresponding edge length with respect to the entire landscape as:

$$P_{child} = L_{child} \sqrt{\frac{P^2_{parent}}{G_{parent}}}.$$

Assuming the entire terrain is drawn in the unit box, then P^2_{child} should be equal to the volume ratio V_{child}/V_{root}. Hence, we compute the area error ϵ_{child} as the ration between the desired area ratios and the current approximate ratio:

$$\epsilon_{child} = \frac{V_{child}}{V_{root}} * \frac{1}{P^2_{child}}.$$

If for any child the error is above a user defined threshold we subdivide the parent grid and recompute the error. Since we are interested in creating a small triangle count we typically start with a parent grid of size four which we then refine if necessary. If the additional computation cost becomes a problem, we can start at a higher resolution grid. The error is computed relative to the root box in order to avoid accumulating errors for deeper levels of the hierarchy.

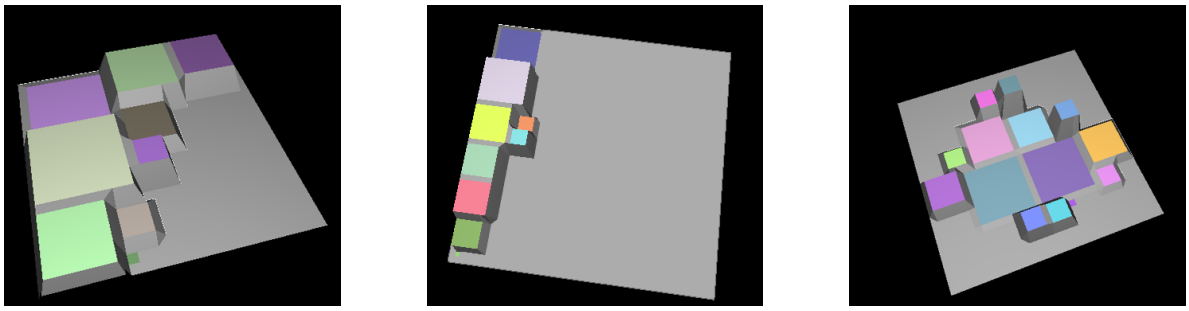
Finally, we find that accepting an increasing amount of error for branches deeper in the hierarchy greatly reduces the overall triangle count without significantly impacting the overall appearance. Hence, we use two error thresholds, minimal T_{min} and maximal T_{max}, between which we interpolate based on the current hierarchy level. More specifically, when rendering level l_{cur} of a hierarchy of maximal depth l_{max} the error threshold is computed as:

$$T_{cur} = T_{min} + \frac{l_{cur}}{l_{max}} (T_{max} - T_{min})$$

We use $T_{min} = 1.5, T_{max} = 200$.

Once the final integer valued sizes are computed, we recursively create a layout of boxes using a modified first fit packing algorithm [Scott]. By precomputing appropriate edge lengths, the layout problem reduces to placing integer sized boxes within a given larger box. However, a straightforward first-fit packing algorithm produces an unappealing layout heavily biased towards one of the box corners, see Figure (a).

Moreover, even though by construction the sum of the target areas of all children is lower than the area of the parent, the actual boxes may not all fit. This problem can be alleviated by using an infinite bin size with a subsequent rescaling as shown in Figure 3(b).



(a) (b) (c)

Figure 3: ***Packing.*** *(a) Applying straightforward first-fit bin packing yielding a landscape in which boxes originate from top-left corner of the parent. This method may not accommodate all the children if the resolution is too small or there are more children than the number of available grid cells. (b) Results of the same algorithm where the bin dimensions are infinite. Box sizes are small and they do not go beyond the parent boundaries. (c) Final algorithm in which we use four bins pivoted around the center of the parent with infinite dimensions. Boxes are shrunk proportionally until they all fit within the parent.*

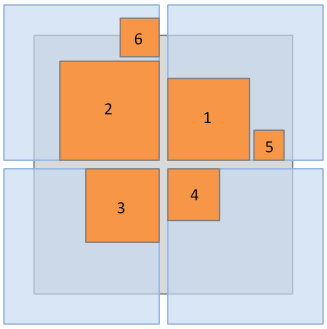


Figure 4: *Boxes are packed first-fit using four bins with infinite size pivoted around the center of parent box (shown in shaded in the background). Each box is assigned to one bin in round-robin order, arranging boxes in a spiral around the center guaranteeing topological correctness. If necessary, children are uniformly shrunk to fit within the parent.*

