Automatic Cross-Sectioning based on Topological Volume Skeletonization

Yuki Mori[1], Shigeo Takahashi[2], Takeo Igarashi[3], Yuriko Takeshima[4], and Issei Fujishiro[4]

[1] Department of Computer Science, The University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8654, JAPAN
yuki@ui.is.s.u-tokyo.ac.jp
[2] Graduate School of Frontier Sciences, The University of Tokyo,
5-1-5 Kashiwanoha, Kashiwa, Chiba 277-8561, JAPAN
takahashis@acm.org
[3] Department of Computer Science, The University of Tokyo / PRESTO, JST
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8654, JAPAN
takeo@acm.org
[4] Institute of Fluid Science, Tohoku University
2-1-1 Katahira, Aoba-ku, Sendai 980-8577, JAPAN
{takeshima, fuji}@vis.ifs.tohoku.ac.jp

Abstract: Cross-sectioning is a popular method for visualizing the complicated inner structures of three-dimensional volume datasets. However, the process is usually manual, meaning that a user must manually specify the cross-section's location using a repeated trial-and-error process. To find the best cross-sections, this method requires that a user is knowledgeable and experienced. This paper proposes a method for automatically generating characteristic cross-sections from a given volume dataset. The application of a volume skeleton tree (VST), which is a graph that delineates the topological structure of a three-dimensional volume, facilitates the automated generation of cross-sections giving good representations of the topological characteristics of a dataset. The feasibility of the proposed method is demonstrated using several examples.

1. Introduction

The visualization of volume datasets is important in many scientific fields, from geophysics to biomedical sciences. Researchers have proposed a number of visualization techniques, including volume rendering and isosurface extraction, but it is still difficult to comprehend the complicated inner structures of volume datasets.

Volume rendering is one of the most commonly used visualization techniques, in which a transfer function (TF) determines how the volume dataset is rendered. The design of the TF, which assigns opacity and color values to each voxel in the data, is crucial and has become a significant research topic. Researchers have proposed many different semi-automatic transfer function design methods, which can be divided into two major categories: image-centric [1, 2] and data-centric [3-6]. An image-centric method presents a broad selection of TFs and generates corresponding visualization

images so that the user can choose the best one [1, 2]. However, these methods do not consider the meaning (*i.e.*, significant features) of target volume datasets. In contrast to image-centric methods, a data-centric method tries to find the best transfer function by rigorously analyzing these features of the input volume datasets [3-6].

Isosurface extraction is another commonly used visualization technique that renders a three-dimensional surface corresponding to points with a single scalar value. Lyness and Blake presented real time isosurface browsing [7], and Itoh and Koyamada presented a method for automatic isosurface propagation [8]. However, both volume rendering and isosurface extraction techniques are still limited in their ability to help researchers visualize detailed structures, because two-dimensional projection of three-dimensional volumetric information always involves some occlusions.

Therefore, users frequently use cross-sections to inspect the inner structures of a volume dataset. Detailed illustrations in biology textbooks and scientific magazines demonstrate the effectiveness of using cross-sections of a volumetric object to explain internal structures. Many researchers have explored ways to enhance cross-sectioning of volume datasets. Owada *et al*. presented an interactive system for designing and browsing volumetric illustrations [9], with which a user can cut a three-dimensional model and see detailed textures of its internal structure in the cross-section. Hinckley *et al*. discussed a two-handed user interface for three-dimensional neurosurgical visualization [10]; their technique involves generating a cutting plane by operating two devices: one corresponding to the target volume; the other corresponding to the cutting plane. Viega *et al*. proposed three-dimensional magic lenses [11]; the lens volume is set interactively, changing the visual appearance of the geometry intersecting the lens volume. Diepstraten *et al*. presented ways to map cutaway renderings directly to modern graphics hardware in order to achieve interactive frame rates [12]. All of these existing systems require users to manually search for an ideal cross-section by fine-tuning many parameters; this is tedious and time-consuming. In addition, users may fail to find a cross-section that appropriately illustrates the volume dataset's inner structures.

