Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2004)
O. Deussen, C. Hansen, D.A. Keim, D. Saupe (Editors)

A Cluster-Space Visual Interface for Arbitrary Dimensional Classification of Volume Data

Fan-Yin Tzeng Kwan-Liu Ma

Department of Computer Science
University of California at Davis, USA

Abstract

In volume visualization, users typically specify transfer functions to classify the data and assign visual attributes to each material class. Higher-dimensional classification makes it easier to differentiate material classes since more data properties are considered. One of the difficulties in using higher-dimensional classification is the absence of appropriate user interfaces. We introduce an intuitive user interface that allows the user to work in the cluster space, which shows the material classes with a set of material widgets, rather than work in the transfer function space. This interface not only provides the user the capability to specify arbitrary-dimensional transfer functions, but also allows the user to operate directly on the classification and visualization results.

1. Introduction

The properties of voxels in a volume data set often have characteristics that form clusters with respect to the different materials in the volume. For this reason, when specifying a 1-D transfer function, a histogram of voxels is often observed since its peaks and valleys can indicate the clusters of different materials in a volume. Similarly, a 2-D joint histogram of voxels' values and gradient magnitudes can be used when specifying 2-D transfer functions to illustrate material clusters in the 2-D domain [KD98]. Such a histogram helps users perform classification since voxels with similar properties are spatially close in the corresponding transfer function space. The utility of displaying the histogram, however, diminishes with higher dimensions since it becomes increasingly difficult to visualize higher dimensional histograms.

Rather than relying on a histogram to perform classification this paper introduces a new visual user interface which displays automatically generated classified results and allows the user to interactively operate on the results to refine the classification. The automatic classification step uses the ISODATA(Iterative Self-Organizing Data Analysis Technique) clustering algorithm. ISODATA takes inputs in arbitrary dimensions and classifies the inputs into classes based on distance in the input data space. In addition to working in arbitrary high dimensions, the user is able to work in a more intuitive cluster/material space rather than the transfer function space. This is very powerful since the user can modify the classes and specify rendering properties such as color and opacity on a per object basis rather than specifying properties based on scalar value and gradient.

2. Previous Work

Much research has been conducted into transfer functions for volume visualization[PLB*01]. He et al.[HHKP96] use genetic algorithm to assist the user in exploring appropriate transfer functions for volume visualization. Marks et al.[MAB*97] create an interface for efficient exploration by replacing the trial and error approach with a space that shows images rendered by all possible transfer functions based on automated analysis. Konig and Groller[KG01] present a user interface paradigm to select transfer functions by a set of specification tools in each property domain.

Multiple dimensional transfer functions take into account more data properties and make the differentiating of materials easier. Levoy[Lev88] first introduced the use of gradient magnitude to enhance the material boundaries in a volume. Kindlmann and Durkin[KD98] suggest looking at the 2-D joint histogram to help specifying the transfer functions. Kniss et al.[KKH01] extend this work by introducing a set of direct manipulation widgets for multi-dimensional transfer functions. They also demonstrate dual domain interaction, where the user can point on a region of a volume in the three dimensional space and immediately see the result in transfer function space. Huang and Ma[HM03] present a technique that can derive a 2-D transfer function by the result of partial region growing from a point selected in the volume space. Tzeng et al.[TLM03] introduce a painting interface for higher-dimensional classification which completely removes the transfer function space and allows the user to work entirely in volume data space. Shamir[Sha03] uses the mean shift procedure to analyze unstructured meshes and presents an algorithm to classify the data based on the feature space. Our technique employs ISODATA to generate the initial classes in a preprocessing step based on arbitrary dimensional inputs, and allows the user to work in an intuitive cluster space for further refinement of the classification.

ISODATA has been used for classifying multidimensional medical data. Panayiotis et al.[MJH*02] compose images obtained by measuring different MRI parameters using ISODATA techniques to identify the ischemic stroke area. Ahmed and Farag[AF96] present a system to segment and label CT brain slices using unsupervised clustering methods. A feature pattern is assigned to each voxel and the unsupervised system segments the 2-D data automatically. Gerig et al.[GMK*92] use both supervised and unsupervised learning algorithms to classify dual-echo volume data sets and study the differences between the learning algorithms. The classified results are displayed using 3-D surface rendering.