However, the layout remains sub-optimal. More importantly, such a layout does not preserve the topology of the terrain, one of the primary goals of the topological landscape. In particular when using the terrain to explore isocontours of a three dimensional data set, maintaining the correct nesting behavior is important. Hence, we extend the original layout algorithm of Weber et al. [Weber et al. 2007a] to fit uneven sized boxes. The original scheme places children in a spiraling layout around the center of the parent box, which guarantees the topological correctness of the terrain. We emulate this behavior by creating our infinite sized bins pivoted around the center of the parent box. We then sort children by saddle value and place them in round robin fashion into the four bins, see Figure 4. In rare cases, the children will spill outside their parent box, in which case we rescale all children to fit. The final layout is created on-the-fly and produces a well centered and visually pleasing layout that preserves the topology and provides a good approximation for the areas, shown in Figure 3(c).

4.3 Rendering

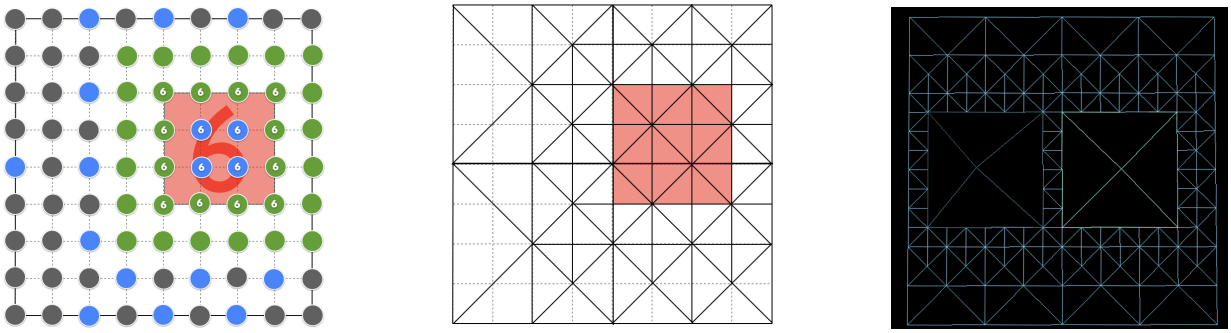
The goal of the rendering algorithm is to create a crack-free display of the terrain with a triangle count as low as possible while avoiding extreme aspect ratios. We use a modified SOAR algorithm [Lindstrom and Pascucci 2002] to create a conforming triangulation for each box depending only on the existence and placement of its children. Starting from the grid used for the layout we flag each grid point covered by a child node (see Figure 5(a)). We then propagate these flags upward in the SOAR hierarchy to determine the smallest conforming SOAR mesh that would render the children at grid resolution (see Figure 5(b)). The original algorithm proceeds by drawing this mesh using a single triangle strip. However, in our case the children will be responsible for drawing their own area and thus we break the triangle strip whenever we enter a child box and discard the corresponding triangles. Note that the children may not draw a mesh at the same resolution as their parent. However, as the grid values are constant all along the edge of the children the resulting T-junctions do not cause problems. An example of a resulting mesh is shown in Figure 5(c).

5 Focus+Context Exploration

As discussed above, contour trees of common data sets often contain thousands of branches and displaying the corresponding terrain will quickly exceed the capabilities of current graphics hardware. Even if such a terrain would be rendered, smaller branches in the lower levels would become virtually invisible or might create very high yet tiny spikes. To address this problem, we once again exploit the hierarchical nature of the branch decomposition and only render branches up to either a given persistence or a given hierarchy level. We also allow the user to zoom into any given child branch to provide a focus+context type interaction to easily explore the entire branch decomposition, see Figure 6. We note that both the layout and the triangulation are created on-the-fly.

