Computers & Graphics 38 (2014) 212–221

Contents lists available at ScienceDirect

Computers & Graphics

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

Special Section on CAD/Graphics 2013

Efficient boundary surface reconstruction from heterogeneous volumetric data via tri-prism decomposition

Ming Wang, Jie-Qing Feng*, Wei Chen

State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China

ARTICLE INFO

Article history:
Received 5 August 2013
Received in revised form
27 October 2013
Accepted 28 October 2013
Available online 19 November 2013

Keywords:
Heterogeneous object
Boundary representation
Isosurface extraction

ABSTRACT

We propose a novel and efficient approach for extracting the boundary surfaces from heterogeneous volumetric data in one pass. Each homogeneous material component is surrounded by a boundary surface, which is composed of piecewise 2-manifold meshes. The key idea is to subdivide each cubical voxel into two tri-prism voxels and to construct the boundary surfaces in a dimension-ascending (DA) way, *i.e.*, from points to lines and then to faces. The extracted boundary surfaces can fully isolate the homogeneous material components, and the information on intersections between boundary surfaces can be explicitly retrieved. The surface reconstruction process can be accomplished efficiently by adopting a case table. The proposed approach is independent of the number of material types employed. Additionally, a new case index encoding approach is proposed to encode all possible cases in a heterogeneous tri-prism voxel that can verify the proposed DA approach in an exhaustive enumeration manner. The experimental results demonstrate that our approach can accurately and efficiently generate a boundary representation of heterogeneous volumetric data.

1. Introduction

Heterogeneous objects are ubiquitous in the real world. A heterogeneous object is composed of multiple homogeneous material components [1]. The reconstruction of a geometric representation of heterogeneous objects is essential in many geometry-related applications [2]. For example, the computation of a geometric representation that distinguishes different functional parts of a digital liver is crucial for surgical planning. In this paper, we focus on reconstructing surfaces at which different materials meet in one pass to display and explore heterogeneous objects.

In heterogeneous object reconstruction, topologies are combined with heterogeneous materials. The meshing approach should meet the following requirements when reconstructing a geometric representation of a heterogeneous object:

- The boundary surfaces should classify the heterogeneous volume into homogeneous parts without T-junctions, cracks or holes.
- Each homogeneous part is enclosed by its corresponding boundary surface. Furthermore, each boundary surface should be a piecewise 2-manifold surface, allowing for additional graphical and geometrical processing.
- The approach should be independent of the number of material types employed.
- The common patches of two boundary surfaces should be topologically consistent.

Only a few efforts have been reported for the efficient reconstruction of heterogeneous objects. Existing solutions address the meshing problem by using voxels (either cubical or tetrahedral). We reconstruct the surfaces from tri-prisms obtained by subdividing cubical voxels [3]. Using the new case index encoding approach we provided herein, the triangulation types in the tri-prisms can be degenerated into a number of cases and verified.

The use of tri-prisms also enables the application of conventional dimension-ascending (DA) schemes, that is, the points along the edges, the edges connecting the points, the triangles are generated sequentially. The interfaces and boundary surfaces can be obtained by carefully organizing the triangles. A hierarchical data structure is designed to facilitate the above-described surface extraction process.

In summary, the main contributions of this paper include the following:

- A new efficient tri-prism-based surface extraction scheme for heterogeneous objects in a DA manner, independent of the number of material types present.
- An effective triangle formation scheme that produces piecewise 2-manifold and orientable boundary surfaces and explicitly preserves the topological relationship.

The remaining sections are organized as follows. After describing the related work in Section 2, a hierarchical data structure is

* Corresponding author at: Zhejiang University, Hangzhou, China.
E-mail address: jqfeng@cad.zju.edu.cn (J.-Q. Feng).

0097-8493/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.10.034

designed in Section 3. The details of the proposed approach are described in Section 4. The implementation results and a discussion are presented in Section 5. Finally, conclusions are drawn and future work is highlighted in Section 6.

2. Related work

Isosurface extraction: The extraction of the boundary surfaces of a 3D shape from scalar field or volumetric data has been an important research topic in computer graphics and scientific visualization for decades. Wyvill et al. first proposed the polygonization method of implicitly defined surfaces [4]. Lorenson et al. then proposed the famous Marching Cube (MC) algorithm, which extracts the isosurface from scalar volume data using a look-up table [5,6]. Bloomenthal proposed an adaptive polygonization method to more efficiently capture fine geometric details [7]. To overcome ambiguity in the extracted isosurface, some improvements and extensions of the MC algorithm have been proposed, and these improvements can successfully preserve the topological consistency [8,9]. As an alternative to Wyvill's method, Poston et al. proposed the skeleton-climbing algorithm to extract the isosurface in a topology-ascending manner, *i.e.*, building the isosurface step by step according to its intersections with grid edges (1-skeleton), then faces (2-skeleton), then cubes (3-skeleton) [10,11]. Fujishiro et al. extended the concept of traditional surface fitting to define and extract a boundary surface associated with the scalar values and a user-specified interval [12,13]. Ju et al. combined the extended marching cubes algorithm with a dual algorithm to generate polygonal surfaces from Hermite data. Different from the cube- or tetrahedron-based isosurface methods, Xu et al. extracted the isosurface from tri-prisms voxels [3]. However, these methods can extract only the interface between two materials and cannot be trivially adapted to the heterogeneous cases.

Heterogeneous object modeling: In a pioneering work, Nielson et al. proposed a marching tetrahedra method to mesh a multi-material object [14]. The method can partition each heterogeneous tetrahedron into homogeneous material parts. However, the resulting boundary surfaces are given in terms of polygon soups. Wang et al. developed a so-called boundary surfaces extraction from the heterogeneous object (BSHO) method to build the interface surfaces in a tetrahedral volume data [1]. However, the BSHO method is time consuming, requires a large amount of memory, and can hardly be performed on a desktop PC. Alternatively, Wu et al. proposed the multi-material marching cubes (M3C) method to reconstruct boundary surfaces from a volumetric medical image by directly extracting the surfaces from cubical voxels [15]. The M3C method is focused on separating heterogeneous objects with accurate geometry but does not maintain the polygon orientations in its meshing step, which may lead to visual defects unless postprocessing is imposed. Feng et al. combined trilinear contouring and volume rendering to directly display the heterogeneous objects on a GPU directly [16]. However, their implementation does not explicitly provide a geometric representation. Dey et al. provided a method to mesh piecewise-smooth complexes defined by multi-label datasets [17,18]. Interface surfaces are iteratively created by a Delaunay meshing algorithm. The triangles in the interface surfaces are well shaped, and the corresponding tetrahedra meshes can be produced at no extra cost. However, their method involves subtle pre-processing steps which have potential impacts on the interface surface generation [17].

