Topological Galleries: A High Level User Interface for Topology Controlled Volume Rendering

Brian MacCarthy, Hamish Carr and Gunther H. Weber

Abstract Existing topological interfaces to volume rendering are limited by their reliance on sophisticated knowledge of topology by the user. We extend previous work by describing *topological galleries*, an interface for novice users that is based on the *design galleries* approach. We report three contributions: an interface based on *hierarchical thumbnail galleries* to display the containment relationships between topologically identifiable features, the use of the *pruning hierarchy* instead of branch decomposition for contour tree simplification, and *drag-and-drop* transfer function assignment for individual components. Initial results suggest that this approach suffers from limitations due to rapid drop-off of feature size in the pruning hierarchy. We explore these limitations by providing statistics of feature size as function of depth in the pruning hierarchy of the contour tree.

1 Introduction

The overall goal of scientific visualisation is to provide useful insight into existing data. As visualisation techniques have become more complex, so have the interfaces for controlling them. In cases where the interface has been simplified, it has often been at the cost of the functionality of the program. Thus, while expert users are capable of using state of the art technology, novice users who would otherwise have uses for this technology are restricted by unintuitive interfaces.

Topology-based volume rendering, while powerful, is difficult to apply successfully. This difficulty is due to the fact that the interface used to design transfer func-

Brian MacCarthy
University College Dublin, Belfield, Dublin 4, Ireland, e-mail: brian.maccarthy@ucd.ie

Hamish Carr
University of Leeds, Woodhouse Lane, Leeds LS2 9JT, England e-mail: h.carr@leeds.ac.uk

Gunther H. Weber
Lawrence Berkeley Nat. Lab, 1 Cyclotron Road, Berkeley, CA 94720 e-mail: GHWeber@lbl.gov

tions requires detailed topological knowledge and experience as well as experience in transfer function design. Expanding the utility of this approach therefore requires development of interfaces where topological skill is not required.

This paper introduces an alternative interface for topological visualisation, with the intent of increasing the options for user-friendly interaction with topological abstractions. This goal is achieved by combining existing state-of-the-art work on topological visualisation with the design galleries approach for exploring feature spaces. The resulting approach organises topological features in a file system-style visual hierarchy for display purposes. Within this hierarchy, thumbnail renderings represent individual features, while transfer functions are represented iconically and can be applied to individual features via drag-and-drop.

While this interface is technically successful, it is still limited by scaling issues, as the reduction of individual features to thumbnails makes it difficult for a user to interpret individual features, and some analysis and discussion is therefore provided on the situations in which this style of interface may be successful.

2 Previous Work

Our interface is based on a simple observation—novice users can be expected to understand that one object may be contained inside another. Thus, by representing the containment relationships implicit in topological analysis, it is possible to construct an interface that is more approachable for novice users than the existing displays of the contour tree itself.

This approach in turn requires indicating the topological features visually to the user—a requirement which leads in the direction of previous work on design galleries, where the space of possible visualisations is shown to the user. Thus, underpinning the contributions in this paper are three distinct areas of previous work: work on design galleries, on direct volume rendering, and on topological visualisation using the contour tree. Of these topics, the one least dependent on the others is volume rendering, and we accordingly discuss it first.

2.1 Volume Rendering

Direct volume rendering, one of the standard techniques for visualising scalar datasets, is based on modelling the interaction of light passing through a translucent medium of varying optical properties [18, 13, 15]. This is achieved in practice by defining a *transfer function*, which defines the optical properties at a given point in space, typically taking one or more data values as input and assigning optical properties accordingly. In general, the inputs may include isovalue [13], gradient [13], moments of inertia [20], boundary information [7], curvature [11], or feature scale [5]. The output is more straightforward, and consists of opacity and

colour, of which the latter must be chosen carefully [21] to ensure that different objects are distinguishable in the rendering. All of these properties, however, operate globally, and the desire to apply different transfer functions to different regions led to the combination of direct volume rendering with topological analysis based on the contour tree.

2.2 Contour Tree

For a given data set, the *contour tree* tracks contours, i.e., the individual connected components of an isosurface, and traces their evolution as they are created, destroyed or split or merge. Since this information makes it possible to segment data-dependent regions and visualise them with different properties, the use of the contour tree for visualisation has been of increasing importance in the last decade. However, it's use dates back to 1963, when Boyell & Ruston [2] used it as a data structure representing the nesting structure of contours on a topographic map—a characterisation that we rely upon in this work.