5.1 Linking Landscapes to Flexible Isosurfaces

Finally, to correlate the dynamical landscape to 3D data set, we utilize flexible isosurfaces [Carr and Snoeyink 2003]. Flexible isosurfaces allow us to match the colors for contours in the 3D visualization to the regions in the terrain corresponding to them. We have implemented a version of flexible isosurfaces in VisIt. However, unlike the original version, we do not employ a continuation method. Instead, we store the branch IDs for each vertex in the data set. When extracting an isosurface, we perform a transfer function lookup to identify the branch corresponding to each isosurface triangle, and associate this information with the triangle. This method allows us to color triangles by contour ID, and filter triangles, when hiding the contour.



(a) (b) (c)

Figure 5: ***Triangulating the grid.*** *(a) For each location in the grid we store the ID of the box that overlaps with it and the boolean flags for SOAR refinement. In the figure IDs are shown as numbers within circles. In this figure, the box has ID 6 therefore at all grid locations that overlap with the box, the IDs are set to 6. Grid locations with empty circles contain a non-occupied value. Green circles show the initial boolean flags, blue circles show the boolean flags turned on by upwards propagation within SOAR hierarchy. Gray circles show flags that are off. (b) This scheme generates an artifact-free triangulation. We use triangle lists instead of two triangle strips as proposed in original SOAR [Lindstrom and Pascucci 2002] to discard triangles that completely overlap with child boxes. In this figure, triangles over the orange child box are discarded. (c) Screenshot from a final rendering of one of our test data sets with one parent box with a peak (on the left) and a child (on the right). We take advantage of the fact that all boxes have the same value on the edges, therefore in triangulation space, continuity between the parent and the child box is not required. This allows boxes to be triangulated and rendered completely independently.*

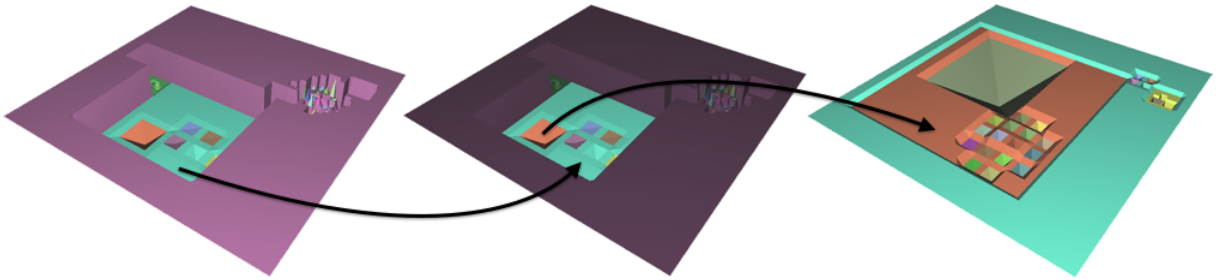


Figure 6: Focus+context exploration. *When marking the cyan box on the left for further exploration, we darken rest of the landscape and enlarge the selected box. Boxes are interactively repacked to a lot more screen space to the focused box. By construction, the children of the highlighted box maintain their relative sizes.*

6 Applications and Results

In this section, we demonstrate the utility of our technique on simple (hydrogen atom, nucleon) and complex, real-world (3D chemical, 5D performance optimization, turbulent combustion) data sets.

Hydrogen data set. Figure 7 shows the landscape for the hydrogen atom data set, which is the probability distribution of an electron in an H_2 atom residing in a strong magnetic field. The figure clearly shows all the major components we expect to see: two lobe components (dark gray and yellow) and a center component (gray) located inside a toroidal ring component (brown). Choosing different isovalues and examining corresponding contours, correlated to the landscape via colors, provides general idea about the data set. Instantly we encounter a problem with occlusion of the middle component by the toroidal ring. To resolve this, we use a focus feature that allows us to hide a selected landscape element and corresponding component, in this case the toroidal ring, to clarify the view, see Figure 7. This feature also can be used to focus on a component, while hiding the rest. These constitute powerful interactive exploration capabilities of our framework.

