Vision, Modeling, and Visualization (2011)
Peter Eisert, Konrad Polthier, and Joachim Hornegger (Eds.)

A Cluster Hierarchy-based Volume Rendering Approach for Interactive Visual Exploration of Multi-variate Volume Data

Petar Dobrev[1], Tran Van Long[2], and Lars Linsen[1]

[1] Jacobs University, Bremen, Germany
[2] University of Transport and Communication, Hanoi, Vietnam

Abstract

Interactive visual analysis of volumetric data relies on intuitive, yet powerful mechanisms to generate transfer functions. For multi-variate data, traditional methods for interactive transfer functions generation are of limited use. We propose a novel approach, where the user operates in a cluster space. It relies on hierarchical density-based clustering of the high-dimensional feature space. The cluster tree visualization in form of a 2D radial layout serves as an interaction widget for selecting clusters, assigning material properties, changing sizes, merging, and splitting. This widget is complemented by a linked parallel coordinates widget. The interactive selections are automatically mapped to a transfer function for a linked 3D texture-based direct volume rendering, where brushing in parallel coordinates leads to the generation of a 3D binary opacity mask that is overlaid with the opacity values obtained from cluster tree selections. In GPU memory, we only hold the density values from the clustering approach and the cluster IDs. The derived density field allows us to interactively change the size of clusters and to compute normals for lighting. We applied our methods to the visualization of multi-variate data consisting of multiple scalar fields as well as derived scalar property fields from single scalar and vector fields. Our approach scales well to arbitrarily high dimensionality as the complexity of the main user interactions do not increase with the number of dimensions.

1. Introduction

When visualizing spatial data using volume rendering a transfer function is employed to map the underlying data properties to material properties such as color and opacity. The quality of the visualization directly depends on the choice of the transfer function. Hence, it is important that the applied transfer function is able to extract salient features of the dataset that provide the viewer with a good understanding of the visualized data. Typically, in direct volume rendering applications the user is provided with either a set of pre-defined transfer functions with several controllable arguments or a tool to generate transfer functions. Such tools operate in the space of the properties of the visualized data and allow the user to assign different colors and opacities to regions in the space. This makes it very difficult to select regions of interest in a multi-variate dataset, where the space of properties is multi-dimensional.

In this paper, we propose a novel, intuitive, yet powerful volume rendering approach for interactive visual exploration of multi-variate volume data. Instead of interacting in the multi-dimensional feature space (e.g., with histograms), our approach builds on the idea of operating in cluster space. It relies on a hierarchical density-based clustering of the underlying data, see Section 3. The hierarchical cluster tree is presented in a 2D radial layout, which serves as the main widget of our interface to interact with the clusters, see Section 4. Assigned material properties for clusters are directly mapped to a transfer function, which is applied to render the multi-variate volume data in a linked 3D texture-based volume rendering view. Moreover, since the clustering is density based, the size of the clusters can be regulated. As a result, we obtain a smooth transition between clusters along the paths of the hierarchical cluster tree. Furthermore, the user can also split and merge clusters. The splitting of clusters is facilitated by a linked parallel coordinates widget, see Section 5. This widget shows a parallel coordinates representation of the selected clusters in the tree. It also allows for selection of data ranges in feature space. Selections in the parallel coordinates widget also lead to an instantaneous

© The Eurographics Association 2011.

DOI: 10.2312/PE/VMV/VMV11/137-144

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

update of the volume rendering. This is facilitated by a binning and indexing strategy of the parallel coordinates plot.

For the volume rendering part, we apply a 3D texture-based volume rendering approach, which assures us highly interactive frame rates, see Section 6. Storing the multi-variate data set in the texture memory of the GPU is not possible for reasonably large data sets. Relying on density-based clustering, we only store density values and cluster IDs in the GPU memory. The density values allow for interpolation and for interactively changing cluster sizes. Moreover, we can use them to compute appropriate normals for a proper lighting.

We show that our approach can be used to intuitively extract meaningful features from multi-variate volume data, see Section 7. The linked coordinates validate the generation of the clusters and provide another component to the comprehension of the data. Few intuitive operations suffice to create the desired visualizations. Bad clustering results can be fixed. Furthermore, the provided interaction mechanisms allow the user to bring in his/her domain expertise. As we are operating in cluster space, the visual representation and interactions used for visual data exploration scale well to higher number of dimensions.

2. Related Work

One way to visualize multi-variate data is to use direct volume rendering and apply a multi-dimensional transfer function. A lot of research has been put into transfer function design due to the integral role it plays in direct volume rendering. Overviews of the main approaches to design a transfer function can be found in the works by Kindlmann [Kin02] and Pfister et al. [PLB*01] as well as in the survey paper by Kniss et al. [KKH02].