To address these problems, we present a method for the automatic generation of cross-sections that reveal characteristic structures of a volume dataset. To extract the characteristic structures of a volume dataset, we use an abstract graph called a volume skeleton tree [13], which represents splitting and merging of isosurfaces with a varying scalar field value. Each node of the graph represents a critical point where merging and splitting happens, and each link represents a region between the isosurfaces represented by its end nodes. Volume skeleton trees have been used to perform important operations in analyzing volume datasets, such as extraction of a critical isosurface or representative isosurface [13] and interval volume decomposition (IVD) [14]. Several authors have proposed methods for extracting similar skeletons from volume datasets [15-17]. While these algorithms are computationally elegant, they do not address changes in isosurface topology (*i.e.*, genera) with respect to the scalar field value, and hence cannot maintain topological consistency of extracted features.

Our method is related to other systems that find characteristic views of target objects. Chakravarty and Freeman [18] presented a method to classify all possible views of an object into a set of characteristic views, within which all views are topologically

identical. Agrawala *et al.*'s system searches for a view that increase the visibility of the parts to create effective assembly instructions [19].

This paper begins with a description of the algorithm for extracting a volume skeleton tree from a volume dataset. Then, Section 3 describes the algorithm for generating cross-sections of a volume dataset based on the extracted VST and cross-section indication techniques. Section 4 is an application of the methodology. Finally, Section 5 discusses the methodology's extensibility and limitations and addresses related issues for future research.

2. The Volume Skeleton Tree

In general, volume data is represented as a three-dimensional single-valued function:

$$w = f(x, y, z) \tag{1}$$

where x, y, and z represent ordinary three-dimensional coordinates and w represents the corresponding scalar field value. A volume is continuous when it is defined as an analytic function, but it usually consists of discrete grid samples. In this case, we apply linear interpolation to construct a continuous volume. Given a continuous volume, isosurfaces can be extracted by collecting points with a constant scalar value. By gradually changing the scalar value, a sequence of isosurfaces with varying topological structure can be determined. A volume skeleton tree (VST) [13] represents such global topological changes in the form of a tree.

A VST node represents a *critical point* where either the number of connected components or the genus of the isosurfaces changes when reducing the scalar value. They are classified into four groups: maxima (C_3), saddles (C_2), saddles (C_1), and minima (C_0), which represent isosurface appearance, merging, splitting, and disappearance, respectively (Figure 1). A VST link represents an isosurface component that is termed *solid* if it expands and *hollow* if it shrinks; solid links are represented as single lines, and hollow links as double lines. The isosurface merging at C_2 and splitting at C_1 has four topological transition paths with different isosurface spatial configurations. For convenience, there is an assumption that all boundary voxels are connected to the virtual minimum with a scalar field value of $-\infty$ [13]. Note that when the node is the virtual minimum, the link incident to a C_0 node is solid, as shown in Figure 1. In this model, the node has coordinates and a scalar field value, while the link has the genus and index of adjacent nodes.

Figure 2(a) shows an isosurface structure of a volume dataset, calculated from the following analytic volume function:

$$\begin{aligned} f(x, y, z) = {} & 4c^2((x-R)^2 + (z-R)^2) \\ & -((x-R)^2 + y^2 + (z-R)^2 + c^2 - d^2)^2 \\ & +4c^2((x+R)^2 + (z+R)^2) \\ & -((x+R)^2 + y^2 + (z+R)^2 + c^2 - d^2)^2 \\ & \text{Where } 0 < d < c,\ c^2 + d^2 \geq 6R^2. \end{aligned} \tag{2}$$

Function (2) has six critical points: P_1, P_2 (appearance), P_3, P_4 (merging), P_5 (splitting), and P_6 (disappearance).

