Interval Volume Decomposer: A Topological Approach to Volume Traversal

Shigeo Takahashi[a], Issei Fujishiro[b], and Yuriko Takeshima[c]

[a]The University of Tokyo, 3-8-1 Komaba, Meguro-ku, Tokyo, 153-8902 Japan;
[b]Tohoku University, 2-1-1 Katahira, Aoba-ku, Sendai, 980-8577 Japan;
[c]Japan Atomic Energy Research Institute, 6-9-3 Higashi-Ueno, Taito-ku, Tokyo, 110-0015 Japan

ABSTRACT

The Interval Volume Decomposer (IVD) is an interface for decomposing an entire volume into interval volumes each of which characterizes a distinctive volume feature. The advantage of the IVD is that it allows us to look inside the volume by peeling interval volumes from outside to inside not only interactively but also automatically. This is achieved due to the rigorous analysis of nested structures of the decomposed interval volumes by constructing a level-set graph that delineates isosurface transitions according to the scalar field. A robust algorithm for computing such level-set graphs is introduced in order to extract significant structures in the volume by putting together local interval volumes into a finite number of global groups. Several decomposition examples of medical and simulated datasets are demonstrated so that the present interface effectively traverses the underlying structures of the volume.

Keywords: interval volumes, level-set graphs, simplification, volume peeling, nested structure

1. INTRODUCTION

Generally, we often recognize object shapes through their 2D projections such as photos, pictures, and computer displays. For example, especially in the case of surface shapes, rotating an object allows us to intuitively capture its overall shape. On the other hand, volume datasets have the depth and thus are rather difficult to look through entirely at a time for seeking their inner complicated structures. While several excellent techniques for illuminating inner structures in a volume have been already established, understanding such inner structures is still a time-consuming task because it requires numerous trial and error visualization processes.

Recently, a technique called *volume peeling* has received much attention because it provides us with the means to clarify the inside of the volume by peeling the outer subvolume. While the technique seems to be very promising, it is likely to offer inconsistent decomposition of the volume because the subvolumes to be decomposed are selected by users heuristically. This means that conventional volume decomposition for peeling inevitably depends on the users' trial and error steps, and its automation has been no doubt impossible.

This paper therefore presents an interface called an *Interval Volume Decomposer (IVD)* and its implementation, for exploring the inside of a volume by peeling off outer subvolumes systematically. This interface effectively decomposes an entire volume into a finite number of *interval volumes (IVs)* that reflect its underlying global features, and thus guides users in the actual process of such decomposition in an easy and intuitive way. Since the interface uses IVs as primitives for the decomposition, it can realize a scenario for their automatic outside-to-inside decomposition. Fig. 1 shows selected frames of such a scenario, where a sheep heart[1] is automatically decomposed into feature IVs using our interface (see Section 5.2 for more details). As shown in this figure, our interface removes the outer IVs one by one because it can identify inner and outer IVs with ventricles and muscles of the sheep heart, respectively.

The IV decomposition in this framework is based on a level-set graph of the given volume. The level-set graph delineates topological transitions of isosurfaces according to the scalar field. The paper also explains a

Further author information: (Send correspondence to Shigeo Takahashi.)
Shigeo Takahashi: E-mail: takahashis@acm.org, Telephone: +81 3 5454 6809
Issei Fujishiro: E-mail: fujishiro@fmail.ifs.tohoku.ac.jp, Telephone: +81 22 217 5245
Yuriko Takeshima: E-mail: takesima@koma.jaeri.go.jp, Telephone: +81 3 5246 2528

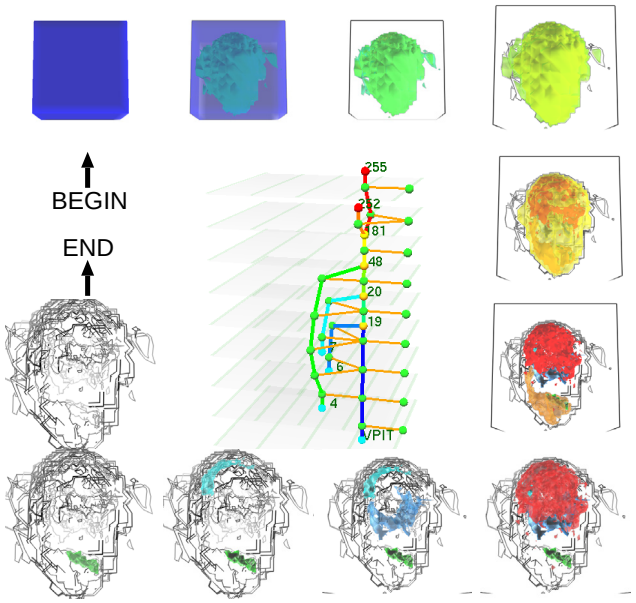

Figure 1. Scenario for decomposing the volume of "sheep heart". The graph at the center shows the corresponding interval volume structure.

robust algorithm for extracting such level-set graphs even when they contain high-frequency noise and degenerate singularities. This algorithm finally produces a relatively small number of IVs by clustering minor IVs together to form significant ones that constitute the volume decomposition reflecting its global features. The level-set graph also allows us to investigate the nested structures of the decomposed IVs, and thus perfectly supports the systematic removal of the IVs from outside to inside.

This paper is organized as follows: Section 2 refers to previous studies relevant to ours. Section 3 explains an algorithm for extracting a level-set graph of a given volume and its decompositions so that the decomposed IVs capture its global features. An algorithm for extracting inclusion relationships between the IVs using the extracted level-set graph is described in Section 4. Section 5 demonstrates our interface for volume decomposition together with its application examples, and Section 6 concludes this paper and refers to future work.

