Computers & Graphics 60 (2016) 55–65

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on SIBGRAPI 2016

Graph-based interactive volume exploration

Daniel Ponciano, Marcos Seefelder, Ricardo Marroquim *

Cidade Universitária, Centro de Tecnologia, bloco H, sala 319, Rio de Janeiro - RJ CEP 21941-972, Brazil

ARTICLE INFO

Article history:
Received 2 March 2016
Received in revised form
20 May 2016
Accepted 28 June 2016
Available online 15 August 2016

Keywords:
Volume exploration
Graph algorithms

ABSTRACT

The exploration of volumetric datasets is a challenging task due to its three-dimensional nature. Segmenting or classifying the volume helps to reduce the dimensionality of the problem, but there remains the issue of searching through the feature space in order to find regions of interest. This problem is aggravated when the relation between scalar values and spatial features is unclear or unknown. To aid in the identification and selection of significant structures, interactive exploration methods are important, as they help to correlate the volumetric rendering with the scalar data domain. In this work, we present a semi-automatic method for exploring volumetric datasets using a graph-based approach. First, we automatically classify the volume from a 2D histogram, following ideas from previous proposals. Then, through a graph structure with dynamic edge weights, a hierarchy is generated to identify similar structures. The final hierarchy allows for an interactive and in-depth volume exploration by splitting, joining or removing regions in real-time.

1. Introduction

Volume visualization has grown to be an accepted and useful tool in many communities [1]. However, due to its volumetric nature, naively applying rendering techniques may lead to a poor inspection of the dataset's internal structures. In most situations, at least some effort must be placed into segmenting the volume or designing transfer functions to have a clear insight about the feature space.

By separating the volume into regions, internal structures can be better isolated and visualized. Nevertheless, volumetric segmentation and classification is a challenging task, and for the general case it still requires manual intervention [2,3]. Furthermore, the segmented regions must be presented in a meaningful way as their correlation with the volumetric rendering is important to provide an intuitive exploration of the dataset.

Transfer functions can further help the visualization by mapping ranges of scalar values to colors and opacities, but also imply in some, either manual or automatic, segmentation of the volume. Moreover, when going beyond one-dimensional transfer functions, their design becomes a complex task [4].

The issue is even more aggravated if there is no deep understanding of the dataset beforehand, for example, when one is exploring the data without previous knowledge about how its internal structures relate to the scalar values. Even for researchers in visualization, sometimes it is hard to extract meaningful images from volumetric data.

Motivated by the need to intuitively and interactively explore a volumetric dataset, we propose a method to navigate a graph-based hierarchy generated from a previous classification of the volume. The hierarchy generation only takes around one minute or less, and once ready, exploration can be performed interactively by hiding, splitting and joining regions. It may serve as an initial exploration of the feature space to quickly highlight the internal structures (Fig. 1), or as a first step in designing transfer functions. The main contribution of our proposal is twofold:

- our graph-based structure is compact and efficient, allowing for real-time exploration, and a natural correlation of regions in the 2D domain and the volumetric rendering;
- by fine controlling the hierarchy generation we are able to isolate noise regions and produce a balanced structure that keeps most relevant segments near the top of the hierarchy, avoiding loading the user with tedious tasks.

The paper is divided into the following manner. In Section 2 we review the most related works and those that inspired our approach. In Section 3 we briefly overview Wang's method for automatically segmenting the volume's 2D histogram. To achieve a balanced structure from the segmentation, we propose new criteria to join segments as described in Sections 4.1 and 4.2. In Section 5 we describe how the hierarchy can be interactively explored. Results are shown in Section 6, followed by conclusions and future research directions in Sections 7 and 8, respectively.

* Corresponding author. Fax: +55 21 3938 8676.

http://dx.doi.org/10.1016/j.cag.2016.06.007
0097-8493/

Fig. 1. The images show the Head dataset at three different moments during exploration. The bottom figures are the corresponding histogram cells that are being rendered. The red and purple regions had their opacity values manually reduced. Between the first and second images the purple region is deleted. In the sequence, the red region is also deleted, remaining only the region representing the skull. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

2. Related work

Several proposals seek to segment the volume or design transfer functions in a semi-automatic or automatic fashion. Other researchers have focused on how to interactively explore the feature domain. Among these, we cite the most relevant methods in regard to our work. For a recent and more in depth state of the art report on the topic, we refer the reader to [5].

Huang and Ma [6] propose the RGVis, an interactive region-growing method to segment the volume and generate transfer functions. Correa and Ma [7] propose a size-based criterion to classify the volume. In another work, they further propose ambient occlusion as a classification criterion [8], and later introduce the idea of visibility histograms as a way to design transfer functions [9].

Kniss et al. [10] propose a collection of widgets to interactively design multidimensional transfer functions. Park and Bajaj [11] describe a method that specifically alleviates the issue of overlapping features in the 2D histogram space. Pinto and Freitas [12] propose a method for designing multi-dimensional transfer functions by reducing the dimensionality, where the exploration occurs on a reduced two-dimensional space.

Wu and Qu [13] propose a system to manipulate transfer functions from the direct volume rendering, using an optimization approach. Users can fuse and delete regions directly from the 3D view. In a similar direction, Guo et al. [14] introduce a What You See Is What You Get system for volume visualization, where the user explores the volume through sketches: operations such as coloring, changing opacity, erasing, and visual enhancements are directly applied on the volume. In a more recent work, Soundararajan and Schultz [4] also propose an approach to directly interact in the spatial domain, and discuss several classification techniques to aid in this task. Guo et al. [15] represent different transfer functions from the same dataset in a multi-dimensional scaling map, the transfer function map. This 2D space can be navigated to explore features in the volume data.

Tzeng et al. [16] use high-dimensional classification methods, such as neural networks and support vector machines, to build transfer functions. They employ a painting interface to segment regions and train the system to classify the rest of the volume. Maciejewski et al. [17] build 2D transfer functions using a non-parametric kernel density estimation to group similar voxels. After generating the function the user can further join, inflate or shrink regions. Praßni et al. [2] describe an uncertainty-aware volume segmentation, where a guided probabilistic approach is employed to alert the user about possible misclassifications. Lindholm et al. [18] describe a boundary aware reconstruction. Their method aims at reconstructing precise boundaries for each feature with a piecewise continuous model. However, they are only able to visualize 2D slices or small regions using their method due to performance issues, and rely on manually setting the transfer functions via widgets to classify the regions. Karimov et al. [3] describe an editing method to correct volumetric segmentation. Their system identifies possible segmentation defects and guides the user during an editing session. Shen et al. [19] propose a model-driven method, where a semantic model is used to label the volume's components.

Ip et al. [20] generate multilevel segmentation based on an intensity-gradient histogram. They use a hierarchy of normalized-cuts to segment the volume. From the automatic segmentation it is possible to interact with the transfer function to further explore the volume by subdividing or hiding segments.

Jönsson et al. [1] take a different route in exploring volumetric datasets, and propose a tool that should be intuitive enough for novice users. Their system is based on the automatic generation of design galleries to guide the user's choices.

Fujishiro et al. [21] propose the automation of transfer functions based on the analysis of 3D field topology. In a more recent topology based approach, Wang et al. [22] automatically generate a 2D transfer function by segmenting the histogram based on the Morse–Smale theory [23], and using the topological hierarchy from the work of Bremer and Edelsbrunner et al. [24,25]. They introduce the notion of persistence as a metric for joining regions and generating an automatic segmentation. They also build a limited hierarchy to allow the user to further explore the volume.