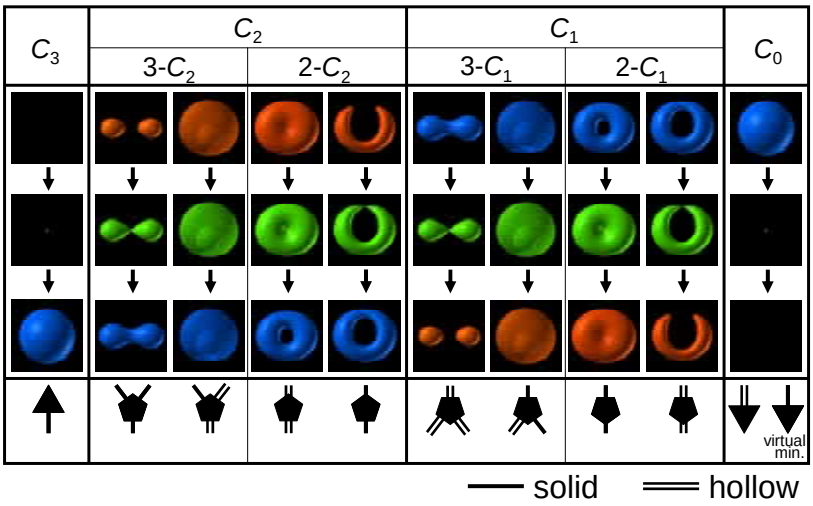


Figure 1. The connectivity of a critical point of each type in a volume skeleton tree: Single and double lines represent links corresponding to solid and hollow isosurfaces, respectively.

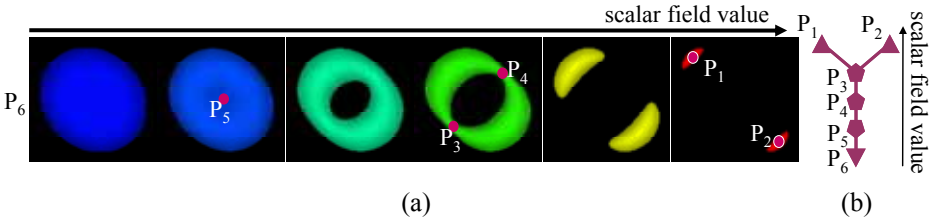


Figure 2. Topological analysis of the analytic volume function (2): (a) A change in isosurface and critical points; (b) The corresponding volume skeleton tree.

As described previously, this algorithm accounts for the virtual minimum; this is artificially added to the volume function (1) so that we can think of an input dataset as a topological three-dimensional sphere S^3. This enables us to check the mathematical consistency of extracted critical points by consulting the Euler formula [11]:

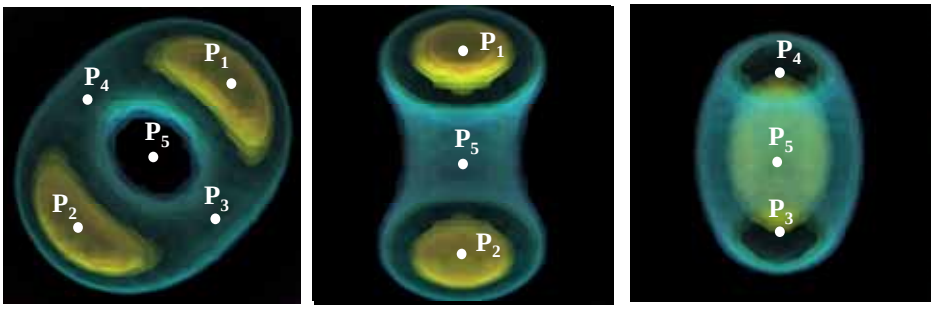
$$\#\{C_3\} - \#\{C_2\} + \#\{C_1\} - \#\{C_0\} = 0 \qquad (3)$$

where $\#\{C_i\}$ represents the number of critical points of index i.

Although the volume skeleton tree effectively captures the topological skeleton of a volume dataset, it is often affected by high-frequency noise that produces a significant number of minor critical points. Therefore, it is necessary to remove minor critical points and simplify the VST in order to exract an important global skeleton from an unknown volume. This process is currently semi-automatic; the system gradually simplifies the tree until the user satisfies the result and stops the process.