2. RELATED WORK

2.1. Volume Manipulation and Deformation

The smallest units for volume decomposition are undoubtedly voxels. As a pioneering work on such voxel-based editing for volumes, Yamaguchi et al.[2] proposed a method that uses octrees to search 3D space in a volume hierarchically. Due to the recent progress in computer performance, this work has led to an approach called *volume sculpting* where virtual volume objects are designed by direct manipulation. Galyean and Hughes[3] proposed the first volume sculpting system by extending the pixel-based 2D canvas for painting systems to voxel-based 3D clay. Wang and Kaufman[4] augmented the reality of the interface by implementing carving and sawing tools and assigning material attributes such as colors and textures to the volume.

On the other hand, as the virtual reality technologies have been developed, an idea of *haptic rendering* has emerged to define a process of generating forces in response to user interactions with virtual objects through haptic devices.[5] Avila and Sobierajski[6] first introduced this idea to the volume sculpting systems, and Chen and Sun[7] improved the idea to realize a real-time sculpting system that enables a haptic device with many degrees of freedom. Volume peeling can be thought of as one of the editing processes used in the volume sculpting systems.

An anatomical metaphor has also inspired several methods especially for dissecting medical volume datasets obtained by CT and MRI scans. For example, Lorensen[8] extended the traditional texture thresholding technique

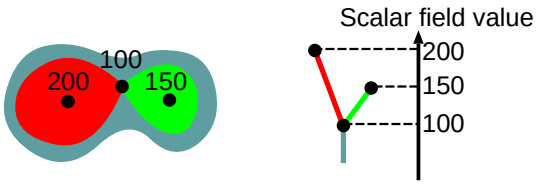

Figure 2. Isosurface transitions (on the left) and the corresponding level-set graph (on the right). The nodes of the level-set graph represent critical points in a volume, and are arranged according to their scalar field values. The same color is assigned to a link of the level-set graph (on the right) and its corresponding subvolume (on the left).

to multi-dimensions to yield a technique called *boolean texture*, which allows us to accentuate a specific set of features of interest from complicated nested geometry while concealing the others. He devised a variety of texture clipping tools, including telescoping and simulated surgery. LaMar et al.[9] generated effects of a magnification lens so that we can observe the inside of a volume through holes on the boundary. McGuffin et al.[10] implemented a set of more general deformation operations, which allow us to observe the interior parts in connection with their surroundings in the volume browsing.

Apart from these software-based approaches, hardware-assisted volume deformation techniques have also been proposed. Kurzion and Yagel[11] implemented fast volume deformation by deflecting cast rays in volume rendering. Wiskopf et al.[12] devised a fast clipping method by exploiting per-fragment operations on the graphics hardware in texture-based volume rendering. More recently, Nagy and Klein[13] presented the concept of volumetric *depth-peeling*, where they sorted the intersections between an isosurface and a cast ray according to the depth, and controlled the transparency of the intersections from front to back to generate the effects of volume peeling.

However, these volume decompositions cannot represent the systematic analysis of the volume features because they strongly depend on users' preference and have no theoretical justifications. Conversely, the present framework offers more systematic decompositions of the volumes because it employs well-defined IVs as decomposition primitives and takes advantage of the level-set graph that outlines the global structure of the volume. The remainder of this section describes related work on level-set graphs and IVs.

2.2. Isosurface Tracking Using Level-Set Graphs

A level-set graph delineates the topological transitions of an isocontour/isosurface according to the scalar field, and serves as a tool for exploring its trajectories in a given dataset. The Reeb graphs[14] and contour trees[15] are among the level-set graphs. Fig. 2 illustrates the transitions of an isosurface in a volume and its corresponding level-set graph, where the nodes of the graph are arranged according to the scalar field from top to bottom. Actually, these nodes locate *critical points* where topological transitions of an isosurface occur as the scalar field value changes. Note also that the same color is assigned to a link of the level-set graph and its corresponding subvolume in the figure.

An optimal algorithm for extracting level-set graphs from volumes is constructed by van Kreveld et al.,[16] and has recently been extended and improved to handle objects of any dimensions by Carr et al.[17] Originally, the level-set graph only pursues the change in the number of connected isosurface components, and cannot detect the change in the topological type (genus) of each connected component. Nonetheless, Pascucci and McLaughlin[18] solved this problem by tracking the Euler number of an isosurface as the scalar field value changes. In addition, Takahashi et al.[19] and Carr et al.[20] presented an algorithm for reducing the complexity of level-set graphs in order to illuminate the global isosurface transitions of the volume.

The level-set graphs were also used to explore the inner structure of the volume in the pioneering work of Bajaj et al.[15] Recently, Carr and Snoeyink[21] presented a more flexible interface that can track any component of an isosurface at an arbitrary scalar field value, together with an improved algorithm for extracting the level-set graphs.

In this way, the level-set graph gives us an important clue as to how an isosurface evolves in a volume. While such isosurface evolution reveals the inner structure of the volume, it still cannot provide an intuitive volume decomposition because the isosurface itself has zero thickness.