3. Histogram generation

We follow the approach by Wang et al. [22] to classify the volume by segmenting a 2D histogram generated from the voxel data. We refer to cells as the elements created by the histogram

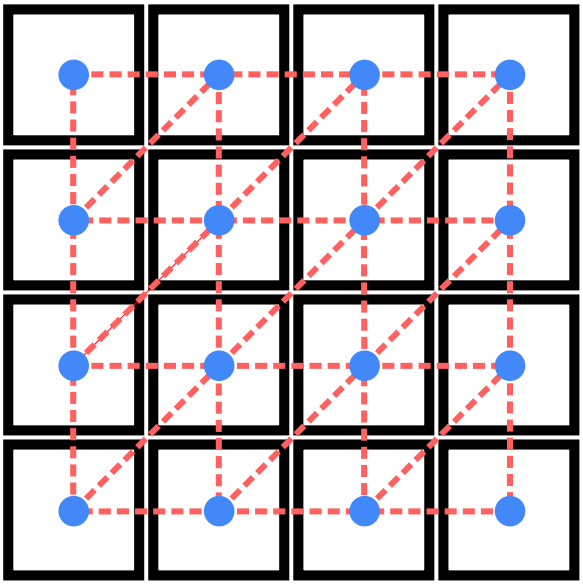

Fig. 2. Top view of the 6-connected mesh created from the histogram. Each blue mesh vertex is placed in a histogram bin and connected to six neighbors following a predetermined pattern. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

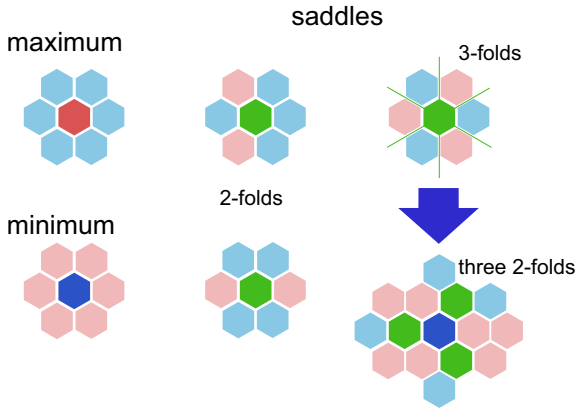

Fig. 3. Maximum (in red), minimum (in blue), and saddle points (in green). Three-folds are converted into three two-fold points to avoid ambiguities when defining the boundaries. In light-blue are neighbors with lower values than the central element, and in pink are neighbors with higher values than the central element. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

segmentation, and to regions when addressing the volume classification, that is, a region is the group of voxels associated with a histogram cell. In the rest of this section we briefly describe Wang's method to segment the histogram.

The 2D histogram is built using as axes any information per voxel derived from the spatial domain (ex. scalar value and gradient magnitude). From the histogram, a mesh is generated using a six-connected pattern, as illustrated in Fig. 2. Then, critical points using the mesh neighborhood are identified, i.e., maximum, minimum, and saddle points. All other points are labeled as regular points. To avoid ambiguities, only 2-fold saddles are admitted, so 3-folds are converted into three 2-fold points, as depicted in Fig. 3. The histogram cells' boundaries are created by descending from saddle points until minimum points are reached (Fig. 4). After creating the borders, it is trivial to classify regular points using, for example, a flood fill procedure starting from each maximum. Refer to [22] for more details.

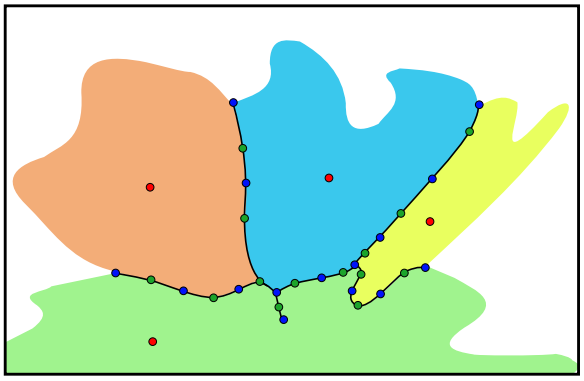

Fig. 4. A top view of the histogram cells. In this image the histogram frequency values are not important, only the labels. Maximum points are drawn in red, minimum in blue, and saddle points in green. The surrounding white area is a flat zero region and can be considered as empty for illustration purposes. Notice how between two saddle points there is always a minimum point, thus by following the descending paths from the saddles a boundary (drawn in black) is formed around the regions. In some cases the border does not fully divide a region, as happened within the green region. This is not a problem, as these stray points will be eliminated in a later step in our approach, when creating the graph structure. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

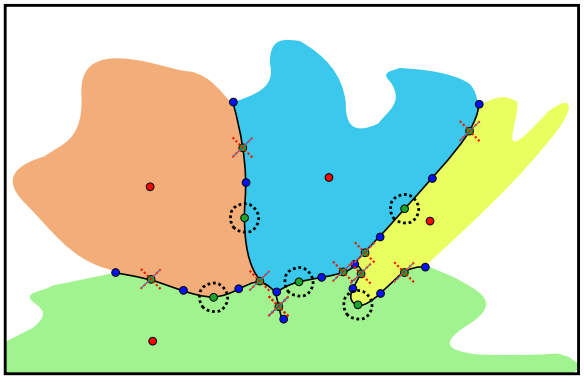

Fig. 5. Maximum points are depicted in red, minimum points in blue, and saddle points in green. Eliminated saddle points are marked with a red cross, and the remaining representative saddle point for each pair of adjacent cells is marked with a black dashed circle. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

4. Graph-based hierarchy

We build upon the initial histogram segmentation by proposing a graph structure to join adjacent cells, and consequently, unify similar volumetric regions. One of the great advantages of graphs is that they have a much simpler structure, and thus are easier to work with than dealing directly with the mesh and relying on topological operators.

Each cell is represented by its maximum point that defines one graph vertex. Each saddle point represents a graph edge and connects two vertices, or maximum points. If there are more than one saddle point between two cells, the edge will be created using the saddle point with lower histogram value, which we call the *representative saddle point*. The remaining saddle points are eliminated, as shown in Fig. 5.

Fig. 6 illustrates the graph generated from the structure in Fig. 5. In the next subsections we describe how we define weights for the edges. These weights guide the creation of the hierarchy described in Section 4.2.

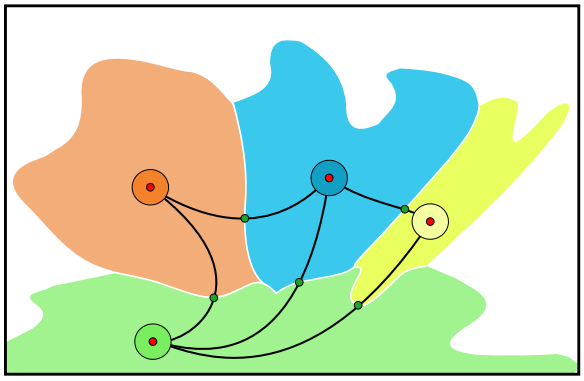

Fig. 6. A graph structure is generated by connecting adjacent cells. Maximum points are painted in red, and the representative saddle points in green. Each maximum point represents a graph vertex, and each representative saddle point an edge. Note how the structure now is much simpler than when working directly with the mesh topology. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

4.1. Weight function

It is important to join cells in a controlled manner, since the order of the join operations will also dictate how one navigates the hierarchy. The operations on the graph structure (i.e. the histogram cells) reflect directly on the volume classification and rendering. Cells representing meaningful volumetric regions, for example, should be kept separated until the last moment, while we wish to quickly absorb cells that may represent noise or non-important features.

Wang et al. used the concept of persistence as a single criterion to unify cells and offered a shallow navigation of the hierarchy. However, we noted that this criterion alone led to some issues. For example, in some cases it did not merge adjacent noise regions first, or merged distinct structures too early. Consequently, more exploration steps are necessary to isolate the noise or separate important regions.

To this end, we propose a definition of a weight function based on the original persistence value, and introduce three adjustment factors. Each factor is based on a criterion of the histogram cells, such as distance, height, and area, to differentiate similar persistence values. We start by reviewing the persistence criterion, and then detail and motivate the proposed factors.

4.1.1. Absolute height variation (persistence)

The *persistence* value is defined as the difference between the height of a representative saddle point and the lowest maximum between the two connecting vertices. It reveals the resistance of a cell to be absorbed by a neighbor with a higher maximum point:

$$\texttt{persistence} = \min\left(h_{max_i}, h_{max_j}\right) - h_{saddle_{ij}} \tag{1}$$

where h is the height of a point, and $saddle_{ij}$ is the representative saddle point that connects cells i and j. Note that the saddle point is always lower than the two maximum points, hence the *persistence* value is always positive.