While the contour tree served initially as an abstract data visualisation method [1], subsequent work [3] also showed its utility for constructing a user interface that supports interaction with individual contours of an isosurface. Further work [4, 17] showed that the data needed to be simplified topologically in practice in order to be useful for visualisation. In this work, the contour tree is recursively simplified by identifying unimportant branches and removing them. Where this removal leaves a degree two node in the tree, two edges can then be collapsed to build up longer and longer paths through the tree that represent ever more important features—a process referred to as *branch decomposition* [17]. As we will see below, however, the branch decomposition is not necessarily ideal for visualisation, and we will discuss an alternate breakdown—the *pruning hierarchy*.

An alternate approach to visualisation is to combine direct volume rendering with contour tree analysis [8]. This combination was initially achieved by computing the nesting depth of a given contour—i.e., the graph distance in the contour tree from a defined exterior contour—and adding it as extra parameter to transfer function design [19]. However, while this method did assign distinct transfer functions to different regions, any two regions with the same isovalue and nesting depth still shared the same transfer function. This approach was subsequently extended [23] to apply arbitrary transfer functions to individual branches in the branch decomposition. However, assigning transfer functions to branches was performed manually and required the user to be deeply familiar with the contour tree. Zhou and Takatsuka [24] attempted to resolve the user interface problem by assigning transfer functions automatically, again based on nesting depth, but choosing different colours for different branches based on existing work on harmonic colormaps [21]. While this interface was substantially automated, it still required the user to understand the topology in order to choose a suitable rule for opacity distribution throughout the tree.

In summary, then, while topological tools have been established as significant visualisation techniques, their use requires either training in topology, fully automatic methods, or user interfaces that do not require topological knowledge on the part of the end user. This paper seeks to develop one such user interface.

2.3 Design Galleries

If we wish to use topology in a user interface, but do not wish to require the user to be familiar with topological analysis, we must develop an interface based on a metaphor with which they are familiar. One way to approach this is to show all the features using an interface, such as *design galleries* [14], in which a user can choose from thumbnails of potential visualisations. By doing so, the interface gives the user advance information as to the probable result of parameter choices.

In the case of topological analysis, several interfaces have used variations on the idea of visual result prediction. The Contour Spectrum [1] showed the contour tree to give the user information as to the number of connected components in each contour. The Safari Interface [10] plotted number of connected components for different time steps and isovalues and provided pop-up renderings of isosurfaces as the mouse hovered over the plot. More recently, topological landscapes [22, 9, 16] used a terrain as proxy representation for the contour tree. Related work on topological volume rendering[23] displayed the transfer function directly on the contour tree. Although these topological interfaces exploit the ability to predict the results of a visualisation, none of them shows the result directly. We therefore discuss the conceptual basis for the Topological Galleries interface next.

3 User Interface

As stated earlier, volume rendering interfaces traditionally have been either very simple to use, but provide only basic functionality, or have been very powerful, but too complex for a non expert user to take advantage of the added functionality. The overall goal of this project was to create an interface that contained all the functionality of these advanced interfaces, but present it so that a user with no concept of topology would be able to understand.

In basing the interface on design galleries, we must first choose how we will select our thumbnails, then decide how they are to be laid out. Detailed choices must then be made about the visual representation and the behaviour of the interface. We have identified three major choices: the representation of the topology, the representation of the transfer functions available, and visual feedback on the transfer function constructed to date.

Topological Representation: We start by observing that the algorithm for simplifying the contour tree naturally induces a hierarchical decomposition of the tree.