2.3. Interval Volumes (IVs)

The IV is formulated by Fujishiro et al.[22,23] and Guo[24] as a generalization of the isosurface, and defined as a subvolume that corresponds to some range of the scalar field value. The subvolumes in Fig. 2 are examples of such IVs. The IV becomes an isosurface when the corresponding range vanishes, and an entire volume when the range covers all the scalar field values. The IVs encourage a quantitative analysis of *region-of-interests* in the volume rather than isosurfaces by taking into account the thickness of each isosurface. A sophisticated tetrahedralization scheme for IVs has also been constructed by Nielson and Sung.[25]

3. INTERVAL VOLUME DECOMPOSITION

This section explains an algorithm for decomposing an given volume into a set of significant IVs by constructing the level-set graph that delineates the corresponding isosurface transitions. In our framework, the decomposition primitive is matched with some link of the level-set graph, and thus is defined to be an IV bounded by end critical isosurfaces (i.e. isosurfaces containing critical points). In practice, our algorithm extracts the level-set graph first by partitioning the given volume into minimal IVs, and then finds a finite number of principal IVs by simplifying the extracted level-set graph.

For constructing the level-set graph, we use an algorithm for topological volume skeletonization proposed by Takahashi et al.[26] This algorithm has been developed by combining the algorithm of Carr et al.[17] for tracking the number of isosurface connected components, the algorithm of Pascucci and McLaughlin[18] for tracking the genus of each isosurface component, and an improved algorithm of Takahashi et al.[19] for IV clustering. Consequently, the topological volume skeletonization algorithm consists of the following seven steps:

1. Adaptive volume tetrahedralization
2. Constructing join and split trees (Fig. 3(a),(b))
3. Tracking the number of isosurface components (Fig. 3(c))
4. Tracking the genus of isosurface components (Fig. 3(d))
5. Constructing the contour tree (Fig. 3(e))
6. Constructing the volume skeleton tree (Fig. 3(f))
7. IV clustering (Fig. 3(g))

Each step will be explained briefly in the remainder of this section. Details are found in the reference.[26]

3.1. Adaptive Volume Tetrahedralization

Before constructing the level-set graph, it is necessary to interpolate between the scalar field values of the given voxels within the 3D volume domain. For this purpose, our algorithm uses adaptive tetrahedralization to efficiently perform a simple linear interpolation.[26] Since this scheme adaptively assigns small tetrahedra to the space where necessary, it can realize low computational complexity and small memory space. Note that we assume that the voxels on the volume boundary share edges with the *virtual minimum*, which has the smallest scalar field value $-\infty$ of all the voxels and is artificially introduced for later use.[19]

3.2. Constructing Join and Split Trees

After the linear interpolation, we use the algorithm of Carr et al.[17] to construct two graphs individually, a *join tree* (JT) that represents isosurface appearance and merging as the scalar field value decreases, and a *split tree* (ST) that represents isosurface disappearance and splitting. Here, both JT and ST contain voxels that participate in the adaptive tetrahedralization as the nodes. For constructing the JT and ST, the algorithm first sorts the list of voxels in descending and ascending orders according to the scalar field respectively, and picks the first voxel from the list to add the graphs by taking into account their connectivities in the tetrahedralization. For example, we can obtain a JT and an ST as shown in Figs. 3(a) and (b), respectively. Note that the value of each node represents its corresponding scalar field value while the letter "V" indicates the virtual minimum.

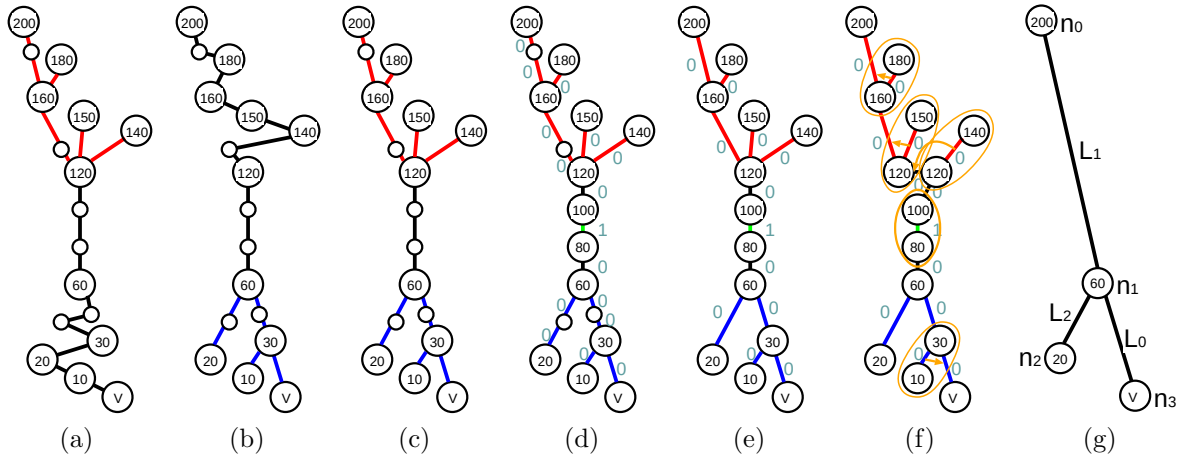

Figure 3. Steps for topological skeletonization algorithm: (a) join tree (JT), (b) split Tree (ST), (c) augmented contour tree (ACT), (d) ACT with genus labels, (e) contour tree (CT) with genus labels, (f) volume skeleton tree (VST), and (g) simplified VST. The value of each node represents its scalar field value, where the node with the "V" is the virtual minimum.

3.3. Tracking the Number of Isosurface Components

The next step is to find the augmented contour tree (ACT) that tracks the change in the number of isosurface connected components. Actually, the ACT is one of the level-set graphs, and is easily obtained from the previous JT and ST. For example, the ACT in Fig. 3(c) is constructed by extracting the isosurface appearance and merging in JT (in red) and isosurface disappearance and splitting in ST (in blue). We are now ready to see that each link of the constructed ACT corresponds to a minimal IV.