Intuitively, a very shallow valley (saddle point) should offer low resistance to be absorbed, while a very deep one represents a clear separation between two peeks. Fig. 7 depicts this concept, and illustrates how very different situations might result in the same persistence value, motivating the three adjustment factors introduced below.

4.1.2. Maximum height variation

We would also like to incorporate lower cells to higher cells first, and leave adjacent cells with similar heights for later. The first weight factor reflects this criterion:

$$w_1 = 1 + \frac{h_{max_i}}{h_{max_j}} \tag{2}$$

where $h_{max_i} \leq h_{max_j}$, and $w_1 \in [1, 2]$.

The *maximum height variation weight* w_1 would prioritize joining the cells in the top case in Fig. 7, before the bottom one.

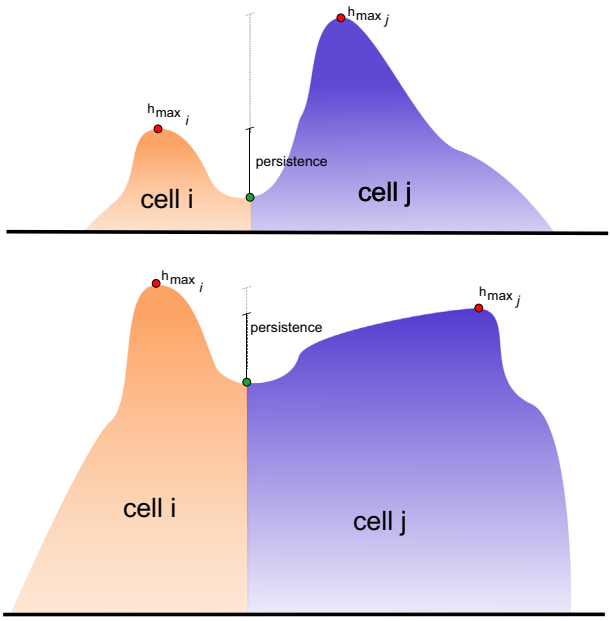

Fig. 7. Profile view of two regions of the histogram, where the heights are relative to the histogram frequency. The persistence value is defined as the height difference from the saddle point connecting the adjacent cells, and the lowest maximum between the two. This figure illustrates two different pairs of cells with the same persistence value. On the top, cell i is much lower than cell j, and the two maximums are closer, while on the bottom they have similar maximum heights and are farther apart.

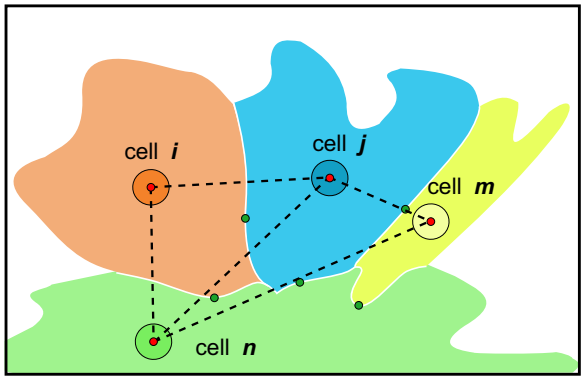

Fig. 8. According to the cell area variation weight w_2, cells i and j have similar areas and should be preserved, while cells j and m should offer less resistance to be merged. According to the maximum distance variation weight w_3, the distance between cells i and j is greater than the distance between cells j and m, so the second pair should be joined first.

4.1.3. Cell area variation

Here we take into consideration the difference between the cells' areas:

$$w_2 = 1 + \frac{A_{reg_i}}{A_{reg_j}} \tag{3}$$

where A_{reg} is the total area of a cell, $A_{reg_i} \leq A_{reg_j}$, and $w_2 \in [1, 2]$.

This second factor prioritizes the union of a small cell with a large one, instead of joining cells with similar areas. Fig. 8 illustrates this

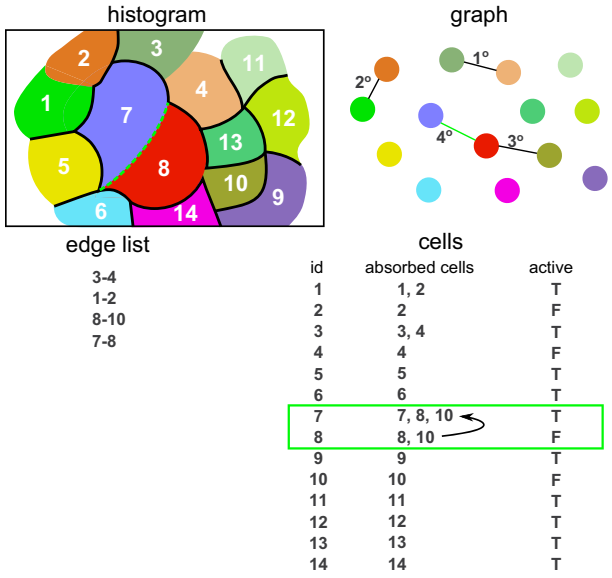

Fig. 9. This figure illustrates the fourth merge operation (green dashed line). When cells 7 and 8 are joined, cell 8 is absorbed by 7 ($h_{max_8} < h_{max_7}$). However, we still keep track of the previous state of cell 8. A flag marks the active state of each cell. The black lines separate regions that were not yet joined. The green box marks the altered lines for the current iteration. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

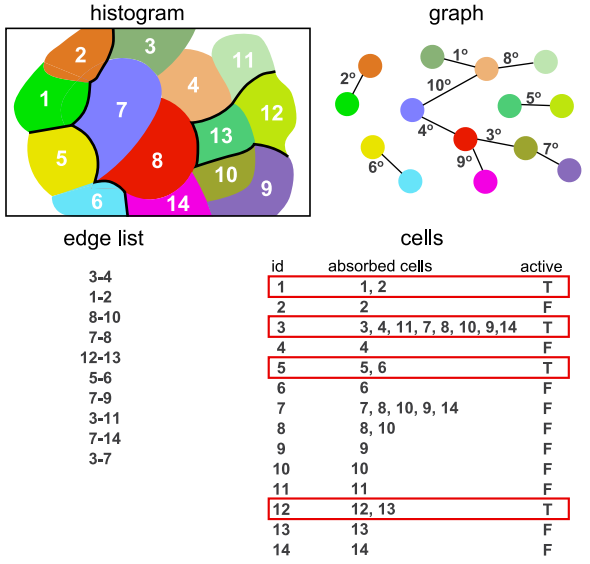

Fig. 10. The remaining cells after the union process for $k=4$. The final four active cells are marked in red in the cells table. The edge list contains the entire history of join operations. The black lines in the histogram indicate the borders between the remaining four cells. The edges of the graph on the top right corner indicate the insertion order of the MST. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

concept. It is important to join small similar cells early on, so they can be quickly isolated when exploring the hierarchy without splitting the region multiple times, specially when these cells represent regions containing only noise.