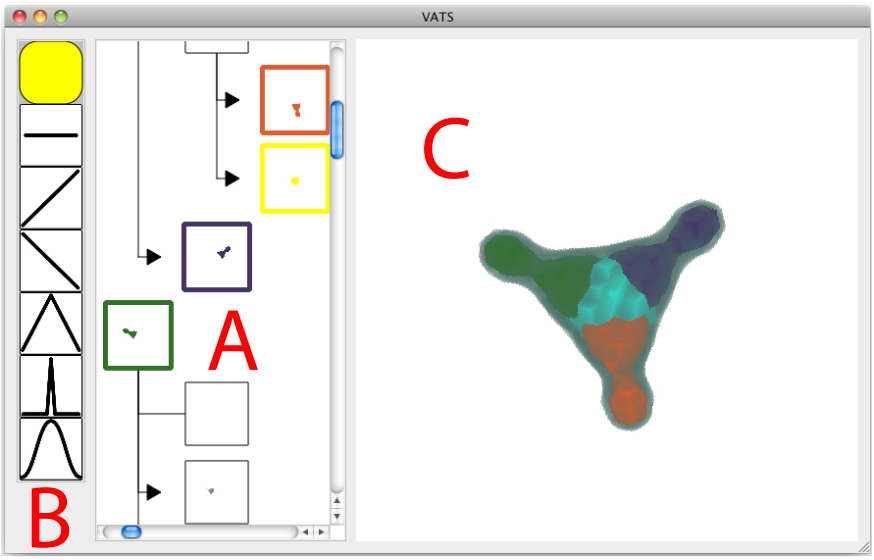

Fig. 1 The Topological Galleries Interface. At the left, a simple palette (B) for choosing colour and type of transfer function. In the middle (A), a hierarchical set of thumbnails showing the pruning hierarchy and the features currently assigned non-default transfer functions. On the right (C) the current rendering. Note that all thumbnails are linked to render from the same viewpoint as the current rendering.

Since this hierarchy corresponds to merging successively larger features of the underlying data, it is a natural choice to display the features according to this hierarchy, as shown in panel (A) of Figure 1. We will discuss in Sec. 4 why we chose pruning hierarchy rather than branch decomposition: here, we merely point out that either is possible.

Since the contour tree usually encapsulates the visual nesting hierarchy of features, previous interfaces [8, 19, 24] have already exploited this for transfer function design. However, although this hierarchy was used internally, it was not part of the user interface, except in the form of the nesting depth (an integer). In our interface, we show this hierarchy explicitly, thus revealing to the user the nesting relationships between objects at different levels of detail.

Moreover, the task of presenting a hierarchy is one that has already been used in interfaces, in particular for presenting hierarchy in graphic user interfaces. We therefore exploit the common interface metaphor of collapse triangles, where a small triangle or other icon is shown where more detail exists, with the triangle orientation dependent on whether the details are being shown. Clicking on a collapse triangle then allows expansion or contraction of a subtree, as shown in Figure 2.

Transfer Function Selection: The second set of choices in the user interface relate not to the topological hierarchy, but to the transfer functions that can be applied to features. In keeping with the desire to keep the interface minimalistic to avoid overloading the user, we limit the transfer functions to six standard types which

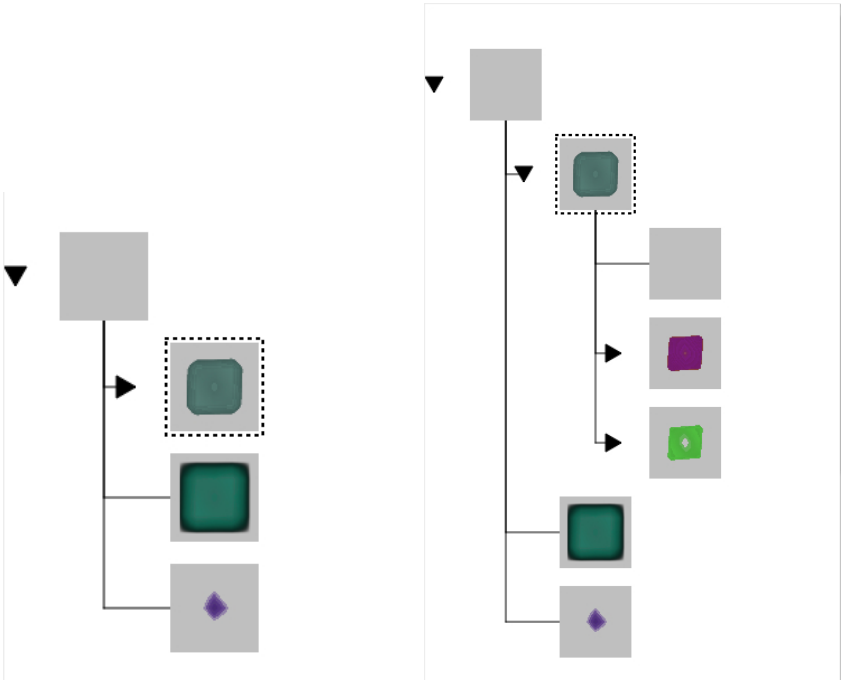

Fig. 2 Expanding a collapsed branch.

have proven useful in practice. These are shown in iconic form in panel (B) of Figure 1, and are discussed in more detail in Sec. 5. In addition, a further icon is shown with the currently available colour for rendering.