The CAD community has also explored heterogeneous object modeling and representation [19]. Wang proposed an alternative solution to the boundary representation of heterogeneous objects [2,20]. However, the information on topological intersections cannot be preserved because the boundary surfaces for the materials are optimized individually. Shammaa et al. combined region growing and graph-cut to precisely classify a volumetric model into several homogeneous material parts [21]. However, as indicated by the authors, this approach cannot robustly process heterogeneous objects containing more than three materials.

3. Hierarchical boundary data structure for the dimension-ascending process

A hierarchical data structure is designed to efficiently reconstruct boundary surfaces and query the topological and geometric information on intersections between/among boundary surfaces, as shown in Fig. 1.

In the proposed approach, the geometric representation of a heterogeneous object $\mathscr{H}$ is given as a union of boundary surfaces. A *boundary surface* is a set of the interfaces isolating one material component from the others, denoted as Ω. An *interface* is a set of separating triangles between two materials, denoted as Γ. Each interface is built only once and is shared by the two boundary surfaces to achieve topological consistency. *Separating triangles* (**ST**s) are obtained by triangulating the loops formed by sequentially connecting the separating line segments. A *separating line segment* (**SL**) is defined by connecting two separating points. A *separating point* (**SP**) is the point on the edge or in the face of a tri-prism at which two, three, or four materials meet. The material information is recorded in the separating points and is inherited by the separating line segments, separating triangles, interfaces, and boundary surfaces at each level.

4. Our approach

Algorithm 1 describes the framework of the approach. The regular cubical volumetric data in Algorithm 1 is a cubical grid sampled over a discrete set $V \subset \mathbb{R}^3$. A cubical voxel is first subdivided into two tri-prisms. The boundary surfaces are reconstructed in a DA manner. First, the **SP**s on the tri-prism's edges or faces are extracted. Second, oriented **SL**s are generated by connecting the extracted **SP**s, and loops are formed by properly traversing the **SL**s. Third, triangles, obtained by triangulating the loops, are clustered as interfaces according to their material indices. Finally, boundary surfaces are generated by grouping the related interfaces.

Algorithm 1.

1: **Input**: labeled regular cubical volumetric data;
2: **for** each cube **do**
3: Subdivide the cube into two tri-prisms;
4: Extract **SP**s on the edges or in the faces;

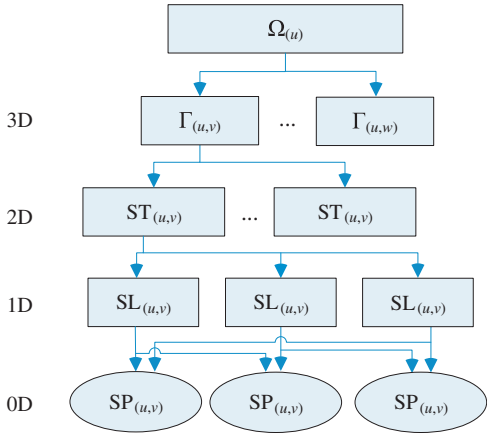


Fig. 1. Hierarchical data structure.

5: Connect the **SP**s to form the **SL**s;
6: **for** each tri-prism **do**
7: Generate loops by connecting the **SL**s;
8: Triangulate the loops;
9: **end for**
10: **end for**
11: Merge the shared **SP**s for 2-manifold preservation;
12: Cluster triangles to generate interfaces;
13: Group the interfaces to form the boundary surfaces.

4.1. Tri-prism decomposition

It is reasonable to perform surface extraction using cubes rather than tri-prisms to generate fewer triangles. However, there may be 8^8 cases in a cube if the underlying region is heterogeneous, making the procedure impractical. Furthermore, it is difficult to properly separate a heterogeneous cube in some cases. As shown in Fig. 2, one such case is the cube, whose diagonal vertex pairs are supposed to have the same material types. An octahedron region at the interior of the cube will remain undetermined if a cube centroid point is not introduced, as shown in Fig. 2. If a cube centroid node (point) is introduced, as reported by Wu et al., a cube will be blockwise partitioned, blocking the diagonal corners with identical materials, as shown in Fig. 2(a) [15]. Our approach partitions the heterogeneous cube into homogeneous subspaces that are not undetermined or overlapped and ensures that at least one homogeneous diagonal corner is connected, as shown in Fig. 2(b). This result is achieved by subdividing the cube into a pair of tri-prisms, as shown in Fig. 3. The two tri-prisms are termed as **OPsm** (Odd Tri-Prism, defined by the cube vertices C_0, C_2, C_3, C_4, C_6 and C_7) and **EPsm** (Even Tri-Prism, defined by C_2, C_0, C_1, C_6, C_4 and C_5). The points P_i ($i=0\ldots18$) in Fig. 3 are potential **SP**s, which will be described in the following subsection.

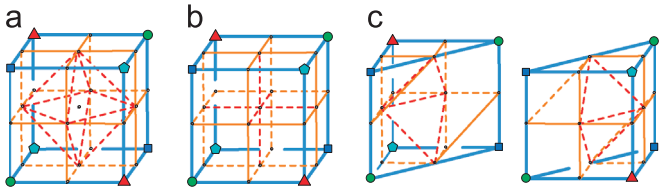


Fig. 2. (a) If a cube centroid point is not introduced, an octahedron outlined by the red dashed lines will remain undetermined. (b) M3C introduces a cube centroid point and generates a blockwise partitioning result [15]. (c) The use of tri-prisms can effectively separate the cube with the diagonal vertices connected as much as possible. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

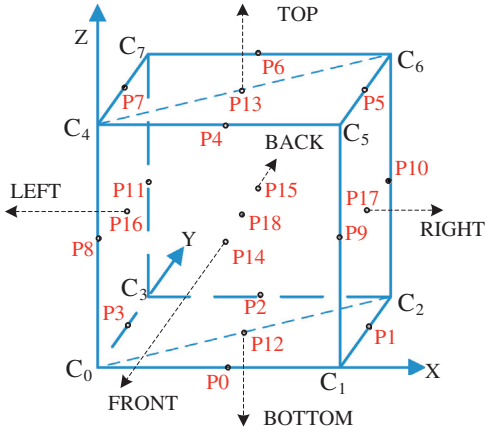


Fig. 3. Subdividing a cube into two tri-prisms. The points $P_i(i=0,\ldots,18)$ are potential **SP**s in a cube. The six dashed arrows indicate the normal vectors of the cube faces.