The idea of using multi-dimensional transfer functions to direct volume rendering goes back to Kindlmann and Durkin [KD98]. They proposed using 2D histograms of the value and the magnitudes of first- or second- order directional derivatives of a scalar field to define transfer functions. Kniss et al. [KKH01] extend their work by introducing a set of direct manipulation widgets for multi-dimensional transfer functions. The usability of such approaches, however, diminishes with the number of dimensions of the transfer function, as it becomes more difficult to operate in higher-dimensional histograms. Akiba et al. [AM07, AMCH07] used a representation of the multi-variate data in parallel coordinates to allow the user to generate a transfer function by brushing regions of interest. Their approach works for data with a feature space of dimensionality larger than three, but the amount of necessary user interaction increases with increasing dimensionality and may at some point get cumbersome. Recently, Daniels II et al. [DANS10] presented an approach for interactive vector field exploration by brushing on a scatterplot of derived scalar properties of the vector field. This approach does not scale to more than two properties.

One way to approach visualization of multi-variate data is to render each of the dimensions independently and combine the results [RTF*06, WLM02]. These approaches are not suitable for data of high dimensionality and ignore the correlation between the dimensions. Another approach is to perform dimension reduction. A survey paper by König [Kön98] presents an overview of the main algorithms in the field of multi-variate data reduction. Takanashi et al. [TLMM02] applied Independent Component Analysis (ICA) on a multi-dimensional histogram to classify the volume domain. Classification becomes equivalent to interactive clipping in the ICA space. Tzeng and Ma [TM04] applied ISODATA clustering on the feature space and allowed the user to interact directly with the extracted clusters. User interactions include assigning material properties to clusters and merging, splitting, growing, and shrinking of clusters. Maciejewski et al. [MWCE09] proposed the addition of non-parametric clustering of the feature space to the 2D histogram approach. Thus, the user does not have to rely on a trial-and-error approach when selecting regions of interest. Instead, user interaction is performed on clusters in the 2D histogram. However, their approach is limited only to two-dimensional datasets. Linsen et al. [LLRR08, LLR09, LL09] used a hierarchical density-based clustering technique and visualized the results in a cluster tree and linked parallel coordinates views.

Our approach follows the ideas of using clustering to provide intuitive operations in cluster space. We provide a GPU-based volume rendering approach that automatically maps selections made via an intuitive cluster hierarchy-based user interface to a respective transfer function. It is a novel approach that is most closely related to the work by Akiba et al. [AM07, AMCH07], Tzeng and Ma [TM04], and Linsen et al. [LLRR08, LLR09, LL09]. Like Akiba et al. [AM07, AMCH07] we base our system on an intuitive user interface, but in contrast to their approach we build on a clustering with the benefit that our approach scales better to high dimensionality. Like Tzeng and Ma [TM04] we allow for interaction with clustering results, but in contrast to their approach we use an explicit visualization of the clustering result that operates as an intuitive widget for user interaction. Moreover, we visualize the properties of the clusters by linking them to a parallel coordinates view. Like Linsen et al. [LLRR08, LLR09, LL09] we use a cluster tree visualization, but in contrast to their approach we allow for an interactive modification of the clusters and map the results to a transfer function that is used in a direct volume renderer. As such, we combine the advantages of the three approaches in a novel user interface for intuitive multi-variate volume visualization. The main contributions of this paper can be summarized as follows:

- Coupling cluster analysis with 3D texture-based volume rendering, which allows for truly interactive visual exploration of all present features by using automatic transfer function generation from cluster-space feature selections..

© The Eurographics Association 2011.

- Using a density-based approach to volume rendering of multi-variate data, which allows for the application of GPU-based techniques (including illumination), as we only need to hold the density values and cluster IDs in GPU texture memory.
- Interactive modification of clustering results with the cluster tree widget including merging, splitting, growing, and shrinking of clusters, which includes brushing in a linked parallel coordinates widget and mapping the respective selections to the volume rendering output using 3D binary masks.

3. Hierarchical Density-based Clustering

Our approach relies on a hierarchical density-based clustering approach of the given multi-dimensional feature space. Hence, we have to use an automatic clustering technique that generates hierarchies of nested clusters based on density estimates in feature space. The hierarchical approach allows for the generation of the cluster tree widget, which is our main interaction widget to explore the cluster space, see Section 4. The computed density estimates can be associated to any voxel of our volumetric domain, which allows for interpolation and for lighting computations in the volume renderer, see Section 6.

Any hierarchical density-based clustering approach, such as the ones by Stuetzle and Nugent [SN07], by Ankerst et al. [ABKS99], or by Linsen et al. [LLRR08, LLR09, LL09] could be deployed. We use the latter, which works as follows: Let n be the number of dimensions. First, we compute densities. To do so, an n-dimensional histogram is built with a pre-defined cell (or bin) size. As all n-dimensional cells of the histogram have same size, the density of a cell is proportional to the number of n-dimensional points that fall into it. This number is used as a density estimate that is assigned to each point in that cell. Then, we generate the hierarchy or density clusters. The root cluster (of density zero) contains all points. We generate the first clusters (of density one) from the connected components in the histogram. We iteratively remove the cells with the lowest remaining density values from the histogram. If this causes a connected component in the histogram to get split into more than one connected component, the respective cluster breaks into the respective sub-clusters. The process stops when reaching the maximum density. For more details on the algorithm, we refer the reader to the papers by Linsen et al. [LLRR08, LLR09, LL09].