4.1.4. Maximum distance variation

The last weight regards the distance between the maximum points of two cells. To keep this factor in the same range $[1, 2]$ as the previous two, we divide by the maximum possible distance,

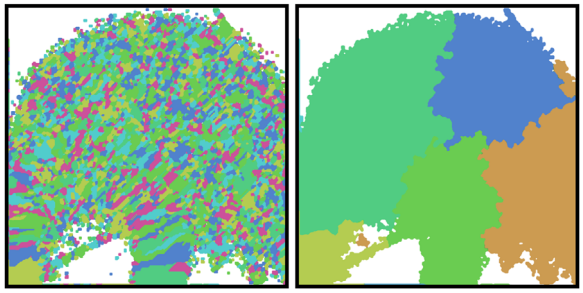

Fig. 11. The cells of a histogram after the initial segmentation, and after the unification process. The colors were randomly attributed to the cells. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

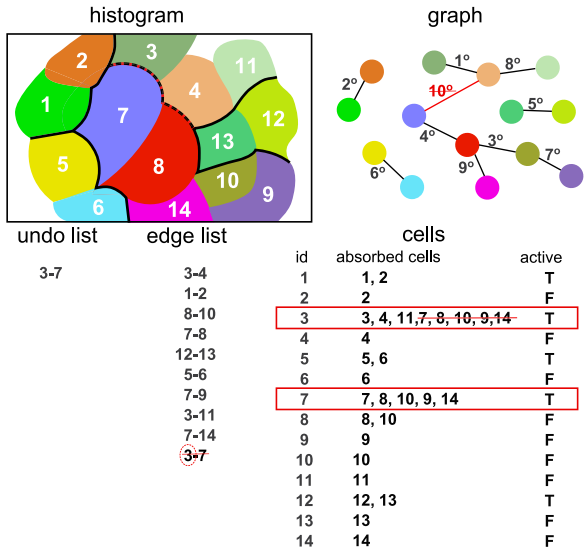

Fig. 12. A cell can be split by undoing a join operation. To split cell 3, the *edge list* is traversed in the reverse insertion order until an edge with index to cell 3 is found (in this case it is the first visited edge). The edge is removed and placed in the *undo list*, and the other cell is restored, in this case cell 7. The two red boxes mark the lines that were modified during the undo operation. The dotted red-black line in the histogram indicates the boundary that is restored with the split operation. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

that is, the diagonal of the histogram space:

$$w_3 = 1 + \frac{\text{dist}(max_i, max_j)}{\texttt{diag}} \tag{4}$$

where $\text{dist}(p_i, p_j)$ is the 2D Euclidean distance between the points $_{pi}$ and p_j. Figs. 7 and 8 depict the distance criterion.

The idea is that the inclination to join two cells should be inversely proportional to the distance between their maximum points. When the maximum points of two adjacent cells are close, there is a greater chance that they belong to the same region. For example, if a cell has all adjacent cells with similar persistence values, this factor would prioritize a merge with the cell with the closest maximum point.

4.2. Unifying regions

To unify cells using the described weights, we propose a graph-based approach using Minimum Spanning Trees (MST). A well known example of 2D image segmentation based on MSTs is the work of Felzenszwalb and Huttenlocher [26]. Our weights are,

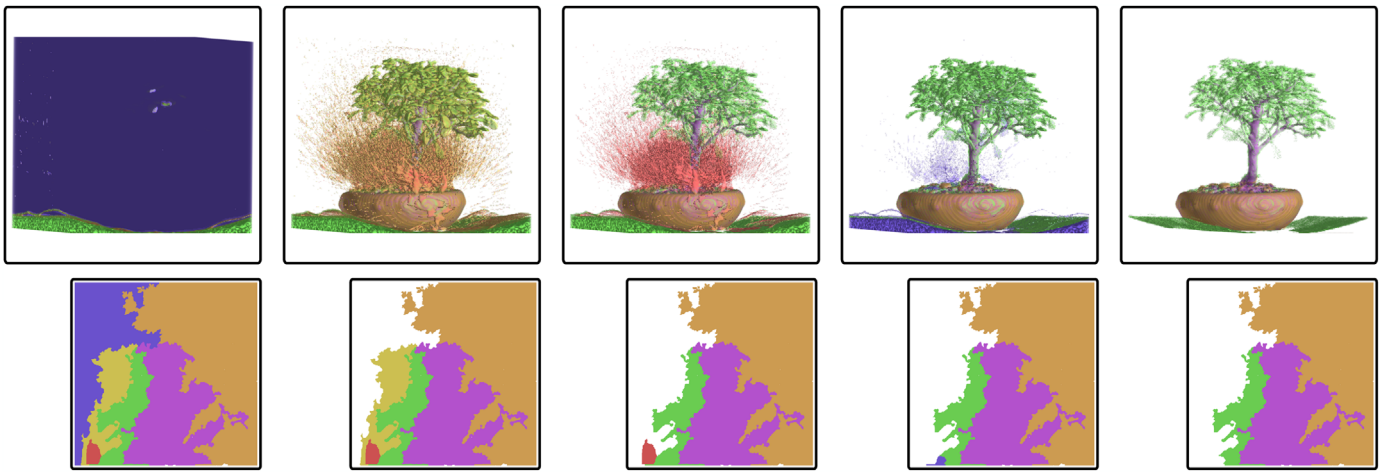

Fig. 13. Some snapshots of the first interactions to remove noise regions. With four delete operations, we are able to completely isolate the noise regions.

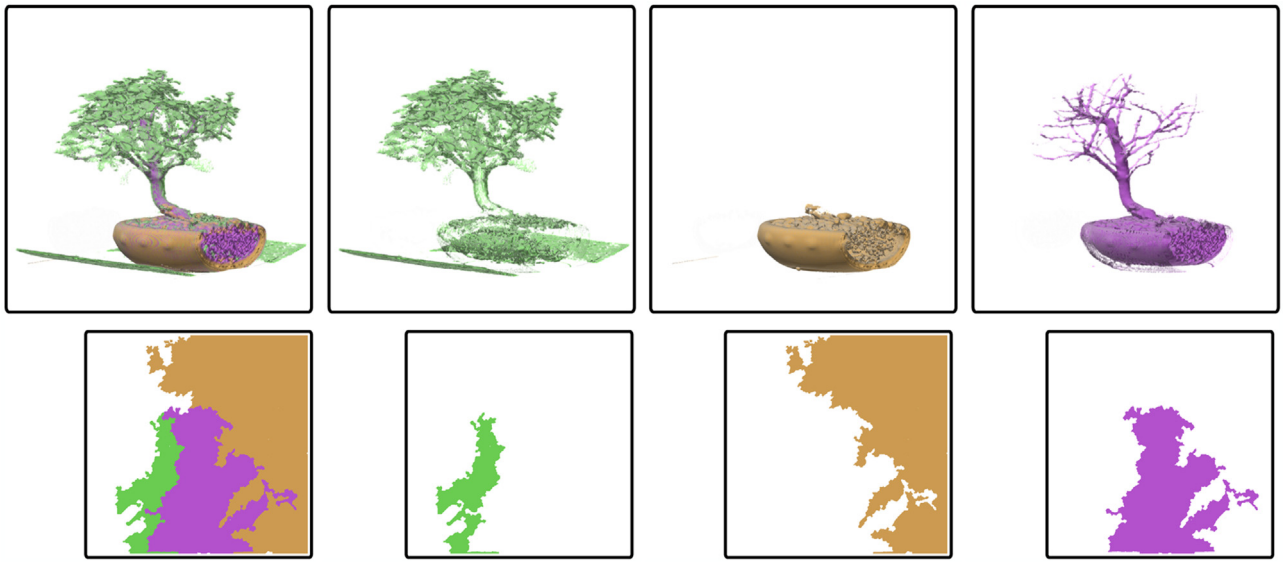

Fig. 14. The final result of the exploration sequence in Fig. 13, followed by the three regions rendered separately.

however, based on the histogram's frequency values, and not pixels colors.

We describe a simple modification of the classic Kruskal algorithm for generating the MST, and stop the iterations before inserting the $(n-k)$-th edge, where n is the number of edges of the initial graph, and k is the number of desired regions in the top level of our hierarchy. Inserting an edge is equivalent to joining two adjacent cells in our case. At the end of this process the volume is classified into k regions, where k is defined by the user. We have noted experimentally that $k=6$ is a fair starting value, and used this value for all our examples.

The final edge weight is defined by the product of the persistence value by the three weight factors previously described:

$$w_{\text{edge}} = \texttt{persistence} * \prod_{n=1}^{3} w_n \tag{5}$$

However, differently from the traditional graph scenario, every time a new edge is inserted, the weights change. When two cells are joined, the weights of the edges connecting other adjacent cells might be updated since the boundaries are modified.