4.2. Material-oriented index encoding method

The traditional case index encoding method is not suitable for heterogeneous cases. For example, there are a total of $3^3(4^4)$ cases of material distribution in a triangular (quadrilateral) face, whereas there are at most three (or six) partitioning patterns of the triangular (quadrilateral) face when symmetry, rotation invariant, and complementarily are considered. To facilitate the subsequent processing, a new case indexing method, called the material-oriented index encoding method (MOIEM), is proposed to remarkably reduce the case number. The MOIEM encodes the case index of a face according to whether the material at the corner is within the set of the materials of the encoded corners.

Let us illustrate the MOIEM by encoding a triangular face with a 16-bit unsigned integer. The material indices are denoted by u, v, w, and x. We assume that $u<v<w<x$ without loss of generality. M_{C_i} denotes the material at corner C_i. MOIEM encodes the triangular face as follows. (a) The first 4-bit is 0 ($0\times0^{***}$) because the face contains only three corner vertices. (b) The second 4-bit is assigned to 1 ($0\times01^{**}$) because M_{C_0}, *i.e.*, u, is the first material encountered in the triangular face. (c) M_{C_1}, *i.e.*, v, is the second material encountered, so the third 4-bit is assigned to 2 ($0\times012^{*}$). (d) The fourth 4-bit is assigned to 1 (0×0121) if $M_{C_2}=M_{C_0}$. If M_{C_2} differs from M_{C_0} and M_{C_1}, the fourth 4-bit is assigned to 3 (0×0123). The quadrilateral face is encoded in a similar manner by the MOIEM.

4.3. Separating points (*SP*s)

SPs are the primary element in the proposed approach. There will be an **SP** on the edge if the material types of a tri-prism edge's two endpoints are different. For a quadrilateral face, there will also be an **SP** introduced in the quadrilateral face if the four corner vertices are associated with more than two material types and if the material types at the diagonal corner vertices are different.

As shown in Fig. 4, the five faces of a tri-prism are F_BOTTOM ($C_1C_0C_2$), F_TOP ($C_3C_4C_5$), F_SIDE1 ($C_2C_0C_3C_5$), F_SIDE2 ($C_1C_2C_5C_4$), and F_SIDE3 ($C_0\,C_1C_4C_3$), where C_i (i=0,…, 5) are the corner vertices of the tri-prism. The points labeled from 6 to 17 are the **SP**s in a tri-prism. It is natural to introduce a face centroid point to separate the

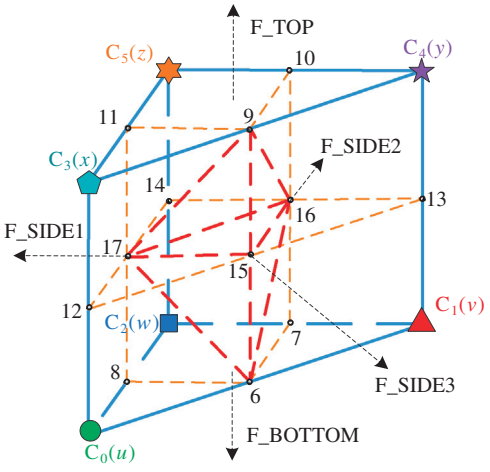


Fig. 4. A tri-prism: The points labeled from 6 to 17 are **SP**s. The thin dashed line segments connecting these points are **SL**s. The five dashed arrows indicate the outwards normal vectors of the tri-prism faces.

heterogeneous quadrilateral faces. However, for the triangular faces in Fig. 4, the points on their hypotenuses act as the face centroid points. In this manner, the **SP** in the tri-prism can be avoided. As a result, the connectivity of the cube can be preserved as much as possible. However, this achievement comes at a cost: $\mathbf{SP}_6$ and $\mathbf{SP}_9$ are considered as **SP**s on the edge and in the triangular face simultaneously, which will make the loop generation step slightly more complex. The positions of the **SP**s can be determined as the mid-point of an edge or as the barycenter of a face.

4.4. Separating line segment (*SL*)

After **SP**s are obtained, **SL**s are generated by properly connecting the **SP**s. There are two types of **SL**s. One is defined on the tri-prism faces, noted as $\mathbf{SL}^{(F)}$, whereas the other is defined in the tri-prism volume, noted as $\mathbf{SL}^{(V)}$, as shown in Fig. 4.

4.4.1. *SL*s on the tri-prism face ($\mathbf{SL}^{(F)}$)

According to the MOIEM, except for homogeneous cases, the triangular and quadrilateral faces can be encoded as 4 and 14 cases, respectively, as shown in Figs. 5 and 6.

The direction of an $\mathbf{SL}^{(F)}$ is important because it will be used to determine the orientation of a loop in the next step. The direction of an $\mathbf{SL}^{(F)}$ is consistently determined such that the corner vertex with the lower material index is at its left side if viewed against its face's normal vector. In Figs. 5 and 6, the direction of the $\mathbf{SL}^{(F)}$s is given under the assumption that the material indices satisfy $u<v<w<x$. The $\mathbf{SL}^{(F)}$ from $\mathbf{SP}_1$ to $\mathbf{SP}_2$ is denoted as $\mathbf{SL}^{(F)}_{(SP_1,SP_2)}$ in the remainder of the paper.

4.4.2. *SL* in tri-prism ($\mathbf{SL}^{(V)}$)

An $\mathbf{SL}^{(V)}$ is defined by connecting two face centroid **SP**s. The $\mathbf{SL}^{(V)}$s outline the skeleton of the internal structure. The $\mathbf{SL}^{(V)}$s,

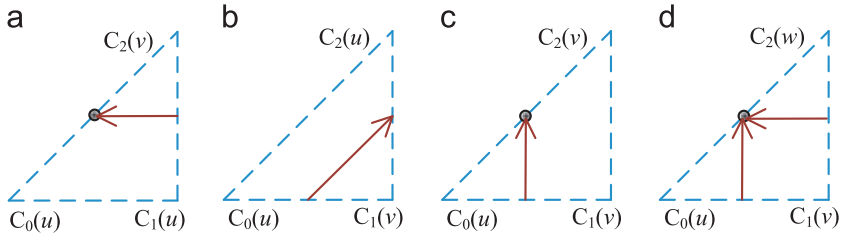


Fig. 5. Except for the homogeneous case, **SL** s in a triangular face can be encoded as four cases. The direction of an $\mathbf{SL}^{(F)}$ is determined such that the corner vertex with the lower material index is at the left side of the $\mathbf{SL}^{(F)}$. (a) 0×0112, (b) 0×021, (c) 0×0122, and (d) 0×0123.