3. Generating Cross-sections

Our basic idea is to use a VST to extract characteristic points from a dataset and then cut the dataset with a plane that best fits these target points. For example, we can consider the volume function (2) introduced in the previous section. Our goal is to extract characteristic cross-sections, such as those shown in Figure 3; The dataset has three axes of symmetry. Figure 3 shows three different cross-sections each of which passes through the dataset's center of gravity and two of these axes. This section describes an algorithm to generate such cross-sections using a VST.



(a) (b) (c)

Figure 3. Examples of cross-sections of the analytic function volume (2): (a) A cross-section passing through many critical points; (b), (c) Cross-sections passing through axes of symmetry.

3.1. Computing Cutting Planes using Volume Skeleton Trees

A simple method to generate a cutting plane is 'least-squares' fitting to a group of critical points of *N*. We can construct a covariance matrix (4) of the points:

$$A = \frac{1}{N}\sum_{i}^{N}\begin{pmatrix} (x_i-\bar{x})^2 & (x_i-\bar{x})(y_i-\bar{y}) & (x_i-\bar{x})(z_i-\bar{z}) \\ (x_i-\bar{x})(y_i-\bar{y}) & (y_i-\bar{y})^2 & (y_i-\bar{y})(z_i-\bar{z}) \\ (x_i-\bar{x})(z_i-\bar{z}) & (y_i-\bar{y})(z_i-\bar{z}) & (z_i-\bar{z})^2 \end{pmatrix} \quad (4)$$

where $\bar{x}$, $\bar{y}$ and $\bar{z}$ are averages of *x*, *y* and *z*, respectively. Then, we calculate the eigenvalues and eigenvectors. We can then construct three planes, each of which passes through the center of gravity of the point group and includes any two of the three eigenvectors. By default, the system uses the plane that includes two eigenvectors paired with the two major eigenvalues, but the user can also select two other cross-sections. This selectivity is important because the default cross-section is not always the desired one. We are testing two techniques to obtain target points; one focuses on VST nodes (critical points) and the other focuses on VST links (interval volumes).

The first technique uses critical points as a fitting target; this provides cross-sections to reveal points where the isosurface topology is going to change.

The second technique computes the center of gravity of each interval volume and uses them as a fitting target; weight is assigned to points based on the volume size, and used when fitting the plane.

3.2. Displaying Cross-sections

A cross-section is displayed to the user using one of the following three methods:

The first method converts each of the field values in a cross-section into a suitable color value and displays them on screen. This provides a user with detailed information because the user can see the exact value of each point in the cross-section.

The second method polygonizes isosurfaces associated with critical points and cuts them at the given cutting plane; the system then makes the near side of the plane transparent, making internal structures visible. This method is useful for allowing a user to know the cutting plane's location and to operate on isosurfaces (*e.g.*, hiding an isosurface by clicking on it).

The third method volume-renders the dataset on the far side of the cutting plane. This display method does not show the exact values of the cross-section, however can be expected to provide a significant visual effect to be able to embed 2.5 dimensional information in 2D images.

Figure 3 shows that our method can find appropriate cross-sections of function (2). In this example, we used the third method; Figure 3(a) is rendered with the best-fitting plane. Figure3 (b) and (c) are rendered with the two next-best candidates.

4. Application to Real Datasets

We applied our method to three datasets taken from quantum science and medical science in order to demonstrate its applicability to real datasets. In each dataset, scalar field values were normalized to an 8-bit range [0, 255]. The system runs in an ordinary PC environment (CPU: Pentium IV, 1.6 GHz, RAM: 1 GB).

As we described in the algorithm section, we implemented and tested two techniques for computing a cutting plane: one using critical points and the other using interval volumes. However, the resulting cutting planes were almost identical, so we only provide results obtained using the first technique. In future work, we will investigate the differences in more details.