Once the clustering is complete, we assign to each cluster a unique cluster ID. Clusters are traversed in the order of extraction and the points they contain are marked with the respective cluster index. This way, the points contained in the children nodes overwrite the cluster ID of their parent. Consequently, each inner node in our interface contains all those points of the cluster that are not contained in any sub-cluster. In particular, the root node contains all points that are not part of any other cluster. As the output of the clustering step, each voxel of our multi-dimensional volumetric data set has an assigned cluster ID and an assigned density value.

The clustering approach we used can extract clusters of any dimension and shape. It is a parameter-free approach that does not require any input such as number of clusters or density levels. It can also deal with noise by defining a minimum density threshold that avoids the generation of tiny clusters. However, as with all density-based clustering approaches, it is sensitive to the choice of the bin (or kernel) size. It is not trivial to choose good values for bin size and noise threshold. In Section 7, we show that in case the values have been chosen too conservatively, using the interaction mechanisms provided by our visual exploration approach one can quickly and intuitively transform the given clustering result to a hierarchy of meaningful clusters.

4. Hierarchical Cluster Tree Widget

The hierarchical cluster tree widget (Figure 2a) contains a representation of the cluster hierarchy as a node-link diagram in a radial 2D layout. A node-link diagram was chosen, as the hierarchy is represented explicitly, properties of the nodes can intuitively be mapped to color and shape, and edges can be used as an interaction item. A radial layout was chosen because of its efficient use of screen space. Each node of the tree is assigned a sector of a disc. The node and its children can only be positioned inside this sector. The opening angle of the sector is proportional to the number of descendants a node has. This way we assure that the space is used optimally. One restriction for the opening angle is that it should not be greater than π. This restriction is imposed, so that the sliders that adjust the size of the clusters (described below) can be properly visualized.

Nodes are drawn as small circles in the node-link diagram. The radius of the circle is proportional to the size of the cluster, i.e., the number of voxels it contains. Moreover, initial color and opacity are assigned to each cluster. The opacity is set to an initial value of 0.3 for each node. The initial color is assigned using the color wheel of the HSV color model. Since leaf nodes contain clusters of high density and probably represent interesting features of the dataset, we would like to assign to them discernible colors. For that, we first count the number of leaf nodes n and choose n equally spaced values from the hue spectrum. Then, depending on where each leaf node lies in the radial layout, the closest hue value from the color wheel that has not yet been taken is assigned to the node. This way we assure that we use the whole hue spectrum. For a consistent coloring of the entire node-link diagram, the hue of each inner node is computed as average of the hues of its children. Saturation is determined by the distance from the center of the tree. The value V of the HSV model is always 1. Once a node is assigned a color, it does not change until the user explicitly selects

© The Eurographics Association 2011.

a new color. The circles representing the nodes are drawn with the assigned color. The areas that these circles enclose are drawn in the same color, but with an opacity equal to the opacity assigned to the corresponding cluster.

The interactions that the hierarchical cluster tree allows the user to do are the following: showing/hiding clusters (selected clusters are marked with a red dot), assigning color to clusters (same color is used in all three visualizations), changing opacity of clusters (mouse wheel event), adjusting size of clusters, merging clusters and splitting clusters. The result of any user interaction is immediately mapped to the transfer function and can be observed in the volume renderer window.

Adjusting size of clusters. The sizes of the clusters can be adjusted by changing their density limits. Any voxel that belongs to a cluster, but has a density value that does not fall into the selected density limits is not taken into account during visualization. The density limits of the clusters are adjusted by means of unobtrusive sliders in the hierarchical tree (Figure 2a). When not interacting with the tree the sliders are not visible. They become visible only when the mouse cursor gets close to the respective node. We differentiate between three types of sliders - belonging to nodes of the tree with two or more children, to nodes with one child, and to leaf nodes.

The first type is represented by an arc that intersects the edges connecting the respective inner node to its children. It can be slid between the node and its children in the radial layout. The position of the sliders defines how the density limits of the cluster are adjusted. Linear interpolation on the density range $[d_{min}, d_{max}]$ is used to determine the density d_{slider} the slider corresponds to. The default behavior of the sliders is to take the density limits of the cluster as $[d_{min}, d_{slider}]$, but a simple click changes the behavior to $[d_{slider}, d_{max}]$. The active part of the original density interval of the cluster is shown with bold lines along the edges connecting parent and children nodes. This can be seen in Figure 2a, where the segments of the edges from the parent node to the intersection points with the slider are shown with bold orange lines. Hence, the density interval $[d_{min}, d_{slider}]$ is selected. Only voxels to which densities within this density interval are assigned are shown in the direct volume renderer. The size of the rendered cluster decreases accordingly.