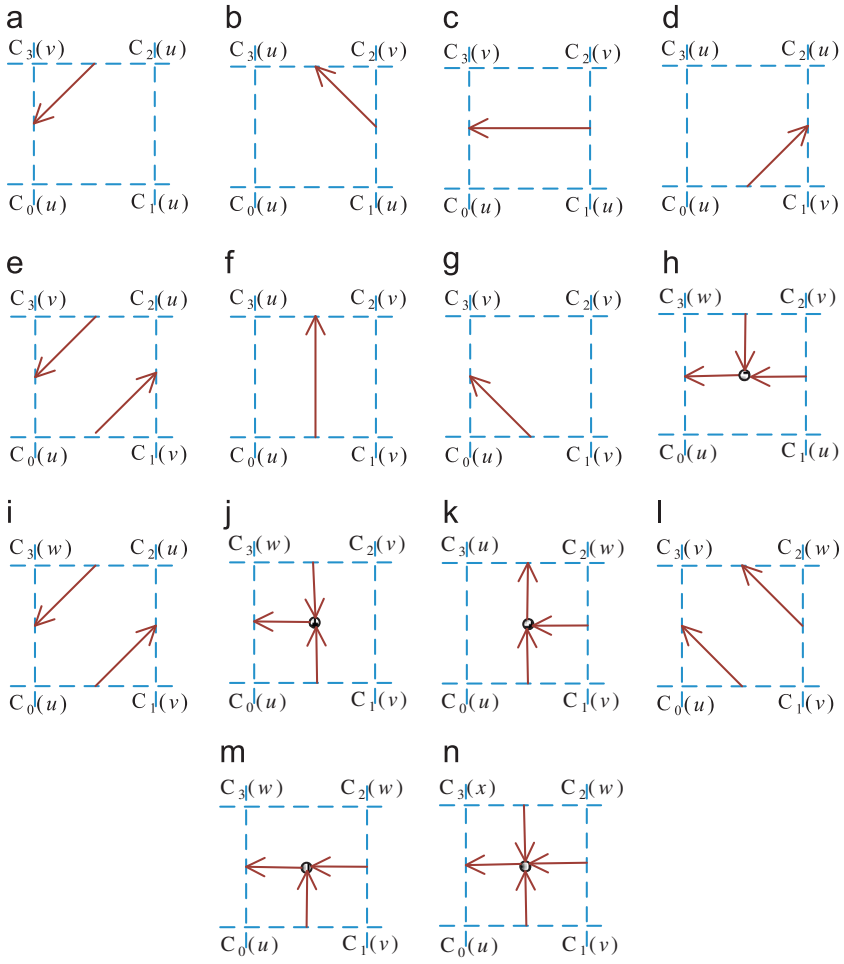


Fig. 6. Except for the homogeneous case, the $\mathbf{SL}^{(F)}$ s in a quadrilateral face can be encoded as 14 cases. (a) 0×1112, (b) 0×1121, (c) 0×1122, (d) 0×1211, (e) 0×1212, (f) 0×1221, (g) 0×1222, (h) 0×1123, (i) 0×1213, (j) 0×1223, (k) 0×1231, (l) 0×1232, (m) 0×1233, and (n) 0×1234.

together with $\mathbf{SL}^{(F)}$s, partition the tri-prism into subspaces that are not undetermined or overlapped. The prerequisite of introducing an $\mathbf{SL}^{(V)}$ is that the two related **SP**s must be extracted. The seven $\mathbf{SL}^{(V)}$s are independently generated according to their respective rules. The detailed rules are described as follows:

- $\mathbf{SL}^{(V)}_{(6,16)}$ will be added if $\mathbf{SP}_7$ is extracted or the $\mathbf{SL}^{(F)}_{(6,13)}$ is generated.
- $\mathbf{SL}^{(V)}_{(6,17)}$ will be added if $\mathbf{SP}_8$ is extracted or the $\mathbf{SL}^{(F)}_{(6,12)}$ is generated.
- $\mathbf{SL}^{(V)}_{(9,16)}$ will be added if $\mathbf{SP}_{10}$ is extracted or the $\mathbf{SL}^{(F)}_{(9,13)}$ is generated.
- $\mathbf{SL}^{(V)}_{(9,17)}$ will be added if $\mathbf{SP}_{11}$ is extracted or the $\mathbf{SL}^{(F)}_{(9,12)}$ is generated.
- $\mathbf{SL}^{(V)}_{(15,16)}$s and $\mathbf{SL}^{(V)}_{(15,17)}$s will be added as long as the prerequisite is satisfied.
- $\mathbf{SL}^{(V)}_{(16,17)}$ will be added if one of the following four conditions is false: (a) no $\mathbf{SP}_{14}$; (b) no $\mathbf{SL}^{(F)}$ sequence formed by $\mathbf{SL}^{(F)}_{(17,11)}$, $\mathbf{SL}^{(F)}_{(11,10)}$ and $\mathbf{SL}^{(F)}_{(10,16)}$; (c) no $\mathbf{SL}^{(F)}$ sequence formed by $\mathbf{SL}^{(F)}_{(17,8)}$, $\mathbf{SL}^{(F)}_{(8,7)}$ and $\mathbf{SL}^{(F)}_{(7,16)}$; (d) no $\mathbf{SP}_{15}$ and no $\mathbf{SL}^{(V)}$ sequence formed by $\mathbf{SL}^{(V)}_{(6,16)}$, $\mathbf{SL}^{(V)}_{(16,9)}$, $\mathbf{SL}^{(V)}_{(9,17)}$ and $\mathbf{SL}^{(V)}_{(17,6)}$.

Furthermore, $\mathbf{SL}^{(F)}_{(9,15)}$ and $\mathbf{SL}^{(F)}_{(6,15)}$ are $\mathbf{SL}^{(V)}$s even though they are generated according to the rule of $\mathbf{SL}^{(F)}$ because $\mathbf{SP}_6$ and $\mathbf{SP}_9$ are also considered as the face centroid **SP**s in the triangular faces.

In the following loop generation step, each $\mathbf{SL}^{(V)}$ will be reused and their directions are trivial.

4.4.3. Special cases

There are two special cases in which the $\mathbf{SL}^{(F)}$ s in F_SIDE3 should be carefully processed, as shown in Fig. 7. (a)When the case indices of all three quadrilateral faces are 0×1232 or 0×1213, $\mathbf{SL}^{(F)}_{(6,9)}$ will be introduced. (b) When the case indices of F_SIDE3 and F_SIDE1 are 0×1221 and 0×1223, respectively, both the tri-prism and its companion tri-prism, *e.g.* the **OPsm** and its **EPsm**, will add the centroid point to their F_SIDE3 ($\mathbf{SP}_{15}$). Thus, $\mathbf{SL}^{(F)}_{(6,9)}$ (or $\mathbf{SL}^{(F)}_{(9,6)}$) in F_SIDE3 will be replaced by $\mathbf{SL}^{(F)}_{(6,15)}$ and $\mathbf{SL}^{(F)}_{(15,9)}$ (or $\mathbf{SL}^{(F)}_{(9,15)}$ and $\mathbf{SL}^{(F)}_{(15,6)}$).