4.1. Tooth

The first target was a medical CT-scanned dataset, hereafter called 'tooth', which consists of 128×128×80 voxel data with 8-bit scalar values. The original dataset (256×256×161, 16-bit) was used as one of three target datasets in an interesting panel in *Visualization 2000* [20, 21]. The dataset suffers from high-frequency noise, and was downsized into 16×16×10 voxels to generate the volume skeleton tree.

An automatic analysis with default settings oversimplified the VST, so we interrupted the simplification process along the way, producing six critical points. This effectively allowed us to distinguish the tooth's four anatomical regions (cement, enamel, dentin, and pulp) (Figure 4).

Figure 5(a) shows a volume-rendered image of the target dataset using an accentuated transfer function; Figures 5(b), (c), and (d) show cross-sections of the dataset. Figure 5(b) shows a volume-rendered image of the cross-section; Figure 5(c) shows a cross-section obtained by cutting off a polygon; and Figure 5(d) shows a volume-rendered image of the dataset on the far side of the cutting plane.

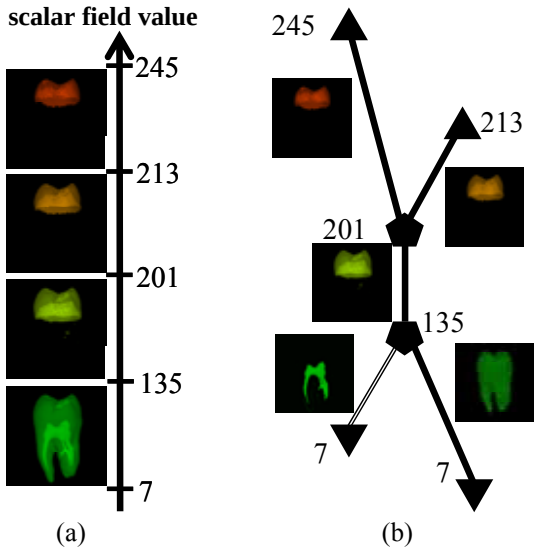


Figure 4. National Library of Medicine (NLM) tooth volume: (a) A change in isosurface and critical points; (b) The corresponding volume skeleton tree.

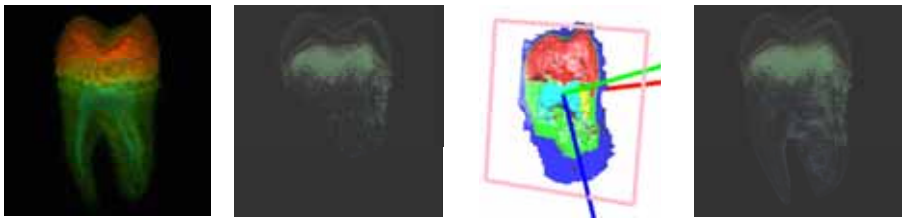
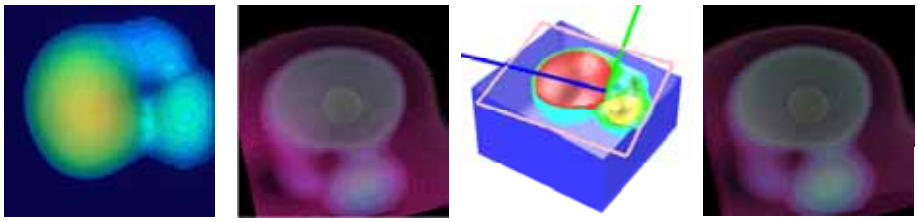


Figure 5. Visualization of the National Library of Medicine (NLM) tooth volume: (a) Volume-rendered image of target datasets using an accentuated transfer function; (b) Volume-rendered image of cross-section; (c) Cutting off a polygon; (d) Volume-rendered image of the dataset on the far side of the cutting plane.

4.2. Antiproton-Hydrogen Atom Collision

The second example is an antiproton-hydrogen atom collision at an intermediate-collision energy in which a single antiproton collide with a single hydrogen atom. Details of the formulation and established numerical schemes can be found in [22].