3.4. Tracking the Genus of Isosurface Components

As mentioned earlier, the ACT can track the change in the number of isosurface connected components while it cannot detect the change in the topological type (genus) of each connected component, including isosurface transitions from a torus to a sphere and vice versa. We avoid this problem by concomitantly using the algorithm of Pascucci and MaLaughlin,[18] which finds the change in isosurface topological type by tracking the change in the Euler number as the isosurface passes through each voxel. Fig. 3(d) illustrates the resultant ACT where the algorithm successfully extracts the critical points that invoke a topological transition from a sphere to a torus and the reverse transition. Here, a number assigned to each link of the ACT indicates the genus (i.e. the number of torus holes) of the corresponding isosurface component.

3.5. Constructing the Contour Tree

Removing the non-critical nodes from the ACT provides us with the contour tree (CT), which is also one of the level-set graphs. This process actually puts the minimal IVs together into feature IVs, which are now bounded by end critical isosurfaces.

3.6. Constructing the Volume Skeleton Tree

This step is intended to resolve all the critical points of multiple degrees in order to construct the *volume skeleton tree (VST)*.[19] The VST is again one of the level-set graphs, and its node has either of the connectivities shown in Fig. 4. According to the types of non-degenerate topological transitions of isosurfaces, the nodes of the VST can be classified into four groups: C_3(appearance), C_2(merging), C_1(splitting), and C_0(disappearance).[19] With the help of this connectivity rule, our algorithm can obtain the VST by resolving a multiple critical point into single (i.e. non-degenerate) ones as shown in Fig. 3(f).

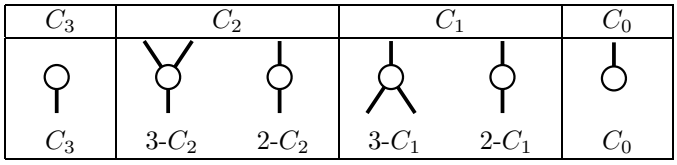

Figure 4. Connectivities of the nodes in the VST[19]: The node represents a critical point and the link represents an IV.

3.7. IV Clustering

The final step is to simplify the VST by removing the minor critical points arising from noise and degeneracy, so as to find the global topological structure of the volume.[19, 26] Indeed, the simplified VST offers the final global IV decomposition for our volume peeling manipulation. In the present framework, our algorithm removes either of the three candidate links C_3–C_2, C_0–C_1, and C_2–C_1 (or C_1–C_2) step by step if the link has the smallest weight. Here, the weight value assigned to each candidate link for removal is defined as

$$\{\text{IV volume}\} \times \{\text{the corresponding interval in the scalar field}\}. \tag{1}$$

This effectively estimates the lifetime of the corresponding isosurface evolution as the scalar field value changes, and enables systematic IV clustering. Fig. 3(f) depicts this process where a link with a yellow circle of the VST will be removed and then implicitly registered with its incident link as indicated by the arrow. Accordingly, the corresponding IV is combined with its neighboring IV by clustering. In this way, the algorithm obtains the final global VST where its links represent the final IV decomposition of the volume (Fig. 3(g)).

4. DETERMINING DECOMPOSITION ORDER OF INTERVAL VOLUMES

In order to peel the decomposed IVs systematically from outside to inside, we need to determine the *IV decomposition order* that takes into account the inclusion relationships between the IVs. The IV inclusion relationships represent spatial configuration where an outer IV completely encloses inner IVs, and thus differ from simple occlusion relationships where front IVs occlude back ones. This section first describes an algorithm for extracting such inclusion relationships automatically from the VST,[27] and then presents how to determine the decomposition order of the IVs for systematic volume peeling.

4.1. Isosurface Inclusion Relationships at Saddles

According to Takahashi et al.,[19] isosurface transitions at saddle critical points of C_2 and C_1 are classified into four cases respectively as shown in the central column of Fig. 5. By following the notation in Shinagawa et al.,[14] we call isosurfaces *solid* if they expand as the scalar field value decreases while *hollow* if they shrink. The leftmost and rightmost columns in Fig. 5 represent subgraphs around the critical points of C_1 and C_2 where a different color is assigned to each link according to whether the corresponding isosurface is solid or hollow. We are now ready to see from Fig. 5 that the IV inclusions occur only in the transition paths of the row (B). In fact, this classification table will help us find systematic decomposition order of IVs in the following process.

4.2. Determining the IV decomposition order by tracing the VST

In the present algorithm, the IV decomposition order is extracted by tracing the links of the VST. For example, suppose that we are going to extract the decomposition order from the VST shown in Fig. 3(g). As shown in Fig. 6(a), our tracing process starts from the virtual minimum n_3 to identify all the links with solid or hollow IVs, because the link incident to the virtual minimum is proven to be an outermost solid isosurface in our framework. Prior to the process, the algorithm prepares two FIFO queues Q_{solid} and Q_{hollow} for solid and hollow links, respectively, and adds links traversed in the upward direction to Q_{solid} while those traversed in the downward direction to Q_{hollow}. In addition, Q_{order} is introduced to retain the links that represent the IV decomposition order. Initially, Q_{hollow} and Q_{order} are empty while Q_{solid} has the link L_0 as described above (Fig. 6(a)).