4.5. Loop generation and triangulation

A loop is generated by properly connecting the **SL**s, which can potentially span a surface to separate a homogenous part from the others regions. The loops can be classified into three types according to the **SL**s they contained: (i) composed of $\mathbf{SL}^{(V)}$ s only; (ii) composed of $\mathbf{SL}^{(F)}$ s and $\mathbf{SL}^{(V)}$ s; and (iii) composed of $\mathbf{SL}^{(F)}$ s only. The loops are constructed in the order of type (i), (ii) and (iii). Except for $\mathbf{SL}^{(F)}_{(6,9)}$, which is also considered as an $\mathbf{SL}^{(V)}$, each $\mathbf{SL}^{(F)}$ is employed only once in the loop generation.

There are a total of seven configurations for type (i) loops. Each configuration contains only one triangle, so the loop is directly denoted as $\Delta_{(p,q,r)}$, where p, q, and r are the indices of the **SP**s. The loop generation rules are closely related to the $\mathbf{SL}^{(V)}$ s. In the rules given below, M_{C_i} indicates the material index of the tri-prism corner vertex C_i, and m^+ (m^-) indicates the material index at the positive (negative) side of the loop (right-hand screw rule)

- $\Delta_{(9,16,17)}$: $m^+ = M_{C_5}$,
$$m^- = \begin{cases} M_{C_0} & \text{if} \quad C_0 \text{ and } C_4 \text{ are connected;} \\ M_{C_1} & \text{else} \end{cases}$$
- $\Delta_{(17,16,6)}$: $m^+ = M_{C_2}$,
$$m^- = \begin{cases} M_{C_0} & \text{if} \quad C_0 \text{ and } C_4 \text{ are connected;} \\ M_{C_1} & \text{else} \end{cases}$$
- $\Delta_{(15,16,17)}$: $m^+ = M_{C_5}$, $m^- = M_{C_2}$;
- $\Delta_{(9,17,15)}$: $m^+ = M_{C_3}$,
$$m^- = \begin{cases} M_{C_4} & \text{if} \quad \mathbf{SL}^{(F)}_{(10,14)} \text{ exists;} \\ M_{C_5} & \text{else} \end{cases}$$
- $\Delta_{(15,17,6)}$: $m^+ = M_{C_0}$,
$$m^- = \begin{cases} M_{C_1} & \text{if} \quad \mathbf{SL}^{(F)}_{(7,14)} \text{ exists;} \\ M_{C_2} & \text{else} \end{cases}$$
- $\Delta_{(15,16,9)}$: $m^+ = M_{C_4}$,
$$m^- = \begin{cases} M_{C_3} & \text{if} \quad \mathbf{SL}^{(F)}_{(11,14)} \text{ exists;} \\ M_{C_5} & \text{else} \end{cases}$$
- $\Delta_{(6,16,15)}$: $m^+ = M_{C_1}$,
$$m^- = \begin{cases} M_{C_0} & \text{if} \quad \mathbf{SL}^{(F)}_{(8,14)} \text{ exists;} \\ M_{C_2} & \text{else} \end{cases}$$

The loops above are generated under the assumption that $m^+ < m^-$. Otherwise, the three **SP**s corresponding to Δ are recorded in the reverse order, and the values of m^+ and m^- are switched.

Let $\mathbf{S}_{(SP)} = \{SP_6, SP_9, SP_{15}, SP_{16}, SP_{17}\}$. The type (ii) loops can be generated in three steps: (a) choose an unemployed $\mathbf{SL}^{(F)}$ whose starting point $\mathbf{SP}_{start}$ is in $\mathbf{S}_{(SP)}$; (b) trace $\mathbf{SL}^{(F)}$ s in a consistent direction, tail to head, until one reaches the $\mathbf{SL}^{(F)}$ whose ending point $\mathbf{SP}_{end}$ is also in $\mathbf{S}_{(SP)}$; and (c) close the $\mathbf{SL}^{(F)}$ sequence with $\mathbf{SL}^{(V)}_{(\mathbf{SP}_{end},\mathbf{SP}_{start})}$ or with the only two $\mathbf{SL}^{(V)}$s in the tri-prism. A detailed example is provided in Fig. 8.

The type (iii) loops are only composed of $\mathbf{SL}^{(F)}$ s with consistent directions. These loops can be generated by tracing $\mathbf{SL}^{(F)}$ s from end to end. The process of loop generation is not complete until all of the $\mathbf{SL}^{(F)}$ s have been traversed.

The orientation of a type (ii) or type (iii) loop is determined by the direction of the $\mathbf{SL}^{(F)}$ s that it contains according to the right-hand screw rule. The material index at the loop's positive (negative) side is identical to the material index of the $\mathbf{SL}^{(F)}$'s left (right) side.

The loops are triangulated directly after being generated. A maximum of five triangles are generated because none of the loops contain more than seven edges. The loops are triangulated in three steps: (1) generate a triangle with three successive **SP**s in the loop, which yields the largest aspect ratio; (2) retain the first and third **SP**s of the successive three to reuse and delete the second **SP**; (3) repeat the previous two steps until fewer than three **SP**s remain in the loop.

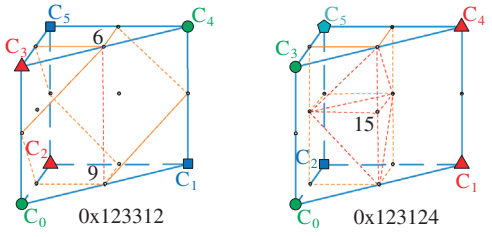


Fig. 7. Two special cases.