Nevertheless, when this update occurs no edge receives a weight inferior to its current value, since joining cells does not create a lower maximum point, or decreases the distance between two adjacent maximum points, or decreases areas. Due to the greedy nature of the algorithm, it is guaranteed that the edges already in the MST will not be re-visited. Consequently, we only need to re-order edges that were not yet inserted in the MST and had their values modified.

When joining two cells, we preserve the one with the higher maximum. An important remark is that when cells are joined we maintain the information about the absorbed cell (lower maximum), as depicted in Fig. 9, and the corresponding edge is stored in an *edge list*. Once the MST is ready, with the *edge list* we are able to easily navigate the hierarchy by undoing operations, as will be detailed in Section 5.

We also keep track of the active state of each cell. At first, all cells are active. When a join operation occurs, the absorbed cell goes into an inactive state, while the preserved region (that now contains the absorbed region) remains active. The active flag is useful when manipulating regions, as will also be described in Section 5.

The join operation, or edge insertion, continues until a minimum number k of cells (or subgraphs) remains. Fig. 10 illustrates an example where $k=4$. Fig. 11 shows an example of the histogram segmentation of a real dataset, before and after the join operations.

The criteria based on height, area, and distance, not only aim at classifying the volumetric regions, but also at achieving a balanced

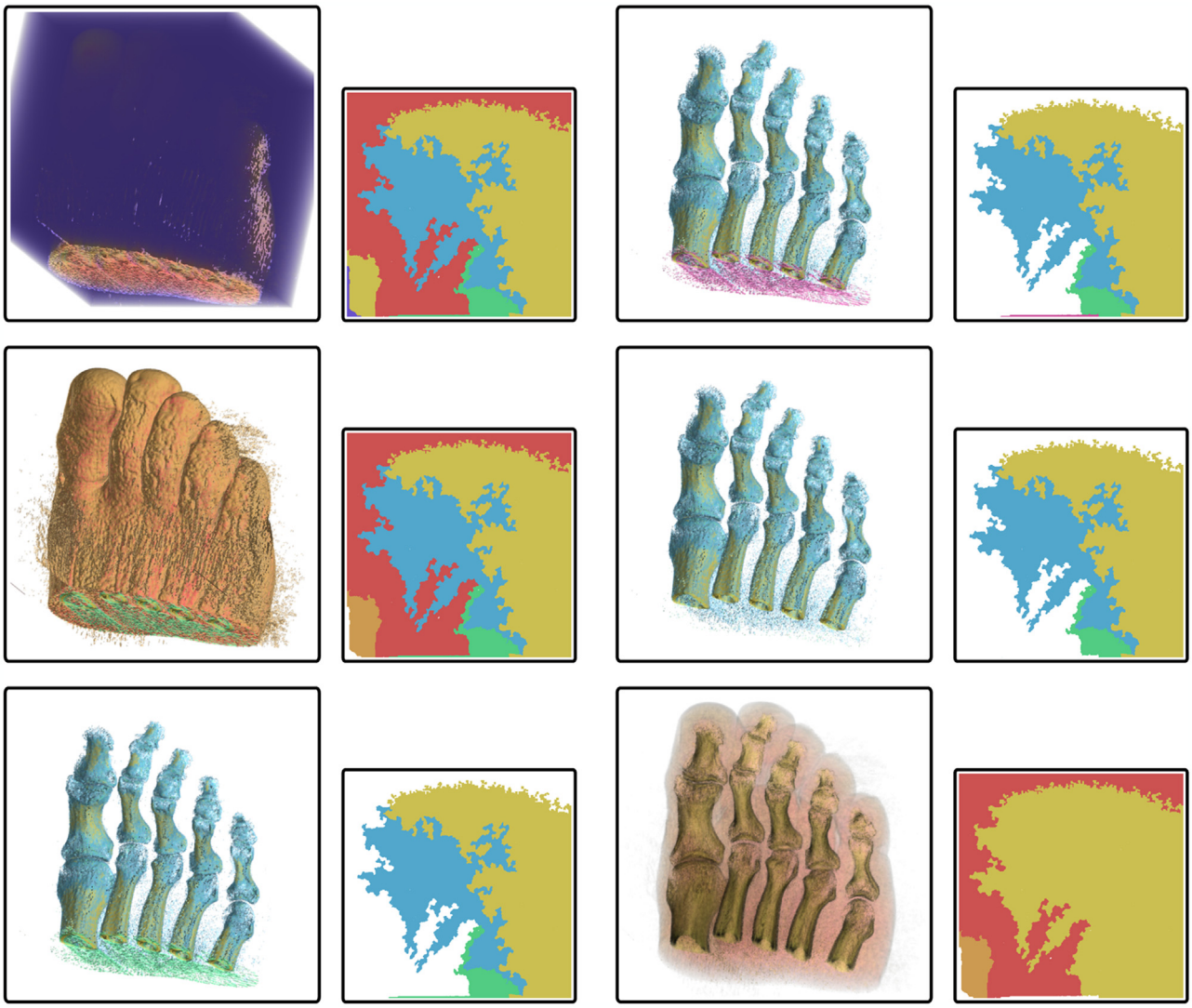

Fig. 15. Exploring the Foot dataset (order is top–bottom, left–right). The surrounding noise is quickly eliminated by removing a single region, leaving the soft tissues and bones (second image). After removing the red and orange regions, the green region still contains noise (third image), so its subdivided (fourth image) and the pink region is eliminated (fifth image). For the last image (right bottom corner) some deleted regions were restored with undo operations, and opacity values and colors were modified for the remaining regions. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

tree. The shallower the hierarchy, the less one has to navigate to separate structures or eliminate noise. In other words, a more balanced hierarchy implies in a less tedious volume exploration.

5. Interactive exploration

Once the MST is ready, navigating the corresponding hierarchy is straightforward. Three operations are permitted: join, split, and delete. Every performed operation is stored, so it can be easily reverted.

Delete: Removing a cell means ignoring the corresponding region during rendering. This can be achieved in time $O(1)$ by setting the active flag to false. This delete operation is recorded so it can be reversed.

Split: To subdivide a cell i, the *edge list* is traversed in reverse order, until the first edge with index to the cell i is found. The other index of this edge references the last join operation for this cell, i.e. the cell that was absorbed during the join operation. This join is reverted by removing the edge from the *edge list*, restoring the absorbed cell and resetting its active flag. Every time a split occurs, we keep track of the operation in an *undo list*. The split operation is depicted in Fig. 12. In the worst scenario, this operation may traverse the whole *edge list* taking $O(n)$ time, where n is the number of edges. However, this is a very improbable case, as we expect the hierarchy to be well balanced this bound is much closer to $O(\log n)$.

Join: Since the interactive exploration starts with the final MST, all performed join operations are already stored in the *edge list*. A join during the exploration phase is actually an operation that reverts a previous split. Since we store all the splits in the *undo list*, we can again expect this operation to be bounded by time $O(\log n)$.

In addition, it is also possible to adjust the opacity value for each region individually, or change a region's color. With this minimalist set of operations one can navigate and explore the hierarchy in a straightforward manner. Moreover, since the whole hierarchy is stored, it allows for a more in-depth exploration of the volume dataset when necessary. Nevertheless, fine structures and details are usually evidenced with a few splits.

The actual hierarchy is transparent to the user. What is shown is simply the current visible cells, and the volume rendering, as illustrated in Fig. 1, and the figures in Section 6. We render cells and corresponding regions with the same color to create the visual connection between these two representations.