3. A Cluster-Space Visual Interface Design

Our work transforms volumetric data into a cluster space representation that can easily be manipulated using per object material property widgets. The initial classes are generated by the ISODATA clustering algorithm, and the whole process is illustrated in Figure 1. To reduce the processing cost of using ISODATA, only a set of randomly chosen voxels are used for classification. The number of chosen voxels depends on the size of a data set, and needs to include a representative distribution of the whole data set in histogram space. In our experience, we found that 1% of the voxels in the data set is typically sufficient for this preclassification step, and is the default value for our system. The user is also able to change the percentage of data used for preclassification.

The output of ISODATA is a set of clusters of voxels with similar characteristics and a set of clusters means. The whole volume is then classified into classes based on the clusters means. The set of material classes is listed on the user interface, and the user can select which materials are shown in the final rendering or can modify the classification using a set of merge and split operations through the cluster space user interface. Figure 2 shows the actual user interface of our system.

ISODATA automatically determines the number of classes in a volume, with the number of material classes adapting according to the characteristics of the data. The classes are rendered with a randomized color and default opacity, and are shown on the interface for the user to choose for applying operations. The operations are shown in the lower left of the interface, and the user can assign visual attributes using the lower middle of the interface.

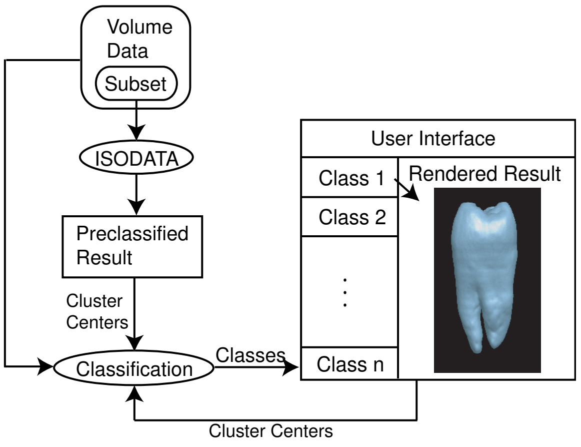

Figure 1: *The cluster-space visualization process. A randomly chosen subset of the data is used as training data to obtain classes of different materials and the cluster centers. The cluster centers are then used to classify the whole volume into classes, and the user can select classes with the interface for rendering. Further refinement of classification can also be performed by operating on the cluster space interface.*

3.1. ISODATA

ISODATA is an unsupervised machine learning algorithm, and is similar to K-means clustering[Mac67], which is the simplest and most fundamental clustering algorithm. K-means can classify multivariate data. In K-means clustering, a user specifies the number of classes (k), and the algorithm arbitrarily selects start positions for the K clusters. Each data point is then assigned to the closest cluster according to the distance of the cluster center and the input data vector. Mean vectors for the classes are then calculated by averaging the values of all the data points in each cluster. The data points are assigned to the new classes using these mean vectors, and this process of reclustering using refined mean vectors is repeated. The final classification result is obtained when the number of iterations reaches a prescribed value or there is not a significant change in the reassignment of data points.

The ISODATA algorithm is an extension of the K-Means algorithm. Unlike K-Means, ISODATA performs splitting, merging, or discarding of clusters according to prescribed criteria after each iteration. Clusters are merged if the num-

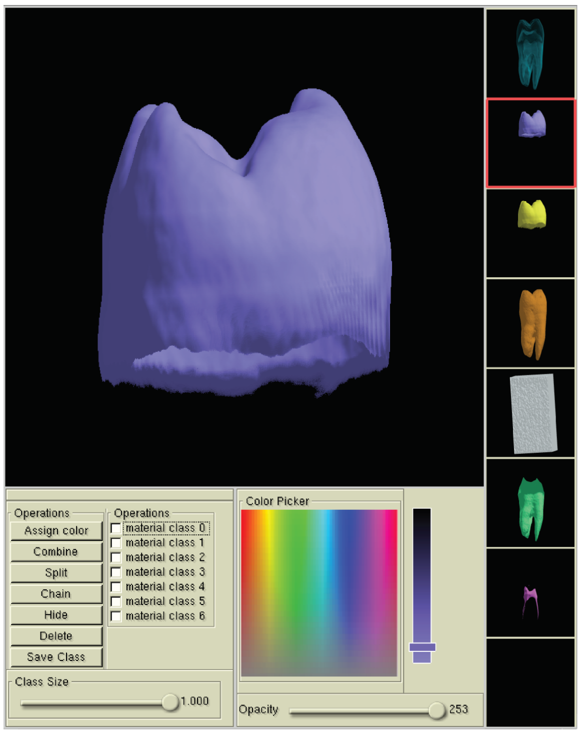

Figure 2: *The cluster-space user interface. The images listed in the right column of the user interface are the classes obtained by the preprocessing step. The user is able to create visualization easily according to the classes by using per object material property widgets in the lower left.*

ber of clusters grows too large or if clusters are too close together. A cluster is split if the number of clusters is too few or if the cluster contains very dissimilar samples. The number of resulting clusters could thus be different from the number initially specified by the users.