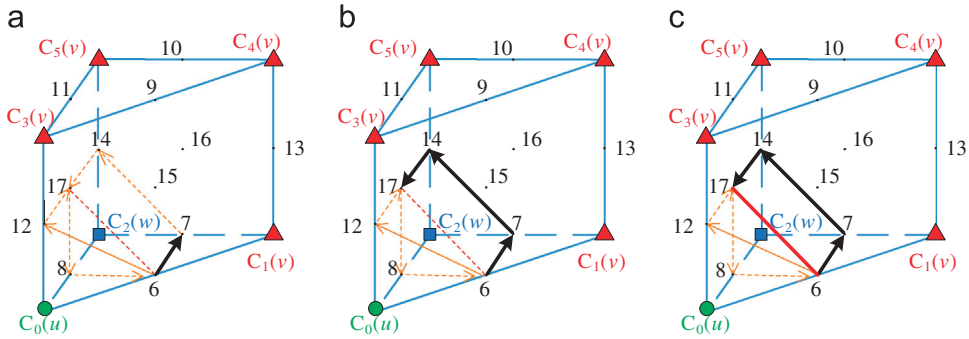


Fig. 8. Generation of a type (ii) loop: (a) begin with $\mathbf{SL}^{(F)}_{(6,7)}$, whose starting point $\mathbf{SP}_6 \in \mathbf{S}_{\langle SP\rangle}$; (b) trace $\mathbf{SL}^{(F)}$ s in a consistent direction until $\mathbf{SL}^{(F)}_{(14,17)}$ is reached, whose ending point $\mathbf{SP}_{17} \in \mathbf{S}_{\langle SP\rangle}$; (c) close the above $\mathbf{SL}^{(F)}$ s sequence with $\mathbf{SL}^{(V)}_{(17,6)}$.

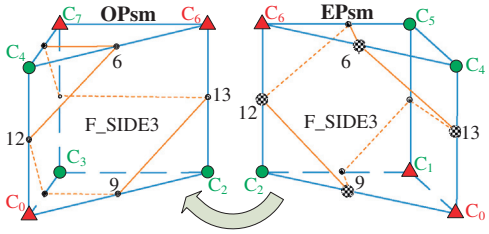


Fig. 9. For the diagonal case, $\mathbf{SL}^{(F)}$ s are connected counter-diagonally in the **OPsm** and primary-diagonally in the **EPsm**. To obtain a watertight representation, the **SP**s of the **EPsm**, *i.e.*, $\mathbf{SP}_6$, $\mathbf{SP}_9$, $\mathbf{SP}_{12}$ and $\mathbf{SP}_{13}$, are replaced with the **SP**s of the **OPsm**, *i.e.*, $\mathbf{SP}_6$, $\mathbf{SP}_9$, $\mathbf{SP}_{13}$ and $\mathbf{SP}_{12}$, respectively.

4.6. 2-Manifold preservation and verification of the separating patterns

The quadrilateral faces with diagonally identical materials should be carefully processed [22]; this situation is termed the diagonal case throughout the remainder of this paper. The $\mathbf{SL}^{(F)}$ generation rule is slightly modified in the diagonal case for the sake of 2-manifold preservation. The $\mathbf{SL}^{(F)}$ s are connected counter-diagonally in the **OPsm** and primary-diagonally in the **EPsm** as shown in Fig. 9.

The following three results demonstrate the following: (a) the proposed DA algorithm can be verified and (b) the surfaces produced by the proposed DA algorithm are watertight and piecewise 2-manifold.

First, the surface patches extracted from a tri-prism are watertight. The MOIEM can be extended to encode heterogeneous tri-prisms with a 32-bit unsigned integer. By exhaustively enumerating the 203 cases from 0×111111 to 0×123456, each case separates a tri-prism into subspaces that are not undetermined or overlapped.

Second, the surface patches extracted from a tri-prism pair are watertight. Table 1 presents the **SP**s' index map for the tri-prism pair to their cube. The numbers in the first row of Table 1 are the **SP** indices of the tri-prisms, whereas the numbers in the remaining rows are the corresponding **SP** indices in the cube. The **SP**s in the cutting plane (F_SIDE3) are independently extracted from **OPsm** and **EPsm** and are merged to obtain a watertight representation. As an example, in Fig. 9, the **SP**s of the **EPsm**, *i.e.*, $\mathbf{SP}_6$, $\mathbf{SP}_9$, $\mathbf{SP}_{12}$, and $\mathbf{SP}_{13}$, are replaced with the **SP**s of the **OPsm**, *i.e.*, $\mathbf{SP}_6$, $\mathbf{SP}_9$, $\mathbf{SP}_{13}$, and $\mathbf{SP}_{12}$, respectively. For the special tri-prisms shown in Fig. 7 (left), the companion **EPsm** would be one of the 17 cases: 0×122232, 0×121231, 0×123233, 0×123243, 0×121232, 0×122231, 0×123232, 0×122233, 0×123242, 0×122234, 0×123231, 0×121233, 0×123241, 0×121234, 0×123244, 0×123234, or 0×123245. For the special tri-prism shown in Fig. 7 (right), the companion **EPsm** would be one of the 10 cases: 0×122122, 0×121121, 0×123123, 0×121122, 0×122121, 0×123122, 0×122123, $0\times 12-3121$, 0×121123, or 0×123124. All of these cases are watertight.

Table 1
The map of separating points between the cube and its tri-prisms.

SP	6	7	8	9	10	11	12	13	14	15	16	17
OPsm	P_{12}	P_2	P_3	P_{13}	P_6	P_7	P_8	P_{10}	P_{11}	P_{18}	P_{15}	P_{16}
EPsm	P_{12}	P_0	P_1	P_{13}	P_4	P_5	P_{10}	P_8	P_9	P_{18}	P_{14}	P_{17}

Third, the patches extracted from adjacent cubes are watertight because the common **SP**s independently extracted from adjacent cubes are replaced by the **SP** with the lowest subscript, and the common faces shared by adjacent cubes are separated with same topology. As an example, in Fig. 10, the patches extracted from a $2\times 2\times 2$ dataset are watertight.

4.7. Interface and boundary surfaces

Each loop is generated with an orientation, having the material with the lower index at its positive side. The triangles produced by triangulating a loop will inherit the loop's orientation. An interface is a set of triangles that isolates one material from the others. The interface is generated by clustering the triangles associated with the same materials, as described in the below equation:

$$\Gamma_{(u,v)} = \begin{cases} \{\Delta s | m^+ = u, m^- = v\} & \text{if } u < v \\ \{\Delta s | m^+ = v, m^- = u\} & \text{if } v < u \end{cases} \tag{1}$$

where $\Gamma_{(u,v)}$ is the interface and Δs denotes the related triangle set. The orientations of the triangles belonging to an interface are clearly consistent.

According to the data structure defined in Section 3, the boundary surface related to a material, *e.g.*, u, is composed of the interfaces that isolate the material u from the other materials. Thus, the boundary surface surrounding the material u can be constructed by grouping the interfaces related to material u, as described in the below equation:

$$\Omega_u = \bigcup_{i=-1,\ldots,n-1, i\neq u} \Gamma_{(u,i)} \tag{2}$$

where n is the number of materials of interest and -1 is the material index for materials that are not of interest. Because all of the interfaces about a specific material are orientable and piecewise 2-manifold, the boundary surface for the material is also orientable and piecewise 2-manifold.