Nucleon data set. Figure 8 shows a similar analysis for the nucleon data set obtained by simulating a two-body distribution probability of a nucleon in the atomic nucleus "16O" when a second nucleon is known to be positioned at distance of 2 Fermi. We can see two blob-like inner contours and one outer contour, all having large volumes. To reveal two blobs inside the outer contour, we hide this outer component (right column). Sweeping isovalues shows that first one, and then the other blob merge with the outer component. Subsequently there are two new components that encapsulate all maxima of the data set. Comparing the volumes of all components found so far brings a rather non-obvious result—the maximum inside one of the new components has comparable value with former three minima components, which is not obvious from isosurface visualization.

Analysis of free energy function in porous materials. To further demonstrate the utility of our framework, we consider a more complicated chemical data set and show the ability of our framework to aid in its analysis. It is based on the analysis of zeolites, a well recognized class of crystalline porous materials that are commonly used as membranes and adsorbents. We are interested in the energy function of a trapped CH_4 molecule as function of location, as it shows absorption sites (energy minima) and possible locations of the guest molecule. Prior work focused on energy states and transitions between them. In contrast, we use topological landscapes and flexible isosurface to approximate spatial locations of diffusion pathways of the CH_4 guest molecule inside the periodic LTA zeolite box. The energy landscape in Figure 9 shows the absorption sites and energy barriers between them (saddles) that correspond to

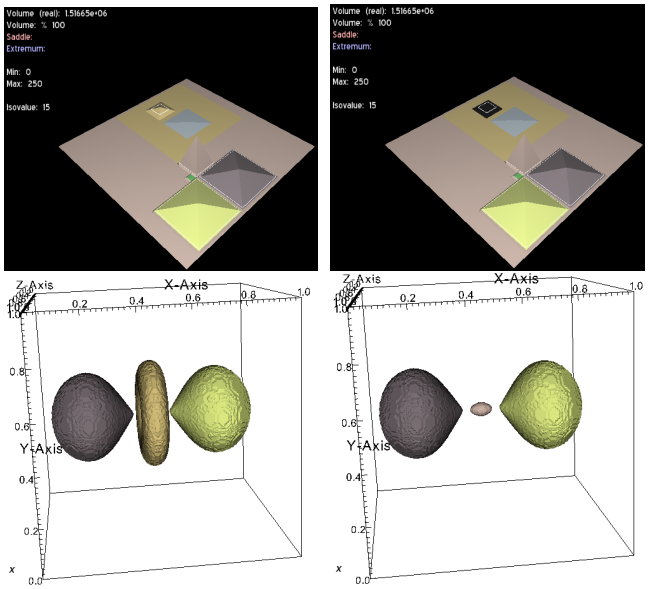


Figure 7: ***Hydrogen atom data set.*** *(Left) View of the landscape together with linked isosurfaces below. (Right) Hiding the green brown peak that corresponds to the ring contour shows the center contour corresponding to the main peak.*

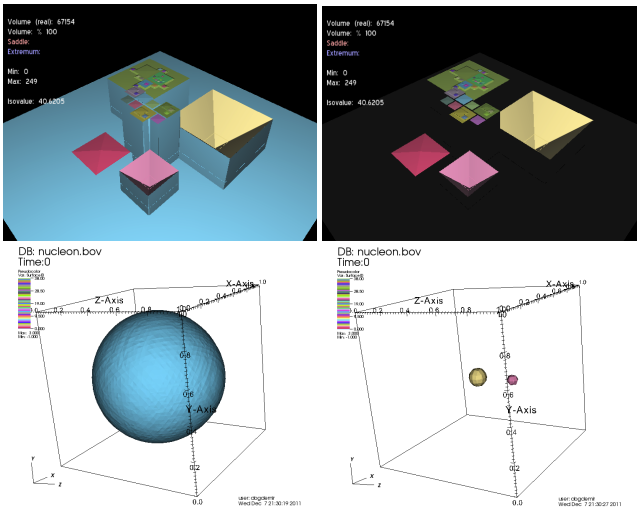


Figure 8: ***Nucleon data set.*** *(Left) General view of the landscape together with linked isosurface. We see only outer blue component that occludes two inner blobs. (Right) Hiding outer blue component resolves the occlusion.*

void channels and cages. By linking to a flexible isosurface view, we show spatial locations for the minima (second image). Using the landscape view, we identify the saddle value for which minima within a cage merge. The resulting isosurface (third image) shows locations, corresponding to minimal energy configurations between the absorption sites, i.e., the location of "channels" between absorption sites. The saddle to the global maximum coincides with the boundary of the cages. The corresponding isosurface (right image) marks the region that the trapped molecule cannot escape, unless the material is subjected to very high temperatures.