Before clustering occurs, the criteria for classification must be determined. A voxel's features, such as its value, gradient magnitude, second directional derivative, and neighbors values, can all be taken into account as the criteria. The most direct way to classify a volume using ISODATA is to apply it to all the volume's voxels. The time complexity of ISODATA clustering is O($nmki$) where n is the number of voxels to be classified, m is the size of the data vector, k is the number of clusters, and i is the number of iterations. In previous works, ISODATA was mainly used in classifying 2-D slices rather than 3-D volumes. Since computational time is highly dependent on the amount of input data, in order to classify entire volume with reasonable computation times, in this work only a randomly selected subset of each volume is fed into the ISODATA routine. It is not necessary to use all voxels since a set of classes and their mean vectors can be approximated by using a subset of those voxels. The result can be used to classify all the remaining voxels according to the distance between the vectors of the voxels features and the mean vectors obtained from those voxels that were classified. Since with a sufficient number of samples, the data distribution of a randomly selected set of voxels is similar to that of an entire volume, a similar classification result is obtained. The threshold for splitting, merging and discarding are set according to the number of samples used for training. For example, using a large amount of data will require a higher threshold for the number of voxels that can be in a single class.

Using ISODATA for the initial classification generally takes about four seconds on a computer with a Pentium 4 2.8GHz CPU, which is acceptable performance for a preprocessing step.

3.2. User Defined Classification

The user interface consists of a set of material widgets for each cluster. The user is able to specify both a color and opacity for a material class and can give a text label to aid in subsequent manipulation. To render multiple materials simultaneously, a user only needs to assign color and opacity values to those classes of interest.

The ISODATA clustering process is a series of merging and splitting based on the numerical analysis. However, detail information and the user's particular interests might not be captured by such a machine learning method. Our interface provides the functionality for the user to modify the classification by applying additional splitting, merging, and changing size to each class according to the visual feedbacks from the system.

In Figure 3, the upper right image shows the classified sample points in a 3-D histogram obtained by ISODATA clustering, and the upper left image is the rendered result of the class shown in light blue in the histogram. The rendered result shows that more than one material are clustered into this class. The user is able to apply splitting to this class, and obtain the result shown in the lower left where only enamel remains. The lower right image is the histogram after splitting.

Figure 4 illustrates the process of merging. The left image shows the rendered result of two classes. Since the two classes are very similar, the users can apply merging to combine the two classes together. The right image is the result of rendering the class after merging.

When classifying an entire volume, every voxel is assigned to the class whose cluster center is closest to that voxel. To better control the classification result, a weight is assigned to each class and expresses its capability to influence the data to be classified[AE83]. A voxel x is classified to class i when the weighted distance of x from the center of

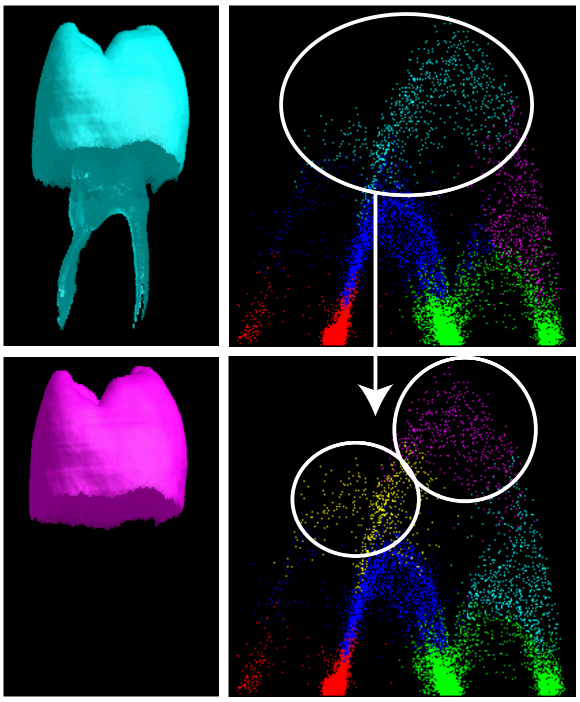

Figure 3: *Splitting a class. The upper row shows the histogram of classes and the rendered image of the class circled in white, which contains two materials. The lower row shows the result after applying a split operation to the class. The class is separated in two and the rendered result of one material class is shown in the lower left.*