Sliders for inner nodes with one child and leaf nodes have a similar behavior. Since there are less than two children nodes, the sliders are not arcs anymore, but conventional sliders. In the case of inner nodes, the slider is slid along the line that connects it to its child. If the node is a leaf node, the sliding is performed along an axis normal to the outer circle. A density value corresponding to the slider position is determined using linear interpolation and the limits of the cluster are adjusted appropriately.

Merging of clusters. To merge nodes, the user enters the merging mode and selects the nodes to be merged. To ensure that the resulting tree after the merging is still consistent, the interface only allows for merging of nodes that are siblings or in parent-child relationships. All selected nodes merge to the node with lowest depth from the selection. The children of all the selected nodes are assigned to that node. Then, the nodes are removed from the tree and their cluster IDs are overwritten in the spatial distribution of the clusters.

Splitting of clusters / creating a new cluster. For splitting only one node can be selected at a time. The properties of the selected node in feature space is shown in the parallel coordinates widget (described below). The user specifies what part of the cluster should belong to the new node by selecting (brushing) ranges of values for individual dimensions of the feature space in the parallel coordinates. Alternatively or additionally, the density slider of the cluster can be used to set the range of density values of voxels that are to be included in the new node. Immediate feedback is given to the user in the visualization window after each interaction, i.e., each selection is directly reflected in the direct volume renderer. When the user confirms the splitting, whatever is visible in the volume rendering is extracted in a new node. The new cluster is a child of the cluster from which it was created. The voxels of the new cluster are no longer part of the parent cluster.

5. Linked Parallel Coordinates Widget

Plotting the data points of a cluster in parallel coordinates gives the user an understanding of the underlying data properties it corresponds to (Figure 1). Since typically clusters contain a lot of data points, drawing every single one as a polyline in the parallel coordinates plot would cause clutter. To mitigate the problem of over-plotting, we apply a binning technique. Each axis of the parallel coordinates is discretized into a number of equally sized intervals. Then, each data point can be described as a sequence of indices, corresponding to the respective intervals for each dimension. The sequence of interval indices defines a bin. We count the number of data points in each bin and store it in a hash table.

When drawing the parallel coordinates representation of a cluster, only the paths corresponding to non-empty bins are rendered. Furthermore, their opacity is proportional to the number of data points in the bin. When visualizing more than one cluster in parallel coordinates it often happens that the clusters occlude each other. To minimize these effects, we sort them according to the number of bins (or paths) in the discretized parallel coordinates. The ones containing most bins, hence, covering larger area in the plot, are rendered first and the ones with the least number of paths are rendered last. This way a cluster covering a large portion of the parallel coordinates plot will not occlude a very dense cluster.

In addition to visualizing what the clusters correspond to in feature space, the parallel coordinates widget also allows the user to create filters for the volume rendering. The filters

© The Eurographics Association 2011.

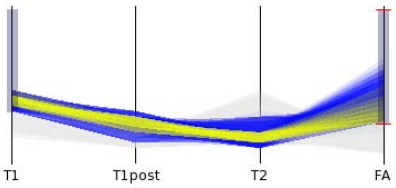


Figure 1: *Brushing in parallel coordinates. Selections (grey vertical bars) along the axes specify what part of the selected clusters is visible. When selections in more than one axis are made, a logical AND operation is performed. The not-selected part of the clusters is rendered in semi-transparent gray for context. Selections can be adjusted using handles (red) that pop up when the cursor is near to the axis.*

are based on selections along the axes of the parallel coordinates (Figure 1). The user can brush what part of the selected clusters should be visible by clicking and dragging the mouse directly on the axes of the parallel coordinates. Selections are depicted by vertical bars. They can subsequently be edited by dragging one of the bar's ends with the mouse, where the handles appear only when the cursor is close to the respective axis. When selections on more than one axis are made, their intersection is computed. The part of the clusters that is not brushed is rendered in semi-transparent gray to provide context, see Figure 1. Feedback is provided to the user in the volume rendering window whenever the selection is changed.

6. Volume Renderer

For visualization of the results of the clustering we employ a 3D texture-based direct volume renderer (Figure 2b). View-aligned slices of the volume are rendered and composited using alpha blending. The renderer supports local illumination. The input to the volume renderer is a 3D grid that only stores a cluster ID and a density value assigned to each voxel. Since these values should not be interpolated between voxels, they are loaded in a 3D texture with nearest neighbor interpolation filters. Using nearest neighbor interpolations, each point in the volume has an assigned cluster ID.