Auto-tuning data set. Our high-dimensional data example is based on a study of auto-tuning strategies for HPC systems [Williams et al. 2011] with the goal of designing an auto-tuner for large complicated algorithm runs. This study uses a lattice Boltzmann magnetohydrodynamics algorithm and its performance evaluation on different HPC systems to explore the available parameters to tune. We analyze the results generated on the "Hopper" system at National Energy Research Scientific Computing center (NERSC).

Each point in data set corresponds to a fixed configuration of available optimization options. The space of input auto-tuning parameters consists of the following five variables: number of processes {8,16,24,48,96,192}, unrolling depth {16,...,512}, virtual vector length {1,...,16}, data-level parallelism {1,...,16}, pre-fetch option {+VL, +128}. The output function is measured as an average performance of the algorithm for each parameter combination, given in floating point operations per second per core.

Initial observation of the landscape in Figure 10, generated for the data set, finds several configurations that are locally optimal, which might suggest some configurations that potentially can be fixed and thrown out of the analysis, reducing significantly run-time of the auto-tuner. We were able to narrow down the configuration (48,512,16,8), which encompasses two regions—an outer big one and big cluster of the solutions inside the inner higlighted white box. This box occupies a large volume (97%), hence we are able to further focus into it without missing much information.

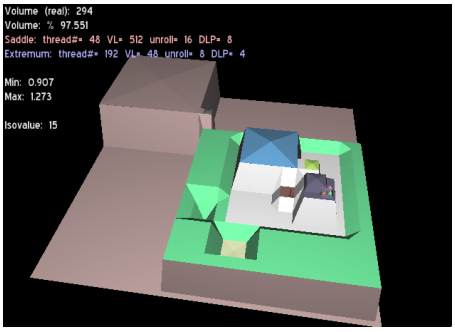


Figure 10: ***Auto-tuning data set.*** *General view of the landscape, with highlighted box that corresponds to the configuration that encompasses big cluster of locally optimal solutions.*

Turbulent combustion data set. Finally, Figure 11 shows one time step of the premixed turbulent combustion simulation analyzed in [Bremer et al. 2011]. The landscape shows the contour tree of fuel consumption and areas of high fuel consumption are considered burning cells. The terrain clearly delineates clusters of high maxima indicating groups of cells. Moreover, the shape of the individual hills shows some interesting characteristics not easily accessible with other techniques. While some clusters show a "crown" of maxima with similar function values, some show rather flat plateaus typically at medium levels of fuel consumption. These are likely areas of slowly dying burning cells in which all original peaks have been smoothed by diffusion type processes. Additionally, some clusters show single high spikes with low area indicating cells in which a single or a small number of maxima that are in the process of splitting and forming a new cell. As shown by the multiple zoom levels the data is highly complex and similar observations would be difficult to obtain using a graph version of the contour tree.

7 Conclusions and Future Work

We have presented dynamic, hierarchical landscapes that make it possible to explore large scale data sets using a topological landscape metaphor. The increased flexibility supports an improved