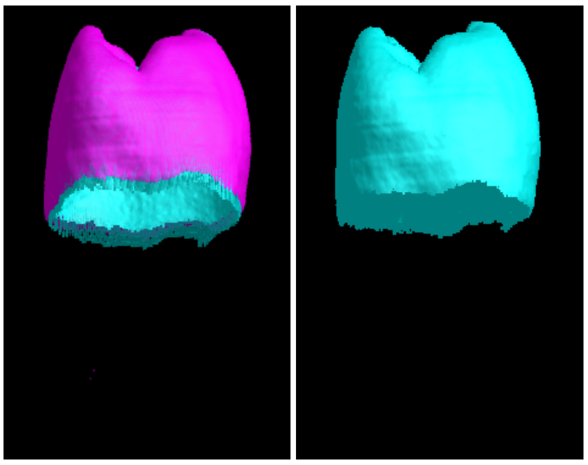

Figure 4: *Merging two classes. The left image shows the rendering of two similar classes and the right image is the result of rendering the merged class.*

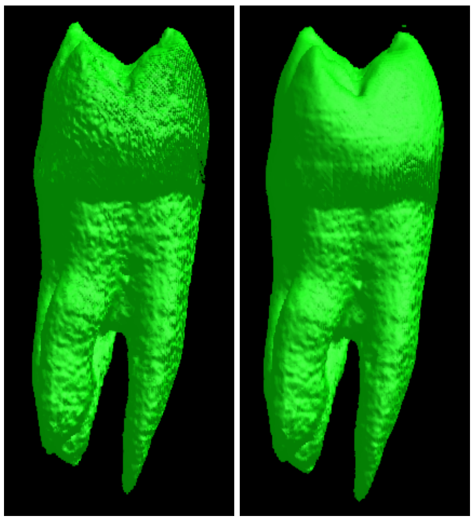

Figure 5: *Modify the size of a class. The user is allowed to change the size of the classes. The left image is the rendered result of a class. The surface is rough and incomplete since part of the material is clustered into other classes. The right image shows the result after enlarging the class. More voxels are included and the classification is more accurate.*

i, which is

$$\frac{\| x - mean_i \|}{weight_i}$$

is minimized. The user can look at one of the class and reduce or enlarge the size of the class by changing the weights. For example, the voxels which are farther from a cluster center and might not belong to that class can be eliminated by reducing the size of that class. Therefore, the user is able to fully control the classified result easily while taking the advantages of using higher dimensional classification.

The left image of Figure 5 is the rendered result of a material class. The surface is rough and uneven since that part of the material is clustered into other classes. To enlarge the class, the user can change the weight with a slider and involve more data points into this class. The right image shows the result after modifying the weight of the class. The surface looks smoother and more complete.

Additional functionalities provided by the user interface include the chaining, hiding and deleting of classes. The chaining operation allows the user to assign the same color and opacity to multiple classes when only changing the visual attributes for one class. Hiding classes keeps the classification results, but removes the class from the user interface and sets the opacity of chosen classes to zero. This lets the user focus on the materials of interest with less informative materials such as the background set to transparent and removed from the interface. Deleting classes removes the chosen classes and releases the voxels into other classes. This deleting operation also makes the classification more flexible and easier to control.

3.3. Rendering

To quickly see the results of the classified volume after modifying the classification such as changing the weights, merging classes and splitting a class, we have implemented a volume renderer in graphics hardware using 3D texture mapping with a fragment program performing classification.

The weights of the classes and the cluster centers are passed as parameters to a fragment program while the data to be classified are stored in textures. RGBA textures are used to store the voxel parameters so that each texture can store four dimensions of the input data. The weighted distance between each input vector and the cluster centers are then calculated in the fragment program to obtain a class id used to look up the corresponding user defined color and opacity. The size of input vectors is bounded by the number of texture units that can be used simultaneously and the size of texture memory. Our current implementation uses up to eight texture units where one is used for storing normals for lighting, another is used for color and opacity map for the classes, and six can be used to store input vectors. Since each texture unit contains four dimensions of the input data, up to 24 dimensions can be used which is sufficient in most cases. Ideally all textures should fit in video memory. If not, the volume must be broken into smaller blocks that are sequentially transferred across the graphics bus for each frame.