To automatically map the cluster selections to a transfer function, the parameters for each cluster are passed from the user interface to the volume renderer. To each cluster ID the parameters color, opacity, and density limits are assigned. When clusters are merged or split during a session, the respective cluster IDs and density limits are updated. Then, each point gets assigned the respective color of the cluster and the cluster's opacity, if the density value at that point lies inside the density limits. Otherwise, opacity is set to zero. Given the density values, we can interpolate smoothly within the density limits to grow and shrink clusters between zero and full size. Finally, the volume can be rendered with the composited transfer function.

Given the density values, we can also estimate surface normals from their gradients. We pre-compute them and load them in a separate 3D texture with linear interpolation enabled. When lighting is turned on, the loaded surface normals are used in Phong's model.

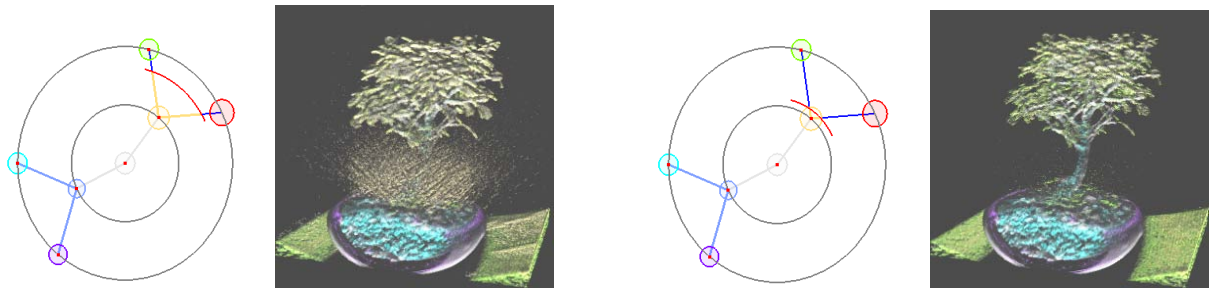
If the user has made selections in the parallel coordinate widget, a 3D mask is created that stores for each voxel a boolean value. Voxels, whose values fall in the selected intervals in the parallel coordinates, are marked with one. The others are marked with zero. The mask is passed along with the spatial distribution of the clusters to the direct volume renderer. Voxels with mask value of zero are rendered fully transparently, i.e., are hidden. This mask can be combined with the transfer function composited above, e.g., for rendering only those voxels that belong to a cluster selected in the cluster tree widget *and* whose values fall in the intervals selected in the parallel coordinates widget.

7. Results and Discussion

Figure 2a-b shows a visualization of the Bonsai tree dataset (Courtesy of S. Roettger, VIS, University of Stuttgart, Germany). using our interface. The scalar field was enhanced to a multi-variate dataset using the magnitudes of first- and second-order derivatives. The cluster hierarchy depicted in the cluster tree widget was obtained by the automatic clustering algorithm. Each cluster is rendered with its default color and opacity, assigned by our algorithm. The red cluster only contains background information and is not visualized. Figure 2c-d shows how the sliders are used to change the size of clusters. The orange-yellow node represents a cluster that mainly includes noise but also part of the Bonsai tree's leaves. By moving the slider of the orange-yellow node, one can reduce the cluster to include only the Bonsai tree's leaves. The accompanying video shows an interactive visual exploration of the dataset without changing the clustering.