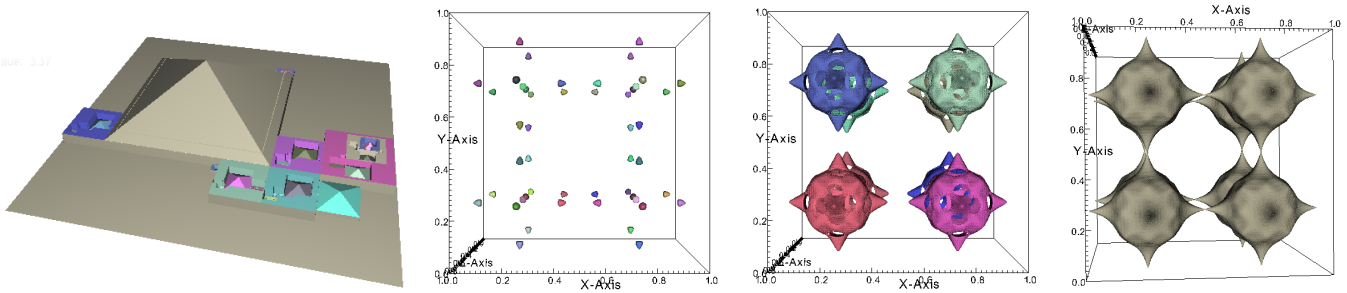


Figure 9: ***Chemistry data set.*** *The left image shows the landscape view revealing the cages (associated with minima) in the data set. The next image shows isosurfaces for the minima in the data set, i.e., absorption sites for the trapped moleculse. The third image shows contours that separate the minima within a cage. The right image shows the isosurface separating the cages that the trapped molecule cannot leave.*

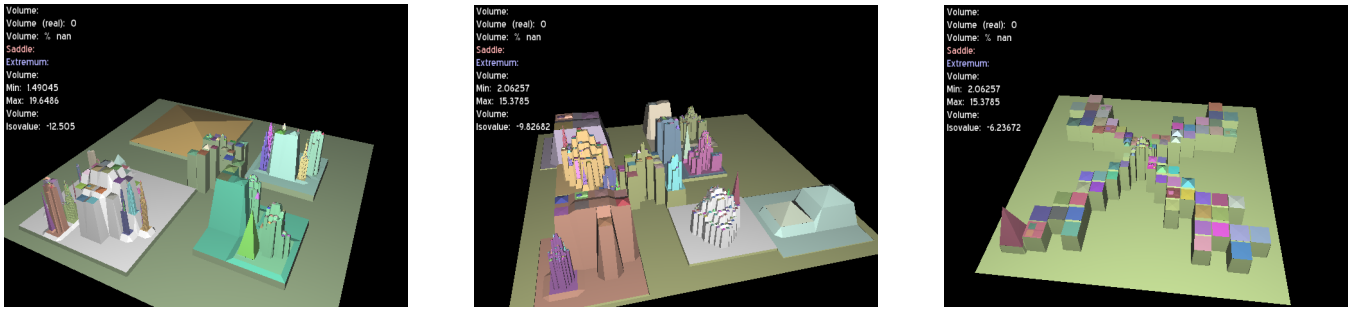


Figure 11: *Three levels of the topological landscape of fuel consumption in a turbulent combustion simulation. (Left) The entire domain showing four intensely burning cells and one low intensity region, likely a dying cluster. (Middle) Zooming into the region highlighted in white reveals a number of well separated sub-cells again with some lower intensity flat regions. Note the small spikes in some of the cells indicating new cells forming as maxima split from their surrounding cells. (Right) A further zoom level reveals a highly nested group of maxima at very similar function value representing the "crown" of the burning cell highlighted in the middle image.*

layout and coupled with focus+context animation and linking to flexible isosurface, this layout makes it easier to explore complicated data sets. In the future, we plan to utilize this framework to visualize time-dependent data, which poses the challenge of changing the layout smoothly over time while maintaining correct topological properties. We also plan to explore using our technique for other terrain rendering applications.

Acknowledgements

This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. DOE under Contract Nos. DE-AC02-05CH11231 (Berkeley Lab.), DE-AC52-07NA27344 (Livermore Lab.) and DE-FC02-06ER25781 (The Univ. of Utah) and the use of resources of the NERSC.

References

BAJAJ, C. L., PASCUCCI, V., AND SCHIKORE, D. R. 1998. Visualization of scalar topology for structural enhancement. In *Proc. IEEE Visualization'98*, 51–58.

BLOW, J. 2000. Terrain rendering at higher levels of detail. *In Proc. 2000 Game Developers Conference*.

BREMER, P.-T., WEBER, G. H., TIERNY, J., PASCUCCI, V., DAY, M. S., AND BELL, J. B. 2011. Interactive exploration and analysis of large scale turbulent combustion using topology-based data segmentation. *IEEE Trans. Vis. Comput. Graph. 17*, 9, 1307–1324.

CARR, H., AND SNOEYINK, J. 2003. Path seeds and flexible isosurfaces using topology for exploratory visualization. In *Proc. Data Vis'03*, 49–58.