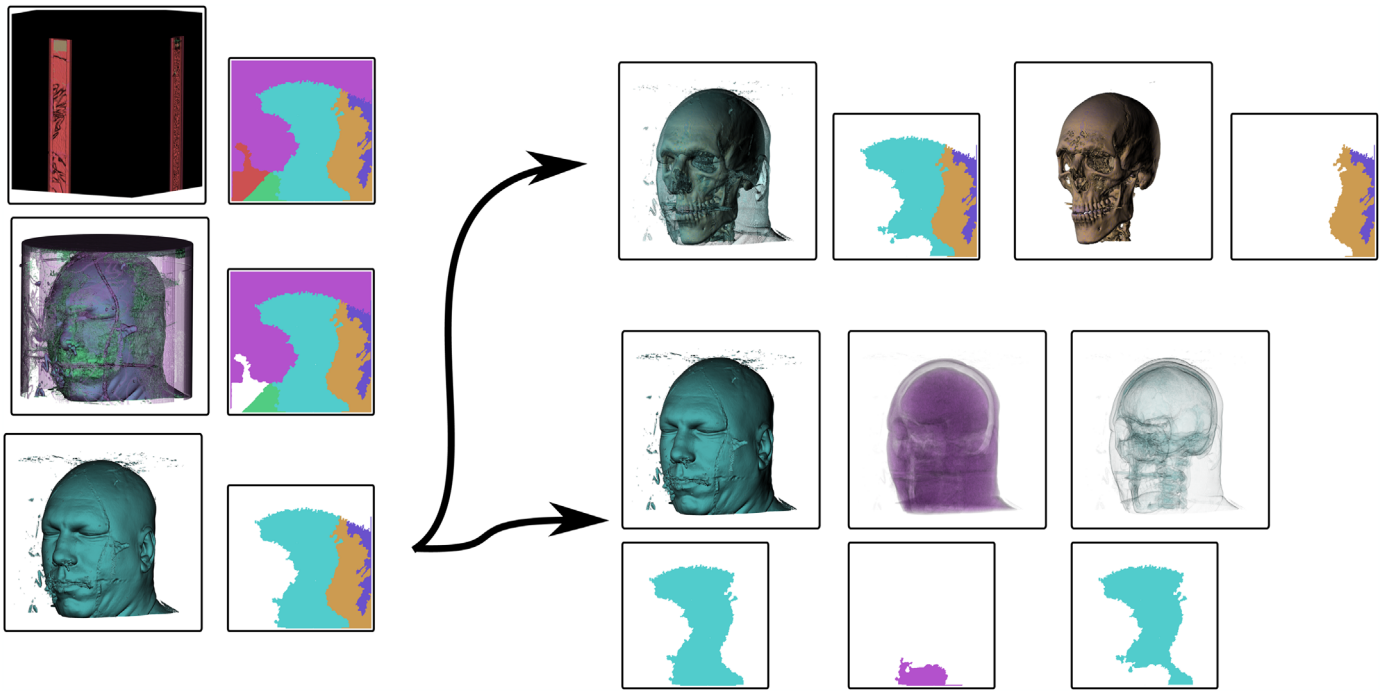

Fig. 16. Exploring the Head dataset. The left column shows the steps taken to remove the outer noise. From the last image on the left, two routes are illustrated, where the top row shows how to quickly arrive at the skull, while the bottom row explores the exterior head structure, where the cyan region is split and the two subregions are depicted separately with lower opacity values. Note that one can easily undo operations to return to a previous stage and follow another exploratory path. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

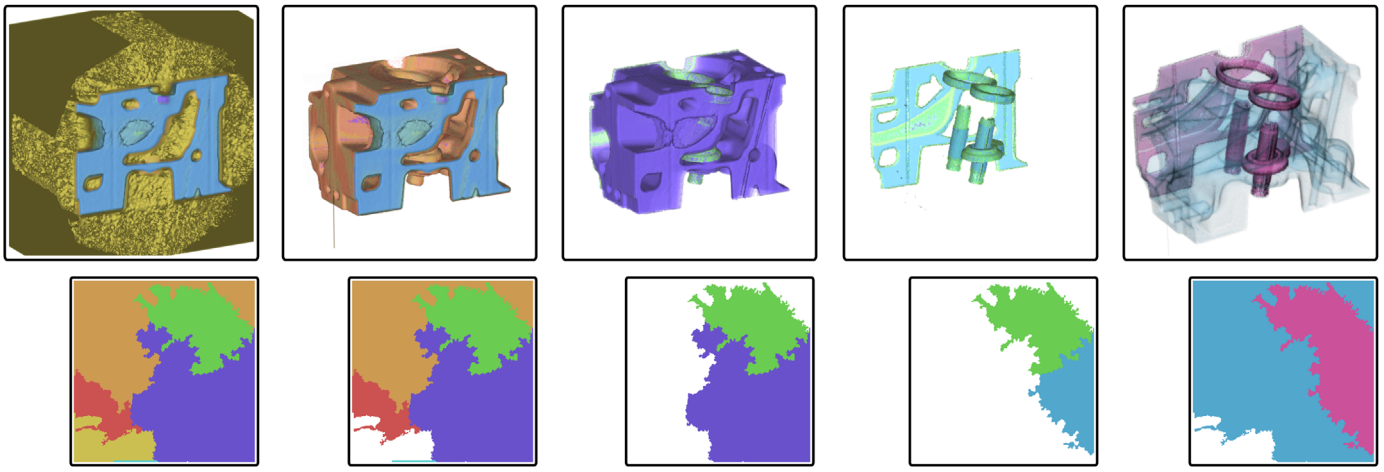

Fig. 17. By removing the surrounding noise and outer shells, we can easily isolate the internal structure. This separation is achieved with no prior knowledge of the dataset. The last image shows the same dataset after restoring the outer shell and changing color and opacity values. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

6. Results

The tests were performed with an i7 Quadcore 3.4 GHz with 16 Gb of Ram, and an nVidia 660GTX. To render the datasets we implemented a GPU ray-casting algorithm with illumination features. It takes in average one minute to process the volume: generate the histogram, segment the histogram, and build the hierarchy. For all tests the histograms' axes are the scalar value (density) vs gradient magnitude, and were generated with dimensions 256×256 and $k=6$. In all images in this section we depict the volume rendering and the corresponding cells windows that compose the user interface.

In Fig. 13 we show the first interactions on the Bonsai dataset to remove the typical surrounding noise. In Fig. 14 we show the resulting regions of the achieved segmentation from Fig. 13. In Figs. 15 and 16 the exploration of two other datasets are illustrated, the Foot and the Head.

The Engine Dataset is a good example where the hierarchy helps to reveal hidden features. The fine structures inside the engine can only be isolated by removing outer layers and splitting a region. Fig. 17 illustrates this procedure.

In Figs. 18 and 19 two more datasets, the Chest and the Carp, are depicted in different moments during exploration sessions.

Comparing with results from other methods we can point out some differences from the three works most similar to ours.

Maciejewski et al. [17] method produces a segmentation of the volume, but offers reduced exploration capability, since regions cannot be further split. From their results, it is notable that the noise of the Bonsai dataset cannot be decoupled from the trunk, for example. Ip et al. [20] offer a navigation of the hierarchy similar

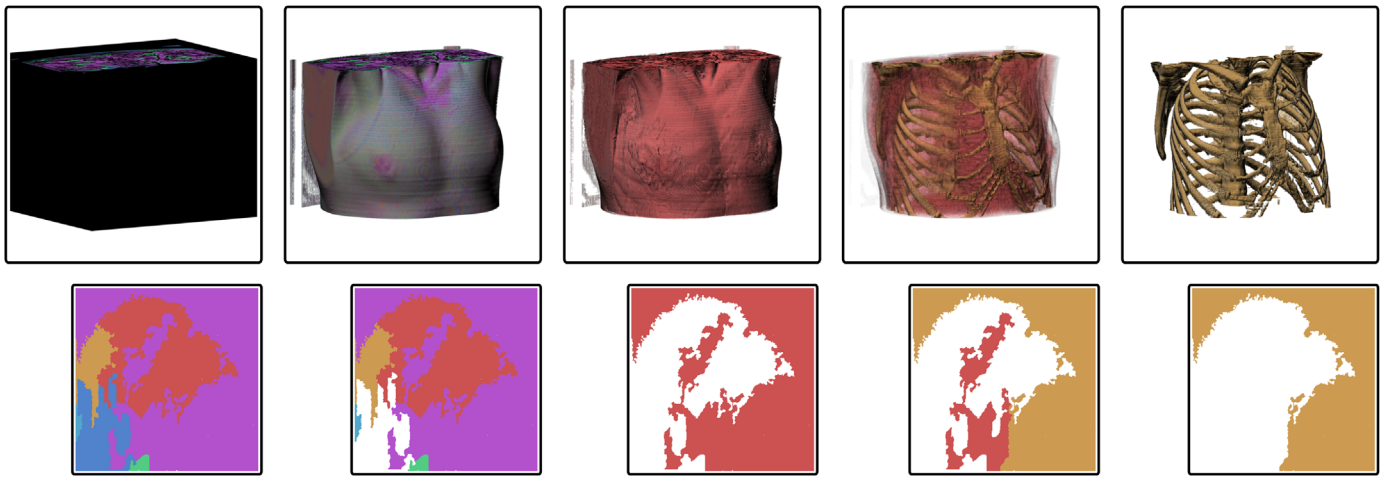

Fig. 18. By first removing the surrounding noise, and then deleting some outer regions, the inner part of the Chest is exposed in the central image. The remaining region had its opacity decreased, the color changed to red, and then the region was split to readily expose the bones in the interior. Finally the red region was deleted remaining only the rib cage. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