To assign these standard transfer function types to a particular feature, we have chosen to employ drag-and-drop assignment. For this, the user assigns different types and colours simply by dragging the type or colour from the relevant box in the tool palette in panel (B) (Figure 1) to the feature in panel (A). The changed transfer function is then applied immediately in panel (C) to the region represented by the thumbnail.

Visual Feedback: Once the user has assigned transfer functions to individual features, it is desirable to provide the user with visual feedback on the choices made to date. This is accomplished by showing a halo around each feature in the hierarchy, as shown in Figure 1. This halo is shown in the colour of the transfer function, and could be extended to indicate the type of transfer function as well.

In addition to these decisions, we recognised that the thumbnails in the topological gallery might be difficult to see, and provided pop-up enlargements of thumbnails whenever a user's pointer hovers over a thumbnail. In practice, as we will see below, the small scale of many features in the data prevents this from being as useful as we had wished.

We note that many of the choices made in this interface could be modified: for example, we chose not to propagate transfer functions to children except those belonging to the same branch as the parent in the branch decomposition. Similarly, where no transfer function was assigned explicitly, we chose to apply a standard default transfer function of constant opacity grey. Both of these could easily be varied, but would be unlikely to affect our conclusions that the interface was more limited than we had originally hoped.

Thus, by using simple but meaningful operations, we simplify the operation of the interface so that specialised knowledge is not required to use it. We next turn to the choice of simplification methodology, as this is significant in practice.

4 Hierarchical Contour Trees

As we have seen in the previous section, the Topological Gallery interface is based on exploiting the hierarchy induced by the simplification process. While the existing work is primarily based on the branch decomposition, this is not necessarily the optimal choice for interface purposes.

To illustrate this point, consider Figure 3. The upper half of this image shows the branch decomposition of a contour tree for a small data set. As individual leaves are removed, larger branches are built up, with the result that the major branch in this case has three children. For larger contour trees, the number of children may become arbitrarily large, with the result that the first level of children of the major branch may have a branching factor too great to represent conveniently in an interface. For example, the root branch in the fuel dataset from volvis.org has 42 immediate children, while the root branch in the lobster dataset has $2,598$ immediate children.

In addition to this, many of the direct children of a major branch are much smaller than the parent. Thus, representing the branch decompositions tends to result in an interface overloaded with small features at the expense of larger ones.

An additional problem arises in the treatment of the saddles. Each feature is represented by a transfer function highlighting the saddle at which the feature terminates. If the parent feature is not also shown at or near that saddle, it becomes difficult to interpret the relationship between features.

In comparison, the pruning hierarchy addresses both of these problems. First, as shown in the lower half of Figure 3, when a saddle is removed to create a major branch, this removal effectively splits the tree into three parts: the new branch at the saddle, and the two branches of the parent caused by splitting it at the saddle. Thus, the hierarchy is ternary in this instance, and expands much less rapidly than the branch decomposition. As a side effect, the features added here are strictly in order of importance, resulting in an interface in which fewer small features are shown.

Finally, the pruning hierarchy explicitly represents all branches at each saddle, easing the task of interpreting feature relationships.

However, while the pruning hierarchy is more suitable than the branch decomposition for the purposes of this interface, it is not without its drawbacks, the most