Figure 6(a) shows a volume-rendered image of the target dataset using an accentuated transfer function; Figures 6(b), (c), and (d) show cross-sections of the dataset. Figure 6(b) shows a volume-rendered image of the cross-section; Figure 6(c) shows a cross-section obtained by cutting off a polygon; and Figure 6(d) shows a volume-rendered image of the dataset on the far side of the cutting plane. By using VST, the system successfully finds the characteristic cross-section that contains a point where the proton collides with the hydrogen atom.



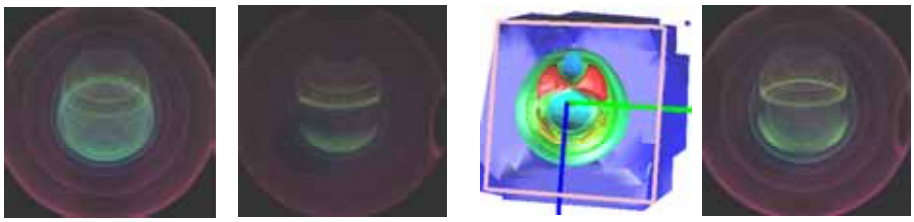
(a) (b) (c) (d)

Figure 6. Visualization of an antiproton-hydrogen atom collision: (a) Volume-rendered image of the target datasets using an accentuated transfer function; (b) Volume-rendered image of the cross-section; (c) Cutting off a polygon; (d) Volume-rendered image of the dataset on the far side of the cutting plane.

4.3. Nucleon

The last target is a 'nucleon' dataset, obtained by simulating the two-body distribution probability of a nucleon in the atomic nucleus of ^{16}O [23]; its resolution here is 41×41×41.

Figure 7(a) shows a volume-rendered image of the target datasets using an accentuated transfer function; Figures 7(b), (c), and (d) show cross-sections of the dataset. Figure 7(b) shows a volume-rendered image of the cross-section; Figure 7(c) shows a cross-section obtained by cutting off a polygon; and Figure 7(d) shows a volume-rendered image of the dataset on the far side of the cutting plane. Figure 7(c) demonstrates that a cross-section actually passes through all the critical isosurfaces.



(a) (b) (c) (d)

Figure 7. Visualization of the nucleon: (a) Volume-rendered image of target datasets using an accentuated transfer function; (b) Volume-rendered image of the cross-section; (c) Cutting off a polygon; (d) Volume-rendered image of the dataset on the far side of the cutting plane.

5. Conclusion and Future Work

This paper proposed a method for automatically generating characteristic cross-sections a given volume dataset of based on topological structure. Furthermore, we have implemented a prototype system to prove the effectiveness of the method and performed an experiment with various datasets. Existing system requires the user's knowledge and experience to find characteristic cross-sections. We automated the process and confirmed that our system can find appropriate cross-sections without tedious manual control.

In the future, we plan to extend our algorithm to generate cross-sections including curved surfaces and multiple planes. We also intend to try other methods, such as medial axes and generalized symmetry for analyzing inner structures, to apply automatic cross-sectioning to other model representations, in particular surface models. Finally, we would like to design better user interfaces for examining volume datasets.

References

1. He, T., Hong, L. Kaufman, A., and Pfister, H.: Generation of Transfer Functions with Stochastic Search Techniques. *Proceedings of IEEE Visualization '96*, (1996), 227-234.
2. Marks, J. *et al*.: Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. *Computer Graphics (Proceedings of Siggraph 97)*, (1997), 389-400.
3. Castro, S., König, A., Löffelmann, H., and Gröller, E.: Transfer Function Specification for the Visualization of Medical Data. *Technical Report* TR-186-2-98-12, Vienna University of Technology, (1998).
4. Fang, S., Biddlecome, T., and Tuceryan, M.: Image-Based Transfer Function Design for Data Exploration. *Proceedings of IEEE Visualization '98*, (1998), 319-326.
5. Kindlman, G., and Durkin, J.W.: Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. *Proceedings of 1998 IEEE Symposium on Volume Visualization*, (1998), 79-86.