We tested our system on an ATI Radeon 9800 Pro graphics card, and achieved 1.5 frames per second rendering to a 512×512 window a $256 \times 256 \times 256$ data set with four dimensional inputs and eight classes. It is desirable to use inputs with number of dimensions and number of classes divisible by four since the vectorized nature of hardware allows for the efficient processing of data in vectors of length four.

A software classification function is also implemented so the user still can utilize this interface when the hardware capability is not sufficient to handle the classification with the dimension user specified. The primary difference between hardware and software classification is that with software classification is performed on the entire data set when the user modifies the classes, and a class id is assigned to each voxel. The class ids are then stored in a texture for looking up color and opacity.

If two adjacent voxels belong to class a and class b respectively, interpolation methods such as linear and trilinear interpolation would make the intermediate values belong to class c, where c is a number between a and b when the intermediate values should belong to either class a or b. To avoid this type of error caused by interpolating class ids, nearest neighbor interpolation method can be used for the texture with class ids, or the linear interpolation needs to be performed in the fragment program after color lookups.

Each software classification process takes several seconds according to the current number of classes. After the classification results are created and stored in a texture, it can be rendered with the looked up color and opacity directly at four frames per second for a $256 \times 256 \times 256$ data set.

3.4. Fuzzy Classification

In some cases, a voxel may belong to more than one class. By default, our system assigns such a voxel to a single class, which could result in incomplete material boundaries. To address such a fuzzy classification problem, our system also allows a voxel to be assigned to multiple classes. Such a voxel is rendered at reduced opacity to give an indication of the level of classication uncertainty.

To handle fuzzy classification, we first find the n closest cluster centers for each voxel and the distances between the voxel and those cluster centers. Both the n closest class ids and scaled distances are stored in hardware textures. The distances are scaled by one over the sum of the n distances to indicate the ratio of distances between the voxel and the neighboring cluster centers, rather than the actual distances.

During rendering, the n closest class ids are read from the textures and their corresponding colors and opacities determined. These colors and opacities are then combined, weighted by the corresponding class distances. The user is given control over the criteria used for combining the different classes, and can specify a threshold to indicate the minimum distance that fuzzy classification is used. Beyond that threshold, attributes are combined using a linear weighting of the class distances.

4. Results

In this section, we present the classification results generated by using the cluster-space user interface. The results were obtained using a CT tooth data set with size $256 \times 256 \times 160$, and a cryosection color brain data set resized to $256 \times 256 \times 256$. In Figure 6, scalar values, gradient magnitudes, and the second derivative are used as the input, and each color on the histogram represents a cluster. Figure 6 shows the results of rendering individual classes. The volume is well classified into the pulp, boundaries, enamel/dentin boundary, tooth/air boundary, enamel, dentin, and regions of the data set that are not part of the tooth. Figure 7 shows visualizations that can be achieved by varying the visual properties of different clusters. The visualizations are obtained by selecting various classes, shown in Figure 6, and assigning a color and opacity to each class. For example, the first image shows semi-transparent dentin with opaque pulp inside, and the enamel covered by the translucent enamel/air boundary which gives an effective overview of all the major structures

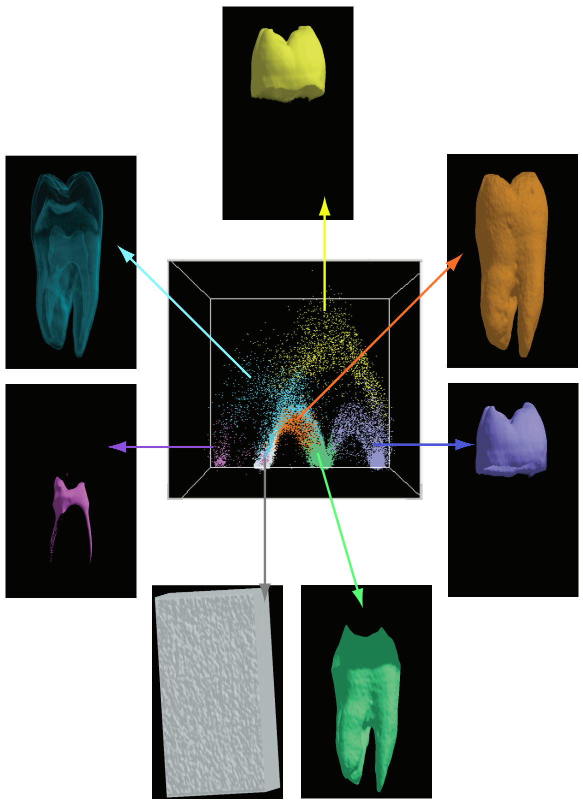

Figure 6: *Volume rendering results for classes of different materials in the tooth data set classified by the ISODATA technique. The histogram in the middle is shown with different colors representing different classes. It illustrates that the classes are well classified according to the data points in the histogram space.*