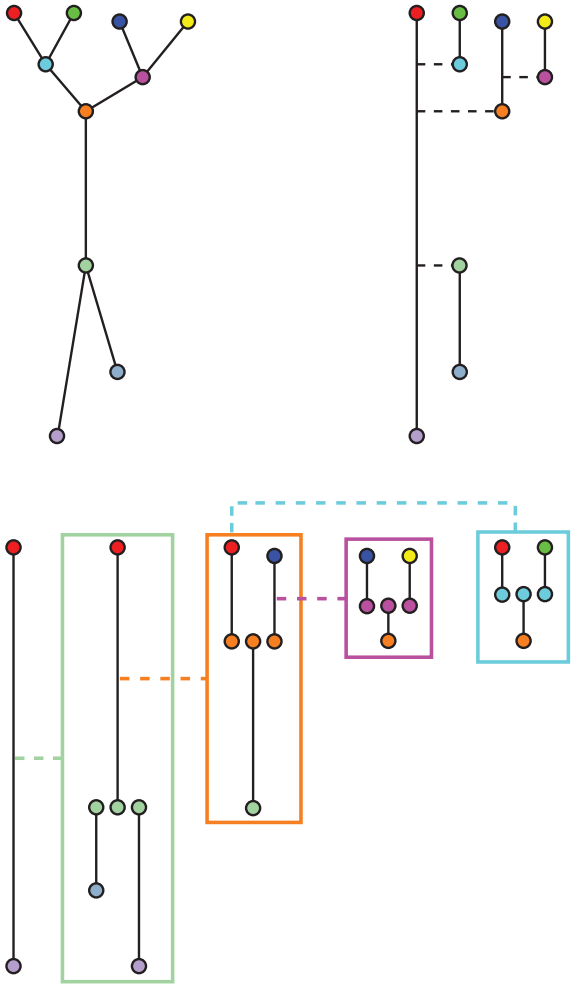

Fig. 3 In the upper illustration, we show a small example of the branch decomposition. In the lower illustration, we show the same example as a pruning hierarchy. Note how the branch decomposition leads to a very high branching factor unsuitable for interface purposes, while the pruning hierarchy keeps the branching factor more manageable.

significant of which is the extra memory it consumes. Compared to a tree resulting from the branch decomposition with n branches, the pruning hierarchy will result in $3(n-1)+1$ branches, and will require correspondingly more memory.

5 Transfer Function Design

Our goal is to develop a high-level interface that supports transfer function design based on topologically defined features without exposing users to an explicit representation of the contour tree. Since we are hiding the contour tree representation from the user, we also need to hide the explicit transfer function assigned to a branch. If we examine the common goals of a transfer function, certain key concepts arise: suppression of unwanted features, highlighting desired features and display of context are among the most common uses of a transfer function. We also chose these transfer function templates based on a review of related (topology-based) volume rendering work [7, 24] and experience gained from designing transfer functions manually for topology based volume rendering. With this in mind, it no longer becomes necessary for users to create their own transfer function on a per-branch basis. Instead, they can select it from a predefined set of high level operations that can be applied to gallery images and corresponds to a set of pre-defined transfer function templates with adjustable parameters:

Constant Opacity assigns the same opacity to the entire branch and is mostly used to suppress a branch entirely by assigning its opacity to zero.

Linear Ramp assigns a linear ramp of opacity from the extremum value of the branch to the saddle. Ramping up from almost transparent to almost opaque is the default choice for every branch in the interface as it frequently gives a good sense of all the features contained in the branch. In certain situations reversing the ramp will expose features which would otherwise be enclosed inside a larger feature.

Hat Shape assigns a linear ramp up from the saddle to the mid-point of the branch, and back down to the extremum. This function's primary purpose is to highlight features in the larger branches which contain multiple structures

Sharp Spike emulates isosurface extraction by suppressing the entire branch except for a user defined isovalue. In order to maintain solid surfaces, any value within a threshold value is treated as the same isovalue.

Gaussian Curve uses the formula described in [12] to create a gaussian distribution of opacity. It is used for highlighting features while retaining the context of their relationship with other features.

6 Implementation

We implemented the Topological Galleries interface using a combination of the Qt toolkit, Scott Dillard's *libtourtre* contour tree library, and a custom OpenCL-based topological volume renderer. As is customary for topological computation, we used symbolic perturbation[6] to guarantee unique critical isovalues. These were later removed from the contour tree as previously described[4].

Implementing the pruning hierarchy required adding two features to the libtourtre library: creating the pruning hierarchy from the branch decomposition and updating the vertex mapping. Since the branch decomposition algorithm in libtourtre stores

the children of a branch in a list based on the order in which they were pruned, it is simple to convert the branch decomposition into the pruning hierarchy.