6. Fujishiro, I., Azuma, T., and Takeshima, Y.: Automating Transfer Function Design for Comprehensible Volume Rendering Based on 3D Field Topology Analysis. *Proceedings of IEEE Visualization 1999*, (1999), 467-470.
7. Lyness, C., and Blake, E.: Real Time Isosurface Browsing. *Proceedings of the 1st International Conference on Computer Graphics, Virtual Reality and Visualization*, (2001), 143-146.
8. Itoh, T., and Koyamada, K.: Automatic Isosurface Propagation Using an Extrema Graph and Sorted Boundary Cell Lists. *IEEE Transactions on Visualization and Computer Graphics*, **1**(4), (1995), 319-327.
9. Owada, S., Nielsen, F., Okabe, M., and Igarashi, T.: Volumetric Illustration: Designing 3D Models with Internal Textures. *ACM Transactions on Graphics*, **23**(3), (*Proceedings of Siggraph 2004*), (2004), 322-328.
10. Hinckley, K., Pausch, R., Proffitt, D., and Kassell, N.: Two-Handed Virtual Manipulation. *ACM Transactions on Computer-Human Interaction* (*TOCHI*), **5**(3), (1998), 260-302.
11. Viega, J., Conway, M. J., Williams, G., and Pausch, R.: 3D Magic Lenses. *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, (1996), 51-58.
12. Diepstraten, J., Weiskopf, D., and Ertl, T.: Interactive Cutaway Illustrations. *Computer Graphics Forum*, **25**(4), (*Proceedings of Eurographics 2003*), (2003), 523-532.
13. Takahashi, S., Takeshima, Y., and Fujishiro, I.: Topological Volume Skeletonization and its Application to Transfer Function Design. *Graphical Models*, **66**(1), (2004), 22-49.
14. Takahashi, S., Fujishiro, I., and Takeshima, Y.: Interval Volume Decomposer: A Topological Approach to Volume Traversal. to Appear in *Proceedings of SPIE Conference on Visualization and Data Analysis 2005*, (2005).
15. Itoh, T., Yamaguchi, Y., and Koyamada, K.: Fast Isosurface Generation Using the Volume Thinning Algorithm. *IEEE Transactions on Visualization and Computer Graphics*, **7**(1), (2001), 32-46.
16. Bajaj, C.L., Pascucci, V., and Schikore, D.R.: The Contour Spectrum. *Proceedings of IEEE Visualization '97*, (1997), 167-173.
17. Carr, H., Snoeyink, J., and Axen, U.: Computing Contour Trees in All Dimensions. *Computational Geometry*, **24**(2), (2003), 75-94.
18. Chakravarty, I. and Freeman, H.: Characteristic Views as a Basis for Three-Dimensional Object Recognition. *Proceedings of SPIE Conference on Robot Vision*, (1982), 37-45.
19. Agrawala, M., Phan, D., Heiser, J., Haymaker, J., Klingner, J., Hanrahan, P., and Tversky, B.: Designing Effective Step-By-Step Assemble Instructions. *ACM Transactions on Graphics*, **22**(3), (*Proceedings of Siggraph 2003*), (2003), 828-837.
20. Pfister, H., Lorensen, B., Bajaj, C., Kindlmann, G., Schroeder, W., Avila, L. S., Martin, K., Machiraju, R., and Lee, J.: The Transfer Function Bake-off. *Computer Graphics and Applications*, **21**(3), (2001), 16-22.
21. Woo, T.: The National Library of Medicine of the National Institute of Health [http://visual.nlm.nih.gov/data/].
22. Suzuki, R., Sato, H., and Kimura, M.: Antiproton-Hydrogen Atom Collision at Intermediate Energy. *IEEE Computing in Science and Engineering*, **4**(6), (2002), 24-33.
23. Meibner, M.: Web Page [http://www.volvis.org/].