in the data set. The third image shows opaque dentin with semi-transparent enamel, which more clearly illustrates the shape of the dentin.

The dimensions of the inputs can be any type of data. For example, to classify a color volume data set, the three color channels can be used as the inputs. Takanashi et al. [TLMM02] developed a method for specifying three-dimensional transfer functions that consists of transforming the RGB color values using independent component analysis into a derived ICA space that allows a transfer function to be specified as a series of 1-D transfer functions aligned to each of the ICA axis. Their method simplifies the task of specifying RGB color transfer functions, but requires the user to work in a derived data space that is further from the original data. With our method the brain can be classified easily with the cluster-space interface, avoiding the requirement of specifying transfer functions.

Figure 8 is the result of a four dimensional classification using the cryosection color brain data set. This data set consists of color photographs of slices of a human brain. During acquisition the brain was surrounded by white ice and placed on a table for slicing and photographing. Some regions in the slices are blood vessels, clots, and gaps which reveal the part of the brain belonging to the deeper slices. Three color channels and the luminance value are used as the input vector. The volume is divided into four classes, the brain, dark regions such as the blood and gaps in the data set, the silver table on which each brain slice was placed, and the white ice.

Figure 9 shows the classification process of the same color brain data set. (A) and (B) shows the volume rendered results of two classes with a cutting plane to show the inner structures. The first class consists of the cerebral and white ice surrounding the brain, and the second class is the lobe. (C) and (D) displays the results of dividing the first class in two and modifying the weights using the cluster-based interface. The right image (E) shows the result of rendering the cerebral class and the lobe class together to show the whole brain.

5. Conclusions

Volume rendering will be even more widely employed in routine scientific studies as soon as more intuitive and effective user interfaces become available. The conventional 1-D and 2-D transfer function editors are handy for straightforward volume classification tasks involving only one or two scalar quantities. Higher dimensional volume classification is becoming increasingly important requires a new visual interface through which fast and correct separation of material classes can be possibly made. We have shown that such an interface is possible by utilizing an intelligent system. The concept of cluster-space visualization is very powerful. We have demonstrated how it leads to a much more intuitive and simpler user interface for volume visualization and how it can be done efficiently with hardware acceleration. We believe by allowing the user to switch between the transfer function space and the cluster space, volume classification and visualization tasks that were too sophisticated previously can become manageable. We plan to study the applicability of the cluster-space approach to other types of volume data, in particular higher dimensional volume data.

Acknowledgements

This work has been sponsored in part by the U.S. National Science Foundation under contracts ACI 9983641 (PECASE award), ACI 0222991, and ACI-0325934 (ITR); the U.S. Department of Energy under Memorandum Agreements No. DE-FC02-01ER41202 (SciDAC program) and No. B523578 (the LANL/UC CARE program), and the National Institute of Health through the Human Brain Project. The tooth data set was generated at GE Aircraft Engines, Evendale, Ohio,