Our algorithm first identifies all the solid links reachable from the virtual minimum only through upward traversal. This process begins with deleting the link L_0 from Q_{solid} and adds it to Q_{hollow}, and accordingly

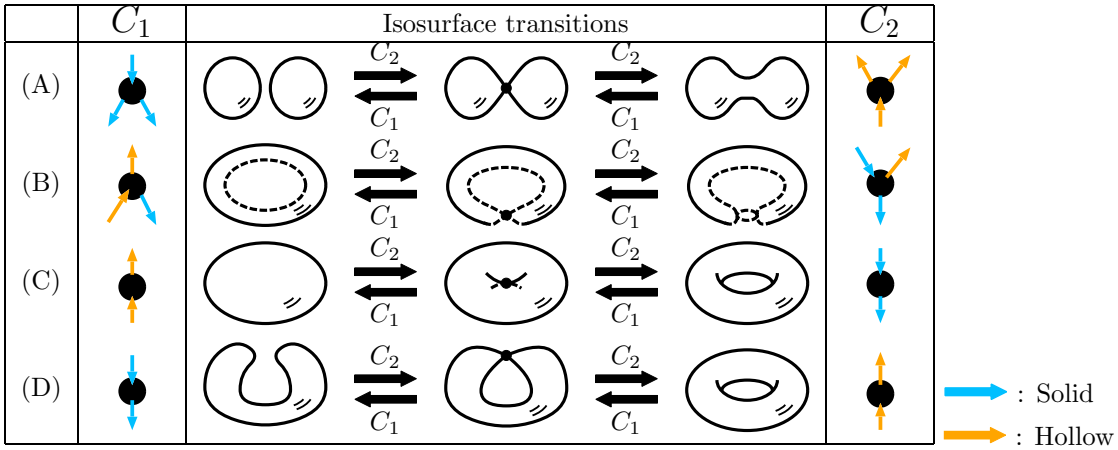

Figure 5. Classification of isosurface transitions at saddle critical points of C_2(splitting) and C_1(merging) according to the spatial configuration in 3D space. The horizontal arrows in the central column indicate isosurface transitions as the scalar field value reduces. Different colors are assigned to solid and hollow links of the VST.

climbing the VST link L_0 from n_3 to the C_1-type critical point n_1 while L_0 is marked "visited." Since n_1 has two downward links one of which is known to be solid, n_1 coincides with the critical point of (B)-C_1 according to the classification of Fig. 5. Note that the arrows in the leftmost and rightmost columns in Fig. 5 indicate the direction of the VST traversals in this process. This lets us insert the solid link L_1 to Q_{solid} while the hollow link L_2 to Q_{hollow} as shown in Fig. 6(b).

The upward tracing process continues to handle the links in Q_{solid} until it becomes empty as shown in Fig. 6(c). After Q_{solid} becomes empty, the algorithm begins to handle the links in Q_{hollow}. In this case, the algorithm resumes the traversal from n_1 to n_2 through L_2 in the downward direction (Fig. 6(d)). Finally, as shown in Fig. 6(e), the tracing process completely fixes the decomposition order of IVs while locating their inclusions by referring to Fig. 5.

The decomposition order of IVs obtained here is justified as follows: As described above, the algorithm first traces the VST from the virtual minimum in the upward direction so that it can add all the reachable solid links to the queue Q_{solid} in the order that they are visited. Since the solid isosurface shrinks as the scalar field value rises, the order of the links coincides with the outside-to-inside IV decomposition order we want. If we have encountered incident hollow links in the previous upward traversal, we trace the VST from the hollow links in the downward direction this time in the order that the hollow links are visited. This downward traversal only visits reachable hollow links that shrink as the scalar field value reduces, which is again consistent with the desired IV decomposition order. In addition, the hollow links in Q_{hollow} represent inner IVs contained in an outer IV that has already been processed. We then perform the second upward traversal of the VST if we have found incident solid links again on the way. This process terminates when all the links of the VST have been handled, and the resultant queue Q_{order} represents the order in which we peel off the corresponding IVs.

Thanks to this algorithm, our interface can retain the IV inclusion relationships automatically from the VST, and stores them as IV inclusion trees as shown in Fig. 7.[28] Here, the IV inclusion trees are represented as planar trees (in yellow) where the nested structures of IVs are explicitly coded. The VST associated with IV inclusion trees are also employed in the present interface, as shown at the centers of Figs. 1, 10, and 11.

5. INTERFACE FOR INTERVAL VOLUME DECOMPOSITION

So far we have seen how to decompose the entire volume into IVs to characterize the global volume features using the topological volume skeletonization. This section describes the details of the Interval Volume Decomposer (IVD), which is an interface for systematically peeling subvolumes from outside to inside. Fig. 8 shows screen images of the IVD where it decomposes a volume dataset having the same VST as in Fig. 3(g). The top window

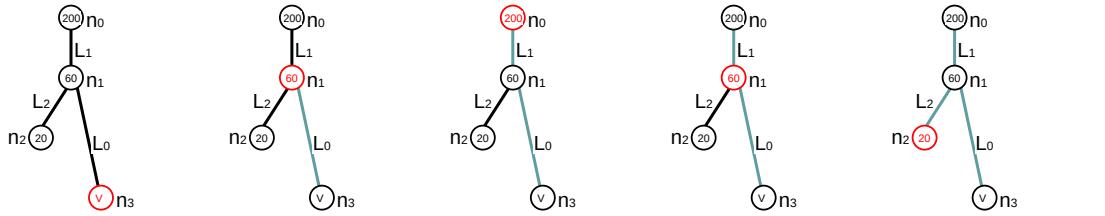

Figure 6. Steps for extracting the IV decomposition order.