Starting at the root branch and using a queuing system, we take the last child added to the current branch, and then split the parent at the child's saddle point into two components, the upper and lower section. The remaining children of the branch are added to these new branches according to whether their saddle points are above or below the first saddle of the first child. Once the children have all been reallocated, the two new branches are added as siblings to the first child of the parent branch, then all three children are added to the queue of branches.

Whenever a branch is split, the vertex to branch mapping becomes invalid so it must be updated. To do this each vertex in the dataset that maps to the parent branch is reassigned to one of the children based on whether the data value at that vertex is greater or less than the value at which the branch is split.

All code was implemented in C++ on an Apple Mac Pro with two 2.2GHz quad-core processors, 6GB of main memory and an Nvidia GeForce 8800GT video card with 512MB of video memory.

Since the thumbnails are generated from the same viewpoint as the main image, we optimised thumbnail rendering with a simple observation: the main image is simply a larger-scale composite of the thumbnail renderings in the hierarchical gallery. As the main image is updated for a given change in the transfer function, therefore, the same samples along the same ray can be used to recompute the thumbnail rendering for the corresponding feature.

We chose not to implement pre-integration or acceleration structures, as the goal of this work was proof-of-concept rather than to apply all possible optimisations.

7 Results

Our first test case is the hydrogen atom dataset. This dataset serves as a useful example to illustrate the capabilities of the topological galleries to show nested features. Figure 4 shows how the thumbnail view allows the user to assign different optical properties to individual regions, but also reveals a weakness that we had not initially anticipated. In this case, the yellow component in the centre of the rendering is of crucial importance in understanding the phenomenon being studied. However, since it is physically small, when it is rendered to a thumbnail, it becomes nearly invisible. We will comment further on this below.

Figure 5 shows the topological gallery applied to the nucleon dataset. In this instance, the interface made it relatively easy to set transfer functions visually, but the small size of some of the components again reduced the utility of the interface.

While our interface potentially allows users without a background in topology to gain insight in the nested structures of a data set, we came to the conclusion that it was significantly less useful than we had hoped, primarily because small components in the thumbnail gallery were hard to see, and even when magnified, were hard to understand without a spatial context.

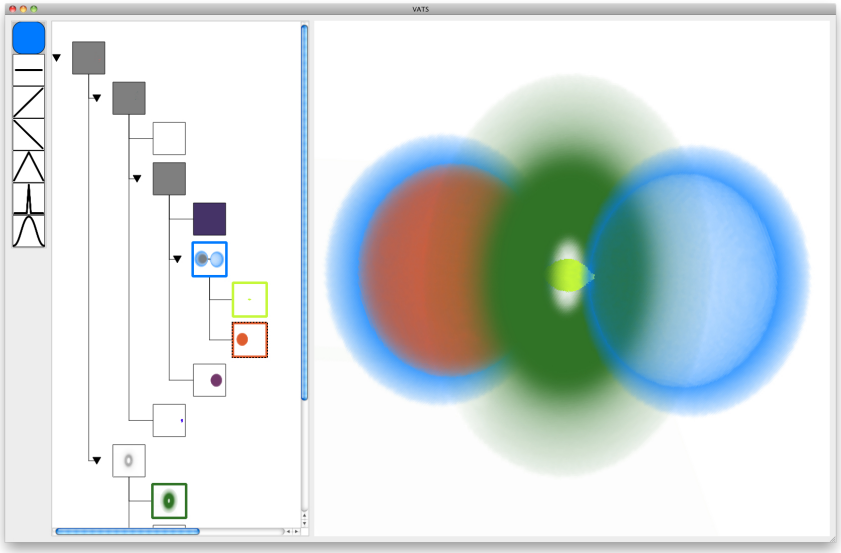

Fig. 4 Topological galleries for the hydrogen atom dataset. While the individual components in the rendering can be identified in the thumbnail gallery, small but significant components such as the yellow component in the middle are effectively invisible in the gallery.

The original design gallery concept was intended to show a range of possibilities for an entire image. As such, all pixels in each thumbnail were typically used. Here, where the thumbnails are being used to show individual subregions of the data, an unexpected problem arises: most of the thumbnails represent small features, whose projected area in the thumbnail is less than 1% of the thumbnail areas.

This occurs because, in order to maintain context, all thumbnails are rendered at global scale. A better solution may be to display features at a maximum size in the thumbnail and use the context zoom to show the position of the feature in the context of the current main image. But this implies rendering features at a local scale, and losing all possibility of contextual information.