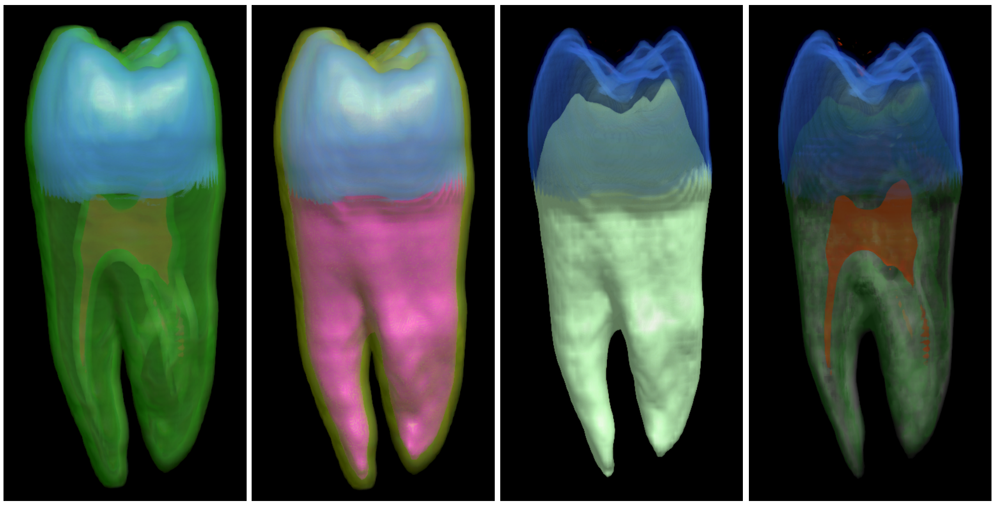

Figure 7: *Volume rendering of multiple materials for the tooth data set.*

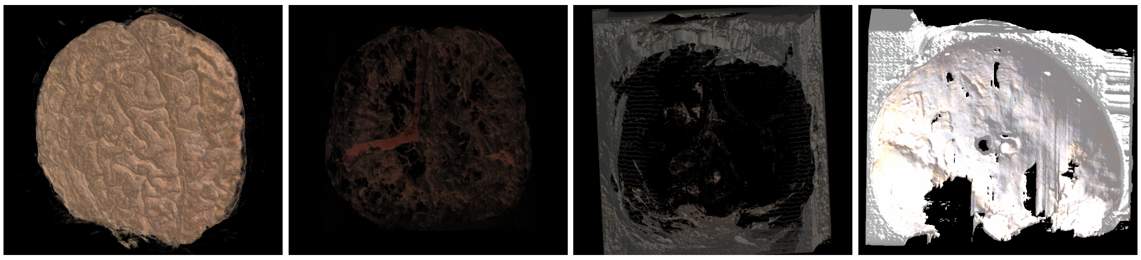

Figure 8: *Four classes of the color brain data set generated by the cluster-space interface are displayed. The four classes are the brain, dark regions such as the blood and gaps of the data set, black materials caused by the table, and the white ice.*

and the cryosection data set was provided by Dr. Arthur Toga of UCLA Brain Research Institute. The authors would also like to thank the members of the UCD visualization and graphics group and the reviewers for their valuable suggestions.

References

[AE83] AURENHAMMER F., EDELSBRUNNER H.: An optimal algorithm for constructing the weighted voronoi diagram in the plane. *Pattern Recognition 17* (1983), 251–257. 3

[AF96] AHMED M., FARAG A.: 3D segmentation and labeling using unsupervised clustering for volumetric measurments on brain CT imaging, 1996. 2

[GMK*92] GERIG G., MARTIN J., KIKINIS R., KÜBLER O., SHENTON M., JOLESZ F. A.: Unsupervised segmentation of 3-d dual-echo MR head data. *image and vision computing, IPMI 1991 special issue 10* (1992), 349–360. 2

[HHKP96] HE T., HONG L., KAUFMAN A., PFISTER H.: Generation of transfer functions with stochastic search techniques. In *Proceedings of IEEE Visualization '96 Conference* (1996), pp. 227–234. 1

[HM03] HUANG R., MA K.-L.: RGVis: Region growing based techniques for volume visualization. In *Proceedings of Pacific Graphics 2003 Conference* (October 2003). 2

[KD98] KINDLMANN G., DURKIN J.: Semi-automatic generation of transfer functions for direct volume rendering. In *IEEE Symposium on Volume Visualization* (1998), pp. 79–86. 1

[KG01] KONIG A., GROLLER E.: Mastering transfer function specification by using VolumePro technology. *In Tosiyasu L. Kunii, editor, Spring Conference on Computer Graphics 2001 17* (April 2001), 279–286. 1

[KKH01] KNISS J., KINDLMANN G., HANSEN C.: Interactive volume rendering using multi-