Similarly, as described in Eq. (3), the geometric representation of a heterogeneous object $\mathscr{H}$ is also the union of interfaces:

$$\mathscr{H} = \bigcup_{i=-1,\ldots,n-1} \Omega_i = \bigcup_{i,j=-1,\ldots,n-1, i\neq j} \Gamma_{(i,j)} \tag{3}$$

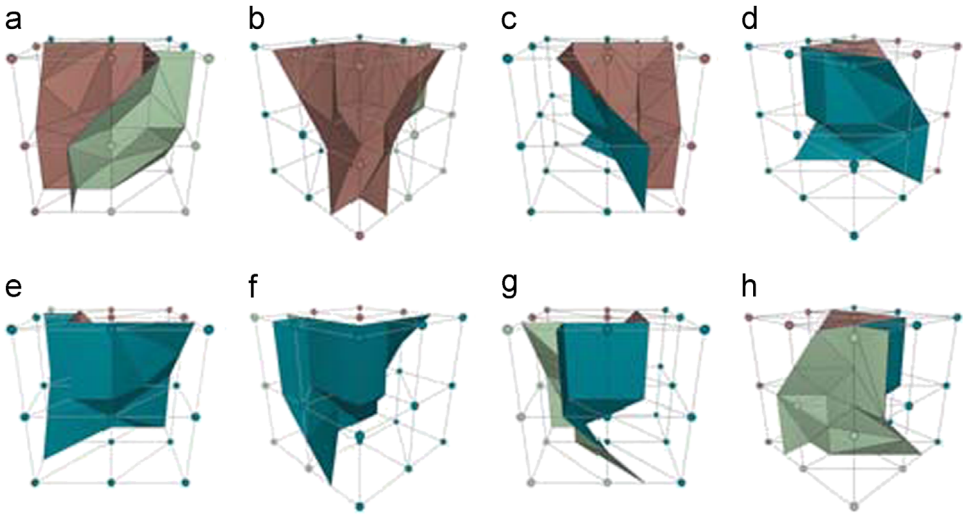


Fig. 10. The patches extracted from a $2 \times 2 \times 2$ cubical dataset are watertight, and they are depicted with different perspective views. (a) 0°, (b) 45°, (c) 90°, (d) 135°, (e) 180°, (f) 225°, (g) 270°, and (h) 315°.

The intersection of two boundary surfaces is exactly equal to the interface between the two materials. This assertion can also be obtained according to the definition of boundary surface in Eq. (2). Thus, the intersections can be efficiently described by the below equation:

$$\Omega_u \bigcap \Omega_v = \Gamma_{(u,v)} \tag{4}$$

5. Implementation and results

The proposed approach was implemented on a PC with an Intel Core i5-2300 2.8 GHz CPU with 8 GB of main memory, a NVIDIA Quadro FX 5800 graphics card, and Windows 7 OS. OpenGL was employed to render the output.

5.1. Implementation and acceleration

In general, more than 60% of heterogeneous voxels in a dataset contain only two materials. Thus, it is inefficient to process all of the voxels in the same manner. Therefore, we designed a strategy for acceleration. The three steps of the strategy are as follows: (a) classify all of the heterogeneous voxels into two groups, two- and multi-material voxels; (b) process the two-material voxels according to the pre-generated case table and accelerate the processing using multi-thread technology; and (c) process the multi-material voxels in a DA manner. The pre-generated case table is derived from the DA implementation, whose indices are encoded according to the MOIEM. The separating triangles in the entries are produced by the DA implementation. Because the case table is derived from the DA implementation, it is compatible with the DA approach. The accelerated DA approach is denoted as DA*.

In addition to the DA and DA* approaches, we implemented the M3C [15] and BSHO [1] method for comparison. The M3C method also faithfully separates the heterogeneous objects. However, the raw mesh exhibits some visual defects because the orientations of the triangles generated by the M3C method are inconsistent. We improved upon the M3C method by rectifying the triangles with inconsistent orientations. Let $\vec{n}$ be the triangle's normal vector, $\vec{v}$ be the vector from the triangle centroid to the nearest cube corner C_n, M_{C_n} be the material index of the C_n and t^+ (t^-) be the material index of the triangle's positive (negative) side. The orientation will be reversed if the triangle satisfies any of the following conditions: (a) $\vec{n} \cdot \vec{v} < 0$ and $M_{C_n} = t^+$; (b) $\vec{n} \cdot \vec{v} \geq 0$ and $M_{C_n} = t^-$.

Another minor improvement is required to accelerate the M3C method in a manner similar to DA*. Regarding the diagonal case, the original M3C method forces the corners with the lower material index to connect, which makes it difficult for the original M3C method to derive a concise case table for two-material voxels. Therefore, we separate the diagonal cases in a fixed pattern: If a diagonal case contains C_1, it is separated as the F_SIDE3 of EPsm in Fig. 9; if not, it is separated as the F_SIDE3 of OPsm in Fig. 9. The improved M3C method is denoted as M3C*.

5.2. Results and comparison

The proposed DA approach was verified by both simulation and measured datasets. The first example is the BluntFin, which is a freely available open dataset [23] (http://www.informatik.uni-erlangen.de/External/vollib/). This dataset is the simulation of airflow over a flat plate with a blunt fin rising from the plate, which is also recognized as the Utah Teapot in Computational Fluid Dynamics Visualization. As shown in Fig. 11, the BluntFin contains three clearly distinct components, which are rendered in maroon, olive, and teal. The boundary surfaces of the components are denoted as Ω_m, Ω_o and Ω_t. Ω_i' ($i = m, o$ or t) provides information for the adjacent component, which is obtained by switching the m^+ and m^- of each triangle in Ω_i. Ω_i' is equal to Ω_i in geometry. The materials that are not of interest are denoted as -1, and the boundary is rendered in gray. Each material is delimited by its boundary surface, which is composed of interfaces. The $\Gamma_{(m,t)}$ is rendered together with $\Gamma_{(o,t)}$ for conciseness because the $\Gamma_{(m,t)}$ contains only six triangles.

The second dataset is the Utah Torso model, which is a high-resolution simulation dataset obtained from finite element analysis of the human thorax [24]. The scale values are sampled via SCIRun software [25]. As shown in Fig. 12, the boundary surfaces of six organs, *i.e.*, the liver, kidney, lung, bone, artery and heart, are extracted in one pass.

The dataset for the liver model shown in Fig. 13(a) was acquired from a patient. A liver can be segmented into functional parts according to the internal blood vessel distributions. For liver surgery, the segment information of the patient's individual liver is crucial for estimating the risk of operation strategies. The proposed approach can reconstruct the segmented liver model in one pass.