In retrospect, it is not surprising that this was a problem, as it was already known that the size of features in the contour tree tends to fall off rapidly. For example, the simplification panel used for the flexible isosurface interface[4] used a log-log plot of feature size versus number of features. Even though the projected area falls off somewhat less fast than the volume of a feature, we observed that most features more than 4 or 5 branches down the pruning hierarchy were on the order of a few percent of the thumbnail size.

Moreover, our current implementation of the pruning hierarchy results in the generation of many empty thumbnails. This is a result of the branch map being split across so many branches: there are multiple branches representing a single vertex and the vertex only maps to one of the branches.

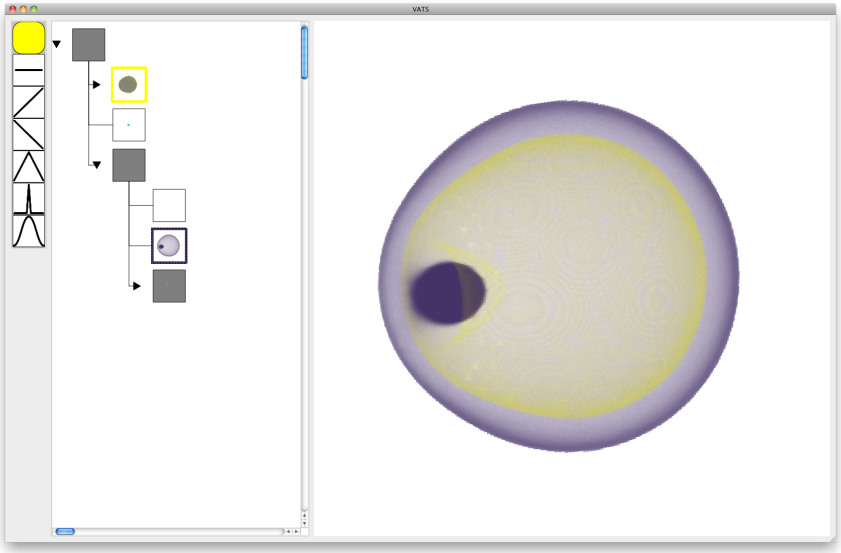

Fig. 5 Topological galleries for the nucleon dataset. As with the hydrogen dataset, while the gallery clearly allows direct selection of individual features, small features are unmanageably small, and the utility is accordingly diminished.

As a result of these factors, for all the appeal of a user-friendly interface for topology, the practical value is more limited than hoped.

8 Conclusions and Future Work

We have applied the design galleries metaphor to topology-controlled volume rendering. While this metaphor greatly simplifies the design of transfer functions and shows promise in that regard, there is a key problem that needs to be overcome for this approach to be useful. Feature size in the pruning hierarchy drops off considerably, and this rapid drop-off results in many almost empty images. These empty images make it very difficult to design transfer functions effectively. In the future, we will examine this behavior for a variety of data sets to determine whether it follows any rules, and whether there are different classes of data indicative of whether the gallery metaphor is likely to fail. We will also look into alternate simplification metrics and examine their impact on the problem. Finally, we will determine whether there are alternate means of re-combining features that are split by the pruning-hierarchy as a means to generate more meaningful gallery images.

Acknowledgements This work was supported by the Director, Office of Science, Office of Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