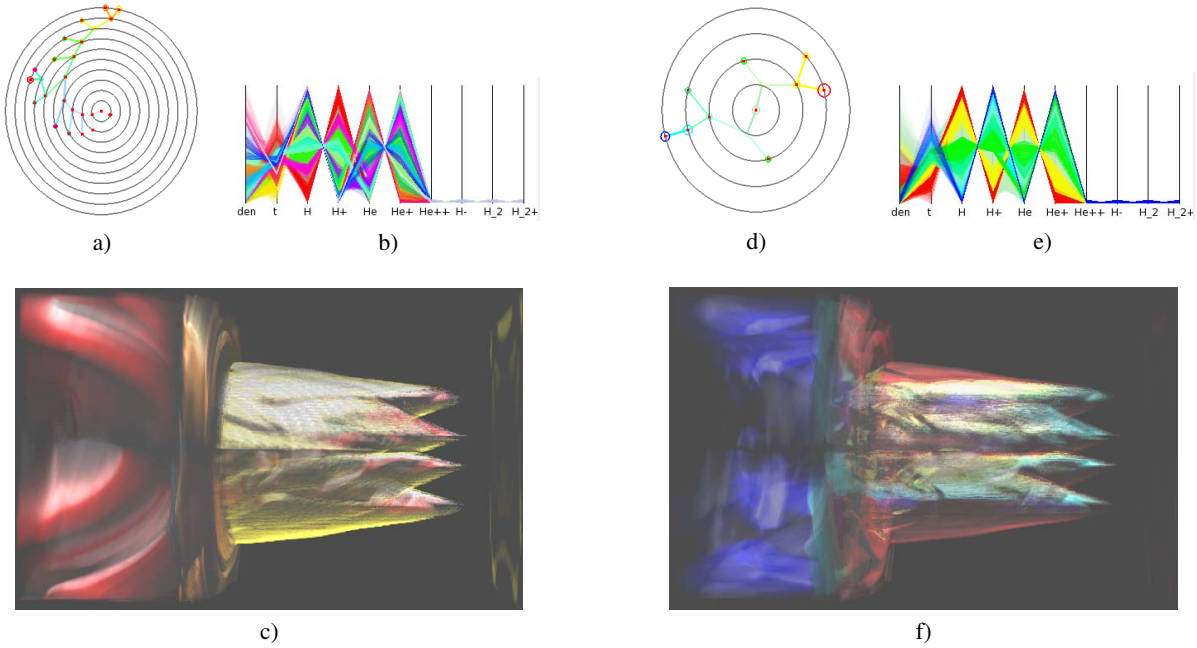
Next, we applied our methods to the multi-variate simulation-based dataset provided in the 2008 IEEE Visualization Design Contest [WN08]. We picked a time slice of this ionization front instability simulation. We considered the ten scalar fields (mass density, temperature, and mass fractions of various chemical elements). What is of interest in this dataset are the different regions of the transition phases between atoms and ions of hydrogen (H) and helium (He). The results of the automatic clustering for this dataset are presented in Figure 3a-c. The cluster hierarchy is rather complicated with many small nodes. This is typically a sign that the bin size for the clustering algorithm have been chosen too conservatively. Instead of starting a tedious trial-and-error process to choose proper bin size, we apply a few fast and intuitive merging steps until the cluster tree exhibits a concise structure (Figure 3d). Using the parallel coordinates representation one can determine which phase belongs to which cluster. The nodes of low depth, which are

© The Eurographics Association 2011.



a) b) c) d)

Figure 2: *Application of the density sliders in the visualization of the Bonsai tree. Moving the slider of the orange-yellow node changes the respective cluster's size. The cluster contains mainly noise (a,b). However, the low-density part of the cluster contains part of the Bonsai tree's leaves that are revealed when moving the slider (c,d).*



a) b) d) e)

c) f)

Figure 3: *Applying our visual exploration approach to a physical simulation dataset with a ten-dimensional feature space (ionization front instability simulation). Hierarchical cluster tree, parallel coordinates representation, and volume rendering of the results of the automatic clustering with default material properties (a-c) and after user interactions to modify the cluster hierarchy and the rendering parameters (d-f).*

placed close to the center of the radial layout, correspond to regions of transition between atoms and ions, i.e., the mass fractions of atoms and ions are comparable. The nodes with high depth, which are placed far from center of the radial layout, correspond to ionized and unionized gas, respectively. To visualize this trend, we applied a color mapping that assigns blue to the cluster of fully ionized gas, red to the cluster of fully unionized gas, and colors in between for the transition clusters. The transition can clearly be observed in the visualization of the clusters' properties using parallel coordinates, see Figure 3e. The direct volume rendering of the selected clusters exhibits the spatial distribution of the transition phases, see Figure 3f. The size of the unionized gas cluster was reduced to lessen occlusion using the density sliders.

In the accompanying video, we also applied our approach to the multi-variate medical datasets provided by the IEEE Visualization Contest 2010 (Courtesy of B. Terwey, Klinikum Mitte, Bremen, Germany). We use MRI images T1, T1 + contrast agent, and T2, plus fractional anisotropy (FA) computed from the DTI dataset on a downsampled ver-

© The Eurographics Association 2011.

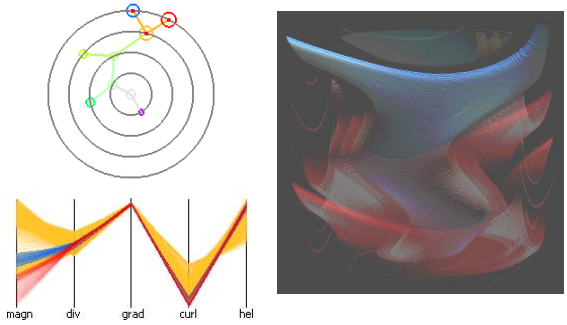


Figure 4: *Using our system to explore a multi-variate field derived from a flow field using vector calculus characteristics. The dataset used is that of a tornado.*