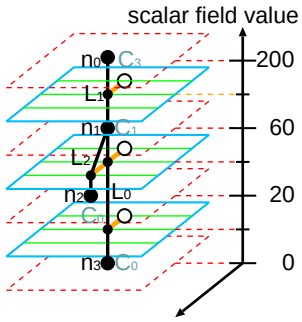

Figure 7. IV inclusion trees (in orange) for the VST shown in Fig. 3(g).

is for displaying the VST together with its IV inclusion trees, as defined in Fig. 7. The bottom window is for presenting the decomposed IVs where a user can peel off outer IVs from the inner ones. The two windows provide an interface for selecting an individual IV using a pointing device as well as changing the viewpoint for each object. In addition, the same color is assigned to a VST link and its corresponding IV in order to exhibit the correspondence between them*. For this purpose, the IVD generates the same number of different colors as that of the decomposed IVs by sampling the hue range from red to blue with uniform intervals.

The remainder of this section is devoted to describing how to accomplish the IV decomposition using the IVD, either interactively (Section 5.1) or automatically (Section 5.2).

5.1. Interactive IV Decomposition

The interactive IV decomposition proceeds by specifying the IV to be excluded using a pointing device such as a mouse. Fig. 8 shows screen images where the outermost IV of the remaining subvolume is interactively removed using the IVD. As shown in Fig. 8(a), a user notifies the interface of the IV to be removed either by clicking the red subvolume in the bottom window or by clicking the red link in the top window. The interface changes the status of the selected IV by reducing the opacity of the corresponding subvolume in the bottom window and emphasizing the corresponding link in the top window, as shown in Fig. 8(b). Note that at this point the user can look at the inner structure of the remaining IVs through the outer translucent one. When the user confirms this selection, the interface finally removes the selected IV and leaves its silhouettes instead in the bottom window, while the corresponding link in the top window is grayed out, as shown in Fig. 8(c). Here, the brightness of the silhouettes is controlled according to the distance from the viewpoint in order to give a depth cue to the user. Of course, the present interface also permits the user to retrieve the eliminated IVs by clicking the corresponding VST link again. Consequently, as demonstrated in Fig. 8, the IVD provides an effective means of eliminating

*Four colors are used for the VST nodes to distinguish between their types: C_3 (in red), C_2 (in yellow), C_1 (in orange), and C_0 (in light blue).

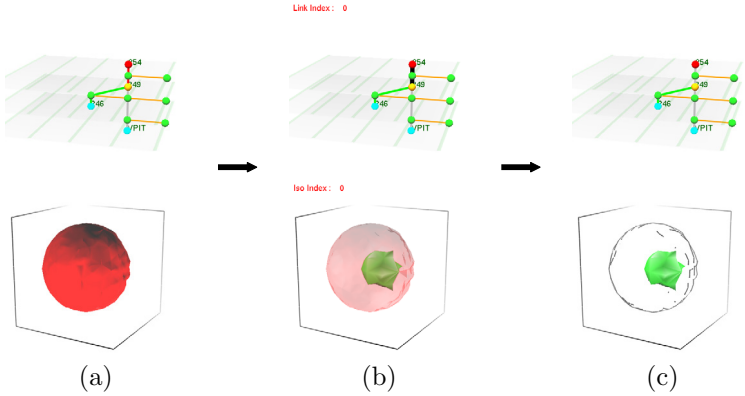

Figure 8. Screen images of the IVD where IVs are decomposed interactively: (a) The outermost IV (in red) is specified by a pointing device. (b) The selected IV becomes translucent (on the right) while its corresponding link is emphasized (on the left). (c) After the selection is confirmed, the IV is removed and its silhouettes are displayed in place (on the right) while the corresponding link is faded (on the left).

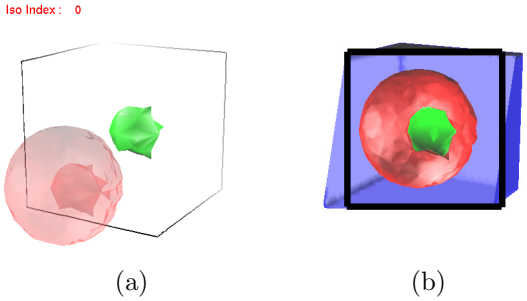

Figure 9. Supplementary operations: (a) An operation for moving the outermost IV, which is controlled by a pointing device. (b) An operation for clipping the remaining subvolume where the clipping plane is indicated by the rectangular frame in black.

decomposed IVs one by one from outside to inside while allowing the user to fully recognize the complicated inner structures of the target volume.

The IVD has additional facilities for exploring the inside of the volume as shown in Fig. 9. For example, the user can move the outermost IV by dragging it with the pointing device in the bottom window in order to confirm the arrangement of the inner IVs as shown in Fig. 9(a). Furthermore, the user can settle and move a clipping plane at any time to examine the cross-sections of the remaining IVs, where the clipping plane is indicated by the rectangular frame (in black) as shown in Fig. 9(b).

5.2. Automatic IV Decomposition

The present scheme allows the user to give a systematic IV decomposition order that reflects the global nested structure of the IVs in the volume. In particular, using the IV decomposition order, the IVD automatically generates a scenario for peeling the IVs from outside to inside and then produces an animation for the corresponding peeling steps.

Figs. 1, 10, and 11 present scenarios for volume peeling using the present IV decompositions. In these scenarios, the decomposed IVs gradually disappear in accordance with the order of the VST links in the extracted list. In each of the corresponding animations, the IVD begins to reduce the opacity of the second IV without waiting for the first IV to fade out completely. This consecutive disappearance of outer IVs makes it possible for users to intuitively understand the complicated inner structures in a volume. Furthermore, the silhouettes

of the IVs will smoothly emerge in the animation because their color gradually becomes conspicuous as the corresponding IVs become transparent.