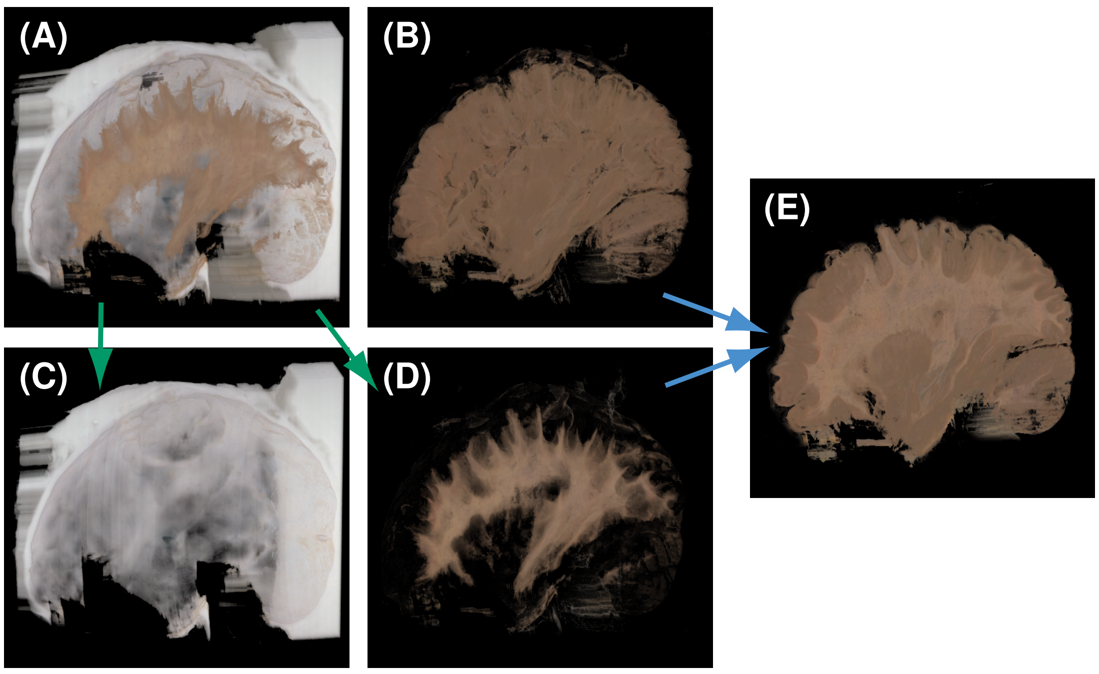

Figure 9: *A classification process of the cryosection color brain data set with part of the brain cut away to show the inner structure. (A) and (B) are two of the classified classes obtained by the initial classification. The first class which consists of white ice and the cerebral is then divided into two by the cluster-space interface and shown in (C) and (D). Finally, the lobe class and the cerebral class are rendered together to create a visualization with the whole brain which is presented in (E).*

dimensional transfer functions and direct manipulation widgets. In *Proceedings of IEEE Visualization '01 Conference* (October 2001), pp. 255–262. 1

[Lev88] LEVOY M.: Display of surfaces from volume data. In *IEEE Computer Graphics and Applications* (May 1988), vol. 8, pp. 29–37. 1

[MAB*97] MARKS J., ANDALMAN B., BEARDSLEY P., FREEMAN W., GIBSON S., HODGINS J., KANG T., MIRTICH B., PFISTER H., RUML W., RYALL K., SEIMS J., SHIEBER S.: Design Galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of SIGGRAPH '97* (1997), pp. 389–400. 1

[Mac67] MACQUEEN J.: Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Symposium on Math, Statistics, and Probability* (1967), pp. 281–297. 2

[MJH*02] MITSIAS P., JACOBS M., HAMMOUD R., PASNOOR M., SANTHAKUMAR S., PAPAMITSAKIS N., SOLTANIAN-ZADEH H., LU M., CHOPP M., PATEL S.: Multiparametric MRI ISODATA ischemic lesion analysis: correlation with the clinical neurological deficit and single-parameter MRI techniques. In *Stroke* (2002). 2

[PLB*01] PFISTER H., LORENSEN B., BAJAJ C., KINDLMANN G., SCHROEDER W., SOBIERAJSKI AVILA L., MARTIN K., MACHIRAJU R., LEE J.: The transfer function bake-off. *IEEE Computer Graphics and Applications 21*, 3 (May/June 2001), 16–22. 1

[Sha03] SHAMIR A.: Feature-space analysis of unstructured meshes. In *Proceedings of IEEE Visualization '03 Conference* (October 2003). 2

[TLM03] TZENG F.-Y., LUM E. B., MA K.-L.: A novel interface for higher-dimensional classification of volume data. In *Proceedings of IEEE Visualization '03 Conference* (October 2003). 2

[TLMM02] TAKANASHI I., LUM E. B., MA K.-L., MURAKI S.: ISpace: Interactive volume data classification techniques using independent component analysis. In *Proceedings of Pacific Graphics 2002 Conference* (2002). 6