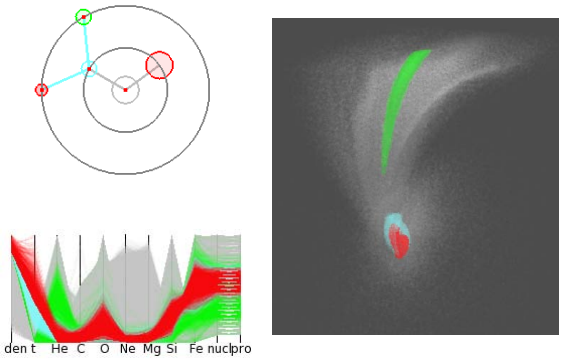


Figure 5: *Using our system to explore a multi-variate dataset without pre-clustering. The feature space is 11-dimensional and comes from a physical simulation of a white dwarf star being torn apart by a black hole. Creating clusters by brushing on two dimensions led to meaningful results after a few interactions.*

sion of size $256 \times 256 \times 176$ to show how interactions via cluster tree widget and parallel coordinates widget can be used to intuitively modify clustering. In particular, the tumor cluster was automatically detected, but could be further improved by brushing in the parallel coordinates widget after investigating its spatial distribution and its properties in parallel coordinates. Other extracted clusters represent the main vessels (red) as well as white matter (orange) and gray matter (green)

Following the ideas by Daniels II et al. [DANS10] and Park et al. [PBL*04], we applied our approach to a multi-variate field of derived properties from a flow field. We derived the five scalar dimensions flow magnitude, divergence, gradient magnitude, curl magnitude, and helicity, as suggested by Park et al. and applied our methods to visually explore respective clusters. Figure 4 shows the result when applied to the tornado data set [CM93] of resolution 256^3.

Although the user interface relies on a hierarchical clustering of the multi-variate data, it can also be used when no clustering is available. In such cases, all data points are put in a single cluster and the parallel coordinates widget is used to create new clusters. We used a time slice of a dataset that comes from an astrophysical simulation of a white dwarf star being torn apart by a black hole. The feature space is 11-dimensional and contains information about mass density, temperature, mass fractions of various elements, and the proton and neutron number at each sample position. We just brushed on the density and temperature dimensions to create meaningful clusters with a few clicks. The accompanying video shows all the user interactions that were performed to retrieve the desired visualization from the not pre-clustered data set. Figure 5 shows the results. The red cluster represents the inner core, the cyan cluster represents the outer core, and the green cluster represents the dense jet. The root node represents lower-density non-background regions.

Although the last example of the preceding section shows that a meaningful clustering can be obtained interactively, this was only true, since brushing on two dimensions sufficed. In general, this would be tedious. Thus, the automatic clustering is extremely helpful to extract features from the high-dimensional feature space. Still, we have seen that interactive cluster modification is desirable. Also, for this example we could not apply illumination, as no density values were available.

Our approach was implemented in C++ using QT, OpenGL, and GLSL. The performance of the direct volume renderer depends on the dataset's resolution, the zoom level and the chosen sampling rate. For the datasets we applied our approach to, we achieved frame rates between 20 and 30 frames per second for a screen resolution of 1680x1050 on an nVIDIA GTX260 graphics card. Assigning material properties to clusters and changing their sizes using the density sliders is instantaneous and takes place in the GPU. Re-clustering operations, such as creating a new node by splitting or merging nodes, are performed on the CPU, but rarely take more than half a second on modern hardware. This allows for a smooth user experience when interacting with the multi-variate data.

Our approach uses post-classification with nearest-neighbor interpolation, which may lead to some aliasing artifacts at the cluster boundaries. However, pre-classification would require us to do classification on the CPU and transmit colors and opacities of all voxels to the GPU. Pre-classification did inhibit true interactive frame rates and the gain in terms of rendering quality was marginal. We also tested the possibility to compute surface normals from the density field on the fly. Again, it slowed down the rendering process significantly such that we decided to always precompute the surface normals. In future work, we plan on testing

© The Eurographics Association 2011.

whether the compression of normal information will enable us load larger volumes into the GPU texture memory without major quality losses.

8. Conclusion

We have presented an intuitive, yet powerful approach for visualization of multi-variate spatial data. It relies on a hierarchical clustering of the multi-variate data. The interface uses two widgets for interaction: a cluster tree widget and a parallel coordinates widget. They allow for interactively exploring the clusters and also for their modification. The result of the user interaction is immediately mapped to a transfer function and the data are visualized using a 3D texture-based direct volume renderer. The derived density field allowed for interpolation when changing cluster size and for normal computations used for illumination. We applied our methods to the visualization of multi-variate data in form of multiple given or derived scalar fields. Since the user operates in cluster space, the proposed approach is suitable for data with a feature space of arbitrarily high dimensionality.

Acknowledgments

The work was supported by the German Research Foundation (DFG, LI 1530/6-1) and the Vietnam National Foundation for Science and Technology Development (NAFOSTED,102.01-2010.09).