It is obvious that the space of interest shrinks step by step as outer IVs are removed one by one in the decomposition scenarios. This motivates us to equip the IVD with close-up views of the target volume as the IV decomposition proceeds. Actually, the interface realizes such an automatic close-up views by controlling the distance between the viewpoint and target volume in proportion to the total size of the remaining IVs.

As described in Section 1, Fig. 1 shows a scenario in which the volume of a sheep heart ($177 \times 177 \times 177$)[1] is dissected from outside to inside using the IVD. This example demonstrates that the present interface works well and provides a systematic process of decomposing nested structures even for such an anatomical volume. Fig. 10 is a simulated dataset ($41 \times 41 \times 41$), where the two-body distribution probability of a nucleon in the atomic nucleus ^{16}O is computed.[29] This dataset contains a two-fold nested structure of IVs and reveals its attractive interior through the IVD. Fig. 11 represents another dataset ($129 \times 129 \times 129$), which is obtained by simulating the antiproton-hydrogen atom collision at intermediate collision energy.[30] Note that the IVD successfully resolves the complicated structure of this dataset even though it contains a four-fold nested structure of IVs. These results prove the potential and feasibility of the present framework.

6. CONCLUSION AND FUTURE WORK

This paper has presented an interface called the Interval Volume Decomposer (IVD) for peeling subvolumes from outside to inside in order to illuminate the underlying structures in a given volume. The decomposition primitives used here are interval volumes that reflect a global level-set graph of the volume. The level-set graph delineates the topological isosurface transitions efficiently, and thus provides us with the systematic means of decomposing the entire volume into IVs. Furthermore, with the level-set graph, the interface can detect the nested structures of the IVs and determine the IV decomposition order based on such structures. Scenarios for decomposing IVs from outside to inside can be automatically generated using the present interface.

Future extensions include enhancing volume peeling operations by providing additional effects such as IV cutting and deformations. The present interface can also be extended to support IV-based volume visualization where we have to elaborate methods of controlling the visualization parameters such as transfer functions. It can also be urged that the IVD gives an ideal set of decomposed IVs for any given volume dataset in a topological sense. However, domain-specific knowledge might require a subset of the IVs to be merged or to be subdivided into smaller ones. Therefore, the post-editing functionality should be incorporated into the IVD for its diversified uses, which is also left as a topic for future work.

Acknowledgment The authors thank Prof. Dr. Gregory M. Nielson for fruitful discussions in the early stage of this work. This work has been partially supported by Japan Society of the Promotion of Science under Grants-in-Aid for Young Scientists (B) No. 14780189, and the Ohkawa Foundation.

REFERENCES

1. H. Pfister, et al., "The transfer function bake-off," *IEEE Computer Graphics & Applications* **21**(3), pp. 16–22, 2001.
2. K. Yamaguchi, N. Inamoto, H. S. Kunii, and T. L. Kunii, "Three-dimensional data input by selection of hierarchically defined objects," in *Proc. of Eurographics '84*, pp. 15–23, 1984.
3. T. A. Galyean and J. F. Hughes, "Sculpting: An interactive volumetric modeling technique," in *Computer Graphics (Proc. of Siggraph '91)*, pp. 267–274, 1991.
4. S. Wang and A. E. Kaufman, "Volume sculpting," in *Proc. of the 1995 Symposium on Interactive 3D Graphics*, pp. 151–156, 1995.
5. H. Iwata and H. Noma, "Volume haptization," in *Proc. of IEEE 1993 Symposium on Research Frontiers in Virtual Reality*, pp. 16–23, 1993.
6. R. S. Avila and L. M. Sobierajski, "A haptic interaction method for volume visualization," in *Proc. of IEEE Visualization 1996*, pp. 197–205,485, 1996.

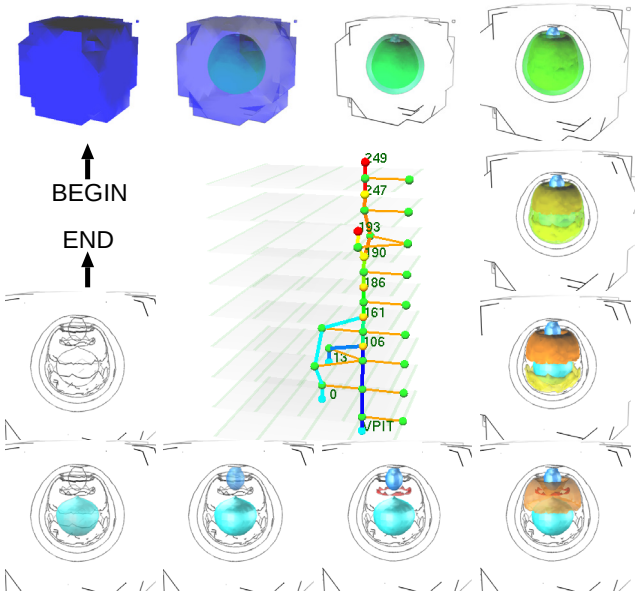

Figure 10. Scenario for decomposing the volume of "nucleon". The image at the center shows the corresponding interval volume structure.