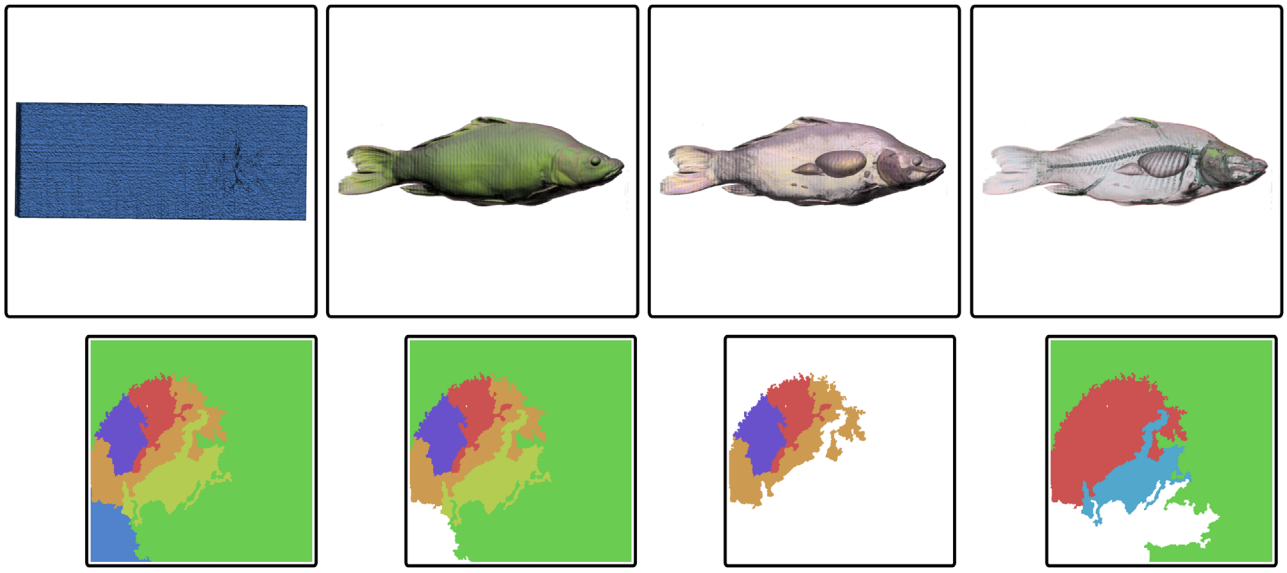

Fig. 19. The Carp dataset at different points during an exploration session. For the last image on the right, some regions were painted with the same color, and the opacity values were modified. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

to ours, but their method does not cleanly separate some structures. This fuzzy segmentation can be noted in their results from the Foot and Head datasets, for example, where the skin is not detached from the surrounding noise.

Wang et al. [22] serve as the basis for generating the initial segmentation for our method, but we have focused on the navigation structure after the segmentation. Instead of using only the persistence value to guide the join process, we added three adjustment factors. This avoided joining significant regions first, or leaving small noise regions to the end, where they would be placed at the top of the hierarchy. We illustrate these issues with the following examples. Fig. 20 shows the case where the underlying structure is on the first sublevel of the hierarchy when applying the factors, or hidden in the fourth sublevel when only the persistence metric is used. Fig. 21 illustrates another situation where with only the persistence metric it becomes difficult to isolate the entire structure. In this case it was not possible to separated the ribs in one single region, and to promptly eliminate the adjacent noise.

7. Conclusions

In this paper we propose an interactive and intuitive way to explore volumetric datasets. A hierarchical structure is generated from a 2D histogram using a graph-based procedure, where the edge weights control the resistance to join two adjacent cells. We propose a weight function based on the persistence value and three correction factors with two main goals: achieving a more balanced structure; and controlling the join operation to merge less significant regions first. Once the edges are attributed weights, we follow ideas from graph algorithms to join similar cells. The resulting hierarchy can then be navigated using three simple operations: join, split and delete. Since our graph-structure is lightweight, we can navigate the whole hierarchy in real-time.

By visually relating the 2D domain with the volumetric rendering, the navigation becomes intuitive even in the face of unfamiliar datasets, where the range of scalar values of interesting spatial features are not known beforehand. We showed through a series of examples how the method is able to isolate noise regions as well as reveal fine features that may be difficult to spot or to

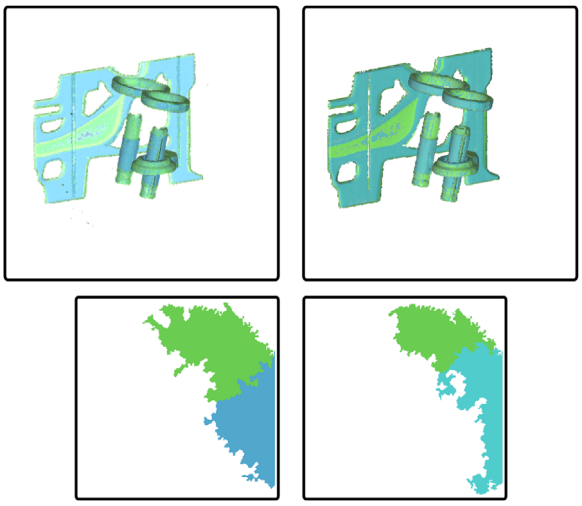

Fig. 20. The image on the left is the result using the additional three weight factors, and on the right using only the persistence value. The final results are very similar, apart from small variations on the histogram's regions. Nevertheless, when employing the factors only one split operation was necessary to reveal the internal structure, while without the factors four splits were necessary, and the extra operations did not reveal any additional relevant structure.

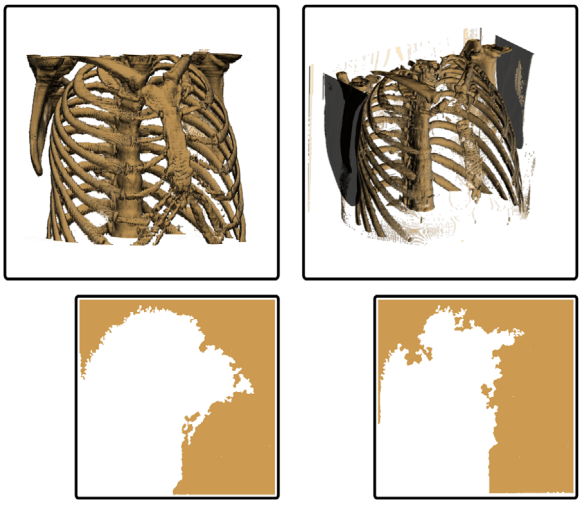

Fig. 21. The left and right images show the results with and without the three weight factors, respectively. Using the factors the ribs were clearly separated in one region, while without some surrounding noise persists and part of the bones remained in another region, that was not easily traceable. The lateral noise of the right image can actually be removed but only with some effort, that is, another sequence of five split and delete operations.

separate manually. We also illustrated how our three correction factors improve upon using only the persistence value.