References

[ABKS99] ANKERST M., BREUNIG M. M., KRIEGEL H.-P., SANDER J.: OPTICS: ordering points to identify the clustering structure. *SIGMOD Rec. 28*, 2 (1999), 49–60. 3

[AM07] AKIBA H., MA K.-L.: A tri-space visualization interface for analyzing time-varying multivariate volume data. In *EuroVis07 - Eurographics / IEEE VGTC Symposium on Visualization* (May 2007), pp. 115–122. 2

[AMCH07] AKIBA H., MA K.-L., CHEN J. H., HAWKES E. R.: Visualizing multivariate volume data from turbulent combustion simulations. *Computing in Science and Engg. 9*, 2 (2007), 76–83. 2

[CM93] CRAWFIS R. A., MAX N.: Texture splats for 3d vector and scalar field visualization. In *IEEE Conference on Visualization* (1993), Nielson G. M., Bergeron D., (Eds.), IEEE Computer Society Press, pp. 261–266. 7

[DANS10] DANIELS II J., ANDERSON E. W., NONATO L. G., SILVA C. T.: Interactive vector field feature identification. *IEEE Transactions on Visualization and Computer Graphics 16* (2010), 1560–1568. 2, 7

[KD98] KINDLMANN G., DURKIN J. W.: Semi-automatic generation of transfer functions for direct volume rendering. In *IEEE Symposium on Volume Visualization* (1998), pp. 79–86. 2

[Kin02] KINDLMANN G.: Transfer functions in direct volume rendering: Design, interface, interaction. In *SIGGRAPH Course Notes* (2002), pp. 1–6. 2

[KKH01] KNISS J., KINDLMANN G., HANSEN C.: Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *VIS '01: Proceedings of the conference on Visualization '01* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 255–262. 2

[KKH02] KNISS J., KINDLMANN G., HANSEN C.: Multidimensional transfer functions for interactive volume rendering. *Visualization and Computer Graphics, IEEE Transactions on 8*, 3 (Jul-Sep 2002), 270–285. 2

[Kön98] KÖNIG A.: A survey of methods for multivariate data projection, visualisation and interactive analysis. In *5th International Conference on Soft Computing and Information/Intelligent Systems* (Iizuka, Japan, October16–20 1998), Yamakawa T., (Ed.), pp. 55–59. 2

[LL09] LONG T. V., LINSEN L.: MultiClusterTree: Interactive visual exploration of hierarchical clusters in multidimensional multivariate data. *Comput. Graph. Forum 28*, 3 (2009), 823–830. 2, 3

[LLR09] LINSEN L., LONG T. V., ROSENTHAL P.: Linking multi-dimensional feature space cluster visualization to surface extraction from multi-field volume data. *IEEE Computer Graphics and Applications 29*, 3 (2009), 85–89. 2, 3

[LLRR08] LINSEN L., LONG T. V., ROSENTHAL P., ROSSWOG S.: Surface extraction from multi-field particle volume data using multi-dimensional cluster visualization. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1483–1490. 2, 3

[MWCE09] MACIEJEWSKI R., WOO I., CHEN W., EBERT D.: Structuring feature space: A non-parametric method for volumetric transfer function generation. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1473–1480. 2

[PBL*04] PARK S. W., BUDGE B., LINSEN L., HAMANN B., JOY K. I.: Multi-dimensional transfer functions for interactive 3d flow visualization. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference* (2004), PG '04, IEEE Computer Society, pp. 177–185. 7

[PLB*01] PFISTER H., LORENSEN B., BAJAJ C., KINDLMANN G., SCHROEDER W., AVILA L. S., MARTIN K., MACHIRAJU R., LEE J.: The transfer function bake-off. *IEEE Comput. Graph. Appl. 21*, 3 (2001), 16–22. 2

[RTF*06] RÖSSLER F., TEJADA E., FANGMEIER T., ERTL T., KNAUFF M.: GPU-based multi-volume rendering for the visualization of functional brain images. In *SimVis* (2006), pp. 305–318. 2

[SN07] STUETZLE W., NUGENT R.: *A generalized single linkage method for estimating the cluster tree of a density*. Tech. Rep. 514, University of Washington, Department of Statistics, 2007. 3

[TLMM02] TAKANASHI I., LUM E. B., MA K.-L., MURAKI S.: Ispace: Interactive volume data classification techniques using independent component analysis. *Computer Graphics and Applications, Pacific Conference on 0* (2002), 366. 2

[TM04] TZENG F.-Y., MA K.-L.: A cluster-space visual interface for arbitrary dimensional classification of volume data. In *In Proceedings of Joint Eurographics-IEEE TVCG Symposium on Visualization* (May 2004), pp. 17–24. 2

[WLM02] WILSON B., LUM E. B., MA K.-L.: Interactive multivolume visualization. In *ICCS '02: Proceedings of the International Conference on Computational Science-Part II* (London, UK, 2002), Springer-Verlag, pp. 102–110. 2

[WN08] WHALEN D., NORMAN M. L.: Competition data set and description. 2008 IEEE Visualization Design Contest, http://vis.computer.org/VisWeek2008/vis/contests.html, 2008. 5

© The Eurographics Association 2011.