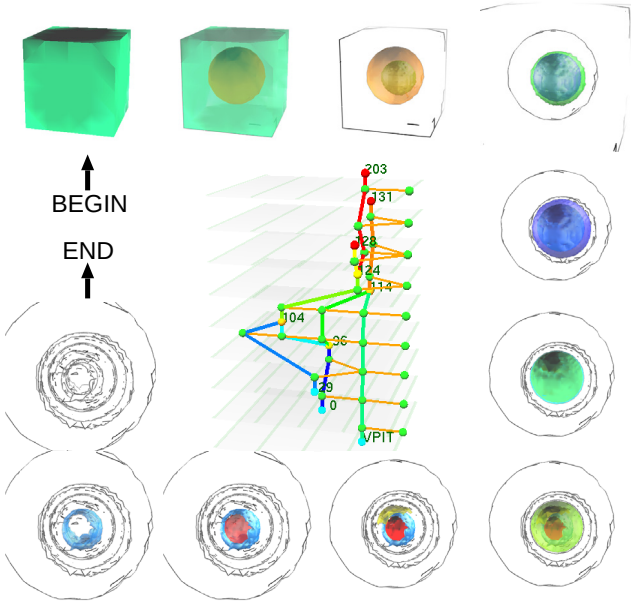

Figure 11. Scenario for decomposing the volume of "antiproton-hydrogen atom collision". The image at the center shows the corresponding interval volume structure.

7. H. Chen and H. Sun, "Real-time haptic sculpting in virtual volume space," in *Proc. of ACM VRST 2002*, pp. 81–88, 2002.
8. W. E. Lorensen, "Geometric clipping using boolean textures," in *Proc. of IEEE Visualization '93*, pp. 268–274, 1993.
9. E. LaMar, B. Hamann, and K. I. Joy, "A magnification lens for interactive volume visualization," in *Proc. of the 9th Pacific Conference on Computer Graphics and Applications (PG2001)*, pp. 223–232, 2001.
10. M. J. McGuffin, L. Tancau, and R. Balakrishnan, "Using deformations for browsing volume data," in *Proc. of IEEE Visualization 2003*, pp. 401–408, 2003.
11. Y. Kurzion and R. Yagel, "Interactive space deformation with hardware-assisted rendering," *IEEE Computer Graphics & Applications* **17**(5), pp. 66–77, 1997.
12. D. Wiskopf, K. Engel, and T. Ertl, "Volume clipping via per-fragment operations in texture-based volume visualization," in *Proc. of IEEE Visualization 2002*, pp. 93–100, 2002.
13. Z. Nagy and R. Klein, "Depth-peeling for texture-based volume rendering," in *Proc. of the 11th Pacific Conference on Computer Graphics and Applications (PG2003)*, pp. 429–433, IEEE Computer Society Press, 2003.
14. Y. Shinagawa, Y. L. Kergosien, and T. L. Kunii, "Surface coding based on morse theory," *IEEE Computer Graphics & Applications* **11**(5), pp. 66–78, 1991.
15. C. L. Bajaj, V. Pascucci, and D. R. Schikore, "The contour spectrum," in *Proc. of IEEE Visualzation '97*, pp. 167–173, 1997.
16. M. van Kreveld, R. van Oostrum, C. Bajaj, V. Pascucci, and D. Schikore, "Contour trees and small seed sets for isosurface traversal," in *Proc. of 13th ACM Symposium on Computational Geometry*, pp. 212–220, 1997.
17. H. Carr, J. Snoeyink, and U. Axen, "Computing contour trees in all dimensions," *Computational Geometry* **24**(2), pp. 75–94, 2003.
18. V. Pascucci and K. Cole-McLaughlin, "Efficient computation of the topology of level sets," in *Proc. of IEEE Visualization 2002*, pp. 187–194, 2002.
19. S. Takahashi, Y. Takeshima, and I. Fujishiro, "Topological volume skeletonization and its application to transfer function design," *Graphical Models* **66**(1), pp. 22–49, 2004.
20. H. Carr, J. Snoeyink, and M. van de Panne, "Simplifying flexible isosurfaces using local geometric measures," in *Proc. of IEEE Visualization 2004*, pp. 497–504, 2004.
21. H. Carr and J. Snoeyink, "Path seeds and flexible isosurfaces using topology for exploratory visualization," in *Proc. of Joint Eurographics-IEEE TCVG Symposium on Visualization*, pp. 49–58, 285, 2003.
22. I. Fujishiro, Y. Maeda, and H. Sato, "Interval volume: A solid fitting technique for volumetric data display and analysis," in *Proc. of IEEE Visualization '95*, pp. 151–158, CP–18, 1995.
23. I. Fujishiro, Y. Maeda, H. Sato, and Y. Takeshima, "Volumetric data exploration using interval volume," *IEEE Transactions on Visualization and Computer Graphics* **2**(2), pp. 144–155, 1996.
24. B. Guo, "Interval set: A volume rendering technique generalizing isosurface extraction," in *Proc. of IEEE Visualization '95*, pp. 3–10, 1995.
25. G. M. Nielson and J. Sung, "Interval volume tetrahedralization," in *Proc. of IEEE Visualization '97*, pp. 221–228, 1997.
26. S. Takahashi, G. M. Nielson, Y. Takeshima, and I. Fujishiro, "Topological volume skeletonization using adaptive tetrahedralization," in *Proc. of Geometric Modeling and Processing 2004*, pp. 227–236, 2004.
27. S. Takahashi, Y. Takeshima, I. Fujishiro, and G. M. Nielson, "Emphasizing isosurface embeddings in direct volume rendering." to appear in the Dagstuhl Visualization 2003 Book.
28. S. Takahashi, Y. Shinagawa, and T. L. Kunii, "A feature-based approach for smooth surfaces," in *Proc. of ACM 4th Symposium on Solid Modeling and Applications*, pp. 97–110, 1997.
29. M. Meißner. Web Page [http://www.volvis.org/].
30. R. Suzuki, H. Sato, and M. Kimura, "Antiproton-hydrogen atom collision at intermediate energy," *IEEE Computing in Science and Engineering* **4**(6), pp. 24–33, 2002.