CARR, H., MÖLLER, T., AND SNOEYINK, J. 2001. Simplicial subdivisions and sampling artifacts. In *Proc. IEEE Vis'01*, 99–106.

CARR, H., SNOEYINK, J., AND AXEN, U. 2003. Computing contour trees in all dimensions. *Comput. Geom.—Theory and Apps 24*, 2, 75–94.

COFFMAN, E. G., GAREY, M. R., AND JOHNSON, D. S. 1997. Approximation algorithms for bin packing: A survey. In *Approx. Algs*. PWS Publishing Company.

CORREA, C. D., AND LINDSTROM, P. 2011. Towards robust topology of sparsely sampled data. *IEEE Trans. Vis. Comput. Graph. 17*, 12 (dec).

DUCHAINEAU, M. A., WOLINSKY, M., SIGETI, D. E., MILLER, M. C., ALDRICH, C., AND MINEEV-WEINSTEIN, M. B. 1997. ROAMing terrain: Real-time optimally adapting meshes. In *Proc. IEEE Vis'97*, 81–88.

HARVEY, W., AND WANG, Y. 2010. Topological landscape ensembles for visualization of scalar-valued functions. *Computer Graphics Forum 29*, 3, 993—1002.

HEINE, C., SCHNEIDER, D., CARR, H., AND SCHEUERMANN, G. 2011. Drawing contour trees in the plane. *IEEE Trans. Vis. Comput. Graph. 17*, 11, 1599–1611.

LINDSTROM, P., AND PASCUCCI, V. 2002. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Trans. Vis. Comput. Graph 8*, 3, 239–254.

LINDSTROM, P., KOLLER, D., RIBARSKY, W., HODGES, L., FAUST, N., AND TURNER, G. 1996. Real-time continuous level of detail rendering of height fields. *Proc. of SIGGRAPH'96*, 109–118.

MIZUTA, S., SUWA, Y., ONO, T., AND MATSUDA, T. 2004. Description of the topological structure of digital images by contour tree and its application. Tech. rep., Institute of Electronics, Information and Communication Engineers.

MIZUTA, S., ONO, T., AND MATSUDA, T. 2006. Contour nest: A two-dimensional representation for three-dimensional isosurfaces. In *Proc. Volume Graph.*, 67–70.

OESTERLING, P., HEINE, C., JÄNICKE, H., AND SCHEUERMANN, G. 2010. Visual analysis of high dimensional point clouds using topological landscapes. In *Proc. IEEE Pacific Vis'10*, 113–120.

PASCUCCI, V., COLE-MCLAUGHLIN, K., AND SCORZELLI, G. 2009. The toporrery: Computation and presentation of multi-resolution topology. In *Math. Found. of Sci. Vis., Comput. Graph., and Massive Data Explor.*, Springer-Verlag, 19–40.

RÖTTGER, S., HEIDRICH, W., SLUSALLEK, P., AND SEIDEL, H.-P. Real-time generation of continuous levels of detail for height fields. In *Proc. WSCG'98 Conference*, V. Skala, Ed.

SCOTT, J. Packing lightmaps. `www.blackpawn.com/texts/lightmaps`.

VAN KREVELD, M. J., VAN OOSTRUM, R., BAJAJ, C. L., PASCUCCI, V., AND SCHIKORE, D. 1997. Contour trees and small seed sets for isosurface traversal. In *Symposium on Computational Geometry*, 212–220.

WEBER, G. H., BREMER, P.-T., AND PASCUCCI, V. 2007. Topological landscapes: A terrain metaphor for scientific data. *IEEE Trans. Vis. Comput. Graph. 13*, 6, 1416–1423.

WEBER, G. H., DILLARD, S. E., CARR, H., PASCUCCI, V., AND HAMANN, B. 2007. Topology-controlled volume rendering. *IEEE Trans. Vis. Comput. Graph. 13*, 2, 330–341.

WILLIAMS, S., OLIKER, L., CARTER, J., AND SHALF, J. 2011. Extracting ultra-scale lattice Boltzmann performance via hierarchical and distributed auto-tuning. In *Proc. Supercomputing Conference*.