1. BAJAJ, C. L., PASCUCCI, V., AND SCHIKORE, D. R. The Contour Spectrum. In *Proceedings of IEEE Visualization 1997* (1997), pp. 167–173.
2. BOYELL, R. L., AND RUSTON, H. Hybrid Techniques for Real-time Radar Simulation. In *Proceedings of the 1963 Fall Joint Computer Conference* (1963), IEEE, pp. 445–458.
3. CARR, H., AND SNOEYINK, J. Path seeds and flexible isosurfaces using topology for exploratory visualization. In *Data Visualization 2003 (Proceedings of VisSym 2003)* (New York, NY, 2003), ACM Press, pp. 49–58.
4. CARR, H., SNOEYINK, J., AND VAN DE PANNE, M. Simplifying Flexible Isosurfaces with Local Geometric Measures. In *Proceedings of IEEE Visualization 2004* (2004), pp. 497–504.
5. CORREA, C., AND MA, K.-L. Size-based transfer functions: A new volume exploration technique. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1380–1387.
6. EDELSBRUNNER, H., AND MÜCKE, E. P. Simulation of Simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics 9*, 1 (1990), 66–104.
7. FANG, S., BIDDLECOME, T., AND TUCERYAN, M. Image-based transfer function design for data exploration in volume visualization. In *Proceedings of IEEE Visualization 1998* (Los Alamitos, CA, USA, 1998), IEEE Computer Society Press, pp. 319–326.
8. FUJISHIRO, I., TAKESHIMA, Y., AZUMA, T., AND TAKAHASHI, S. Volume data mining using 3D field topology analysis. *IEEE Computer Graphics and Applications 20*, 5 (Sept./ Oct. 2000), 46–51.
9. HARVEY, W., AND WANG, Y. Topological landscape ensembles for visualization of scalar-valued functions. *Computer Graphics Forum 29*, 3 (2010), 993–1002.
10. KETTNER, L., ROSSIGNAC, J., AND SNOEYINK, J. The Safari Interface for Visualizing Time-Dependent Volume Data Using Iso-surfaces and Contour Spectra. *Computational Geometry: Theory and Applications 25*, 1-2 (2001), 97–116.
11. KINDLMANN, G., WHITAKER, R., TASDIZEN, T., AND MOLLER, T. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of IEEE Visualization 2003* (Washington, DC, USA, 2003), IEEE Computer Society, p. 67.
12. KNISS, J., PREMOZE, S., IKITS, M., LEFOHN, A., HANSEN, C., AND PRAUN, E. Gaussian transfer functions for multi-field volume visualization. In *Proceedings of IEEE Visualization 2003* (Washington, DC, USA, 2003), IEEE Computer Society, p. 65.
13. LEVOY, M. Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8*, 3 (May 1988), 29–37.
14. MARKS, J., ANDALMAN, B., BEARDSLEY, P. A., FREEMAN, W., GIBSON, S., HODGINS, J., KANG, T., MIRTICH, B., PFISTER, H., RUML, W., RYALL, K., SEIMS, J., AND SHIEBER, S. Design galleries: a general approach to setting parameters for computer graphics and animation. In *Proceedings of ACM SIGGRAPH '97* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 389–400.
15. MAX, N. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (June 1995), 99–108.
16. OESTERLING, P., HEINE, C., JÄNICKE, H., AND SCHEUERMANN, G. Visual analysis of high dimensional point clouds using topological landscapes. In *Proceedings of IEEE Pacific Visualization 2010* (2010), S. North, H.-W. Shen, and J. van Wijk, Eds., pp. 113–120.
17. PASCUCCI, V., COLE-MCLAUGHLIN, K., AND SCORZELLI, G. Multi-resolution computation and presentation of contour trees. *Algorithmica 38*, 2 (October 2003), 249–268.

18. SABELLA, P. A rendering algorithm for visualizing 3D scalar fields. *Computer Graphics (Proceedings of ACM SIGGRAPH 88) 22*, 4 (1988), 51–58.
19. TAKESHIMA, Y., TAKAHASHI, S., FUJISHIRO, I., AND NIELSON, G. M. Introducing topological attributes for objective-based visualization of simulated datasets. In *Proceedings of Volume Graphics 2005* (2005), pp. 137–145.
20. TENGINAKAI, S., LEE, J., AND MACHIRAJU, R. Salient Iso-Surface Detection with Model-Independent Statistical Signatures. In *Proceedings of Visualization 2001* (2001), pp. 231–238.
21. WANG, L., AND MUELLER, K. Harmonic colormaps for volume visualization. *IEEE/EG Symposium on Volume and Point-Based Graphics* (2008), 322–325.
22. WEBER, G. H., BREMER, P.-T., AND PASCUCCI, V. Topological landscapes: A terrain metaphor for scientific data. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (November/December 2007), 1416–1423. LBNL-63763.
23. WEBER, G. H., DILLARD, S. E., CARR, H., PASCUCCI, V., AND HAMANN, B. Topology-controlled volume rendering. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (March/April 2007), 330–341.
24. ZHOU, J., AND TAKATSUKA, M. Automatic transfer function generation using contour tree controlled residue flow model and color harmonics. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1481–1488.

Disclaimer