We do not advocate to achieve the best volume classification, as there are some more advanced techniques to do so, but to allow for an interactive exploration of the datasets main features. This could be used for example as an initial inspection of the volume to highlight its main features, or could be combined with other classification methods to refine the visualization. It could also help as a first step in designing more complex transfer functions.

8. Future work

The main limitation of our method is, of course, that the navigation is restricted by the generated hierarchy. Even though it is very helpful for an initial exploration, we would like to explore ways to fine tune the classification. We have used Wang's method for the initial histogram segmentation, but in fact other techniques could be adapted to work with our graph-based hierarchy.

Another idea in this direction is to explore a dynamic histogram generation, that is, to recreate the histogram and the hierarchical structure during navigation for specific regions. This would allow for more freedom during navigation, since a part of the model could be isolated and treated separately. It would also be interesting to explore different attributes when generating the histogram, apart from scalar and gradient values.

We have not at this point, explored ways to automatically set opacity values for each region. This is left for the user during navigation. However, the weight parameters could give a good indication to which active regions are more or less important.

Finally, enhancements to the interaction decisions could be made by, for example, evidencing regions in volumetric space when selecting corresponding cells. One straightforward way to achieve this effect is by raising the opacity of the selected region while lowering the opacity of the others.

Acknowledgments

We would like to acknowledge Brazilian funding agencies CAPES (Coordination for the Improvement of Higher Education Personnel) for the grant of the first author, and CNPq (National Counsel of Technological and Scientific Development) for the grant of the second author.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cag.2016.06.007.

References

[1] Jönsson D, Falk M, Ynnerman A. Intuitive exploration of volumetric data using dynamic galleries. IEEE Trans Vis Comput Graph 2016;22(1):896–905. http://dx.doi.org/10.1109/TVCG.2015.2467294.

[2] Praßni JS, Ropinski T, Hinrichs K. Uncertainty-aware guided volume segmentation. IEEE Trans Vis Comput Graph 2010;16(6):1358–65. http://dx.doi.org/10.1109/TVCG.2010.208.

[3] Karimov A, Mistelbauer G, Auzinger T, Bruckner S. Guided volume editing based on histogram dissimilarity. Comput Graph Forum 2015;34(3):91–100. http://dx.doi.org/10.1111/cgf.12621.

[4] Soundararajan KP, Schultz T. Learning probabilistic transfer functions: a comparative study of classifiers. Comput Graph Forum 2015;34(3).

[5] Ljung P, Krüger J, Groller E, Hadwiger M, Hansen CD, Ynnerman A. State of the art in transfer functions for direct volume rendering. Comput Graph Forum 2016;35(3):669–91. http://dx.doi.org/10.1111/cgf.12934.

[6] Huang R., Ma K.L. Rgvis: region growing based techniques for volume visualization. In: 11th Pacific conference on computer graphics and applications, 2003. Proceedings, 2003, p. 355–63. http://dx.doi.org/10.1109/PCCGA.2003.1238277.

[7] Correa C, Ma KL. Size-based transfer functions: a new volume exploration technique. IEEE Trans Vis Comput Graph 2008;14(6):1380–7. http://dx.doi.org/10.1109/TVCG.2008.162.

[8] Correa C, Ma KL. The occlusion spectrum for volume classification and visualization. IEEE Trans Vis Comput Graph 2009;15(6):1465–72. http://dx.doi.org/10.1109/TVCG.2009.189.

[9] Correa CD, Ma KL. Visibility histograms and visibility-driven transfer functions. IEEE Trans Vis Comput Graph 2011;17(2):192–204. http://dx.doi.org/10.1109/TVCG.2010.35.

[10] Kniss J, Kindlmann G, Hansen C. Multidimensional transfer functions for interactive volume rendering. IEEE Trans Vis Comput Graph 2002;8(3):270–85. http://dx.doi.org/10.1109/TVCG.2002.1021579.

[11] Park S., Bajaj C. Feature selection of 3d volume data through multi-dimensional transfer functions. Pattern Recognit Lett 2007;28(3):367–74 (Advances in Visual information Processing: Special Issue of Pattern Recognition Letters on Advances in Visual Information Processing. (ICVGIP 2004)). http://dx.doi.org/10.1016/j.patrec.2006.04.008.

[12] Pinto F.d.M., Freitas C.M.D.S. Design of multi-dimensional transfer functions using dimensional reduction. In: IEEE-VGTC symposium on visualization. The Eurographics Association; 2007, p. 131–8. ISBN 978-3-905673-45-6. http://dx.doi.org/10.2312/VisSym/EuroVis07/131-138.
[13] Wu Y, Qu H. Interactive transfer function design based on editing direct volume rendered images. IEEE Trans Vis Comput Graph 2007;13(5):1027–40. http://dx.doi.org/10.1109/TVCG.2007.1051.
[14] Guo H, Mao N, Yuan X. Wysiwyg (what you see is what you get) volume visualization. IEEE Trans Vis Comput Graph 2011;17(12):2106–14. http://dx.doi.org/10.1109/TVCG.2011.261.
[15] Guo H, Li W, Yuan X. Transfer function map. In: 2014 IEEE Pacific visualization symposium, 2014, p. 262–6. http://dx.doi.org/10.1109/PacificVis.2014.24.
[16] Tzeng FY, Lum E, Ma KL. An intelligent system approach to higher-dimensional classification of volume data. IEEE Trans Vis Comput Graph 2005;11(3):273–84. http://dx.doi.org/10.1109/TVCG.2005.38.
[17] Maciejewski R, Woo I, Chen W, Ebert D. Structuring feature space: a non-parametric method for volumetric transfer function generation. IEEE Trans Vis Comput Graph 2009;15(6):1473–80. http://dx.doi.org/10.1109/TVCG.2009.185.
[18] Lindholm S, Jonsson D, Hansen C, Ynnerman A. Boundary aware reconstruction of scalar fields. IEEE Trans Vis Comput Graph 2014;20(12):2447–55. http://dx.doi.org/10.1109/TVCG.2014.2346351.
[19] Shen E, Xia J, Cheng Z, Martin RR, Wang Y, Li S. Model-driven multicomponent volume exploration. Vis Comput 2015;31(4):441–54. http://dx.doi.org/10.1007/s00371-014-0940-7.
[20] Ip CY, Varshney A, JaJa J. Hierarchical exploration of volumes using multilevel segmentation of the intensity-gradient histograms. IEEE Trans Vis Comput Graph 2012;18(12):2355–63. http://dx.doi.org/10.1109/TVCG.2012.231.
[21] Fujishiro I, Azuma T, Takeshima Y. Automating transfer function design for comprehensible volume rendering based on 3d field topology analysis. In: Visualization '99. Proceedings, 1999. p. 467–563. http://dx.doi.org/10.1109/VISUAL.1999.809932.
[22] Wang Y, Zhang J, Lehmann DJ, Theisel H, Chi X. Automating transfer function design with valley cell-based clustering of 2d density plots. Comp Graph Forum 2012;31(3pt4):1295–304. http://dx.doi.org/10.1111/j.1467-8659.2012.03122.x.
[23] Smale S. On gradient dynamical systems. Ann Math 1961;74(1):199–206.
[24] Bremer PT, Hamann B, Edelsbrunner H, Pascucci V. A topological hierarchy for functions on triangulated surfaces. IEEE Trans Vis Comput Graph 2004;10(4):385–96. http://dx.doi.org/10.1109/TVCG.2004.3.
[25] Edelsbrunner H, Harer J, Natarajan V, Pascucci V. Morse–Smale complexes for piecewise linear 3-manifolds. In: Proceedings of the nineteenth annual symposium on computational geometry. SCG '03. New York, NY, USA: ACM; 2003, p. 361–70. ISBN 1-58113-663-3. http://dx.doi.org/10.1145/777792.777846.
[26] Felzenszwalb PF, Huttenlocher DP. Efficient graph-based image segmentation. Int J Comput Vision 2004;59(2):167–81. http://dx.doi.org/10.1023/B:VISI.0000022288.19776.77.