The last example, shown in Fig. 14, is the Schaedel model. The dataset is provided with courtesy of Tiani Medgraph (http://www.tiani.com/). There are 10 materials (objects) in the Schaedel model,

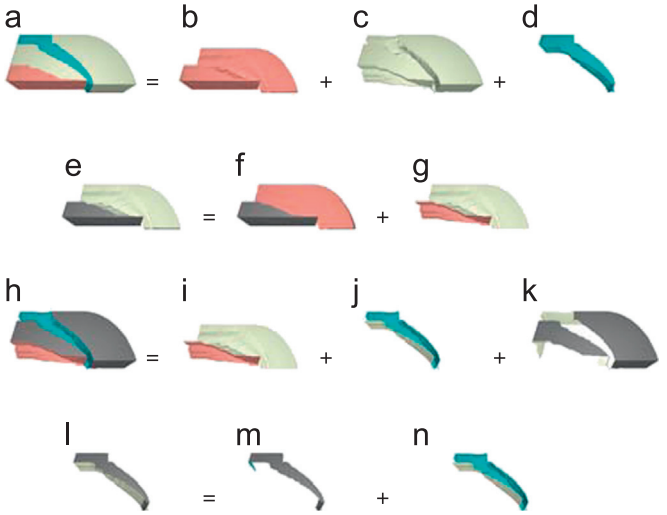


Fig. 11. The BluntFin (a) contains three materials, which are rendered in maroon (b), olive (c) and teal (d). Each boundary surface is formed by interfaces. The proposed DA approach provides not only the boundary of each material but also information for the adjacent component. (a) $\mathcal{H}_{BluntFin}$, (b) Ω_m, (c) Ω_o, (d) Ω_t, (e) $\Omega_m^{'}$, (f) $\Gamma_{(-1,m)}$, (g) $\Gamma_{(m,o)}$, (h) $\Omega_o^{'}$, (i) $\Gamma_{(m,o)}$, (j) $\Gamma_{(o,t)}$, (k) $\Gamma_{(-1,o)}$, (l) $\Omega_t^{'}$, (m) $\Gamma_{(-1,t)}$, and (n) $\Gamma_{(o,t)}$. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

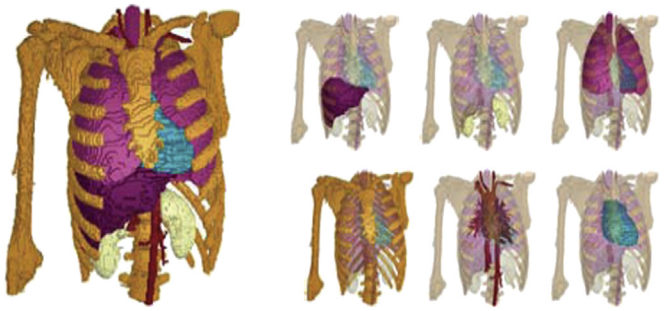


Fig. 12. The Utah Torso model contains six organs, *i.e.*, the liver, kidney, lung, bone, artery and heart.

and the geometric representation of the Schaedel model is also constructed in one pass.

The Schaedel* model in Table 2 is a downsampled version of the last dataset. The statistics of model complexity and runtime are depicted in Figs. 15 and 16, respectively. The runtimes of the approaches are the average of 10 runs. The BSHO method is performed on the GPU. The runtime of the BSHO for the Schaedel dataset is not included because the BSHO method requires too much memory to obtain results on a desktop graphics card. Although the BSHO method is the most memory- and time-consuming approach, it can generate high-quality boundary surfaces as shown in Fig. 13(b). The video memory malloc and the merging step (the synchronous operator) are the bottleneck of the BSHO method. The proposed DA approach exhibits performance comparable to that of the M3C method and can generate high-quality boundary surfaces comparable to those of the BSHO

Fig. 14. The Schaedel model contains 10 organs, *i.e.*, the skull, brain, eyes, clavicle, artery, teeth, respiratory system, C1, C2 and cervical vertebra. The boundary surfaces of the objects are constructed simultaneously.

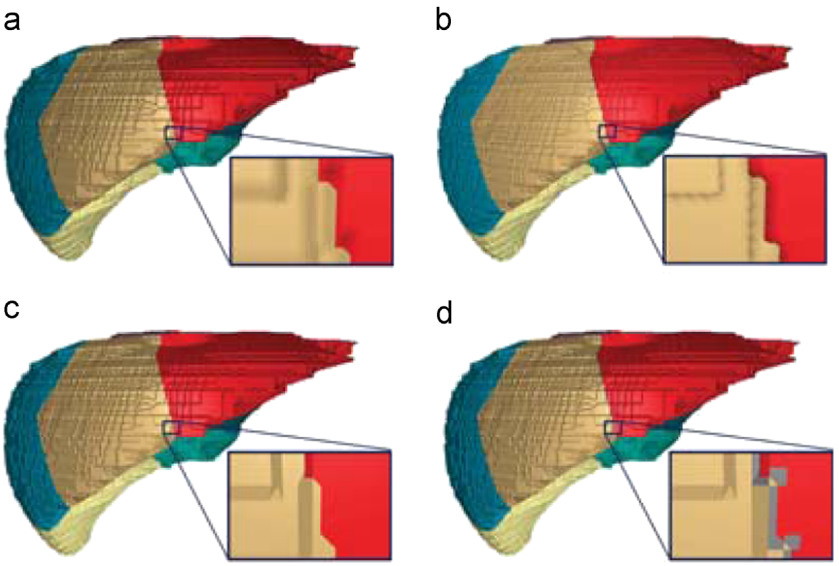
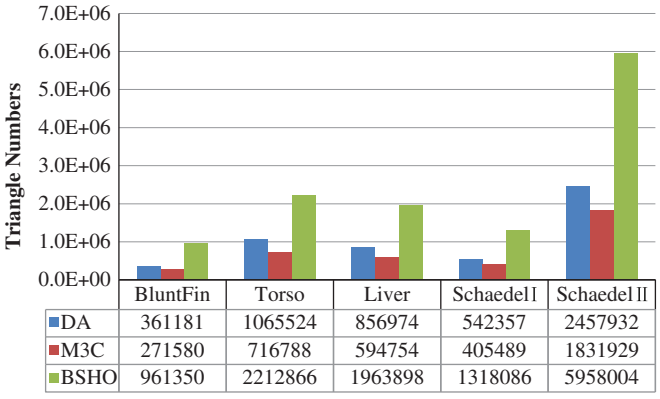


Fig. 13. All of the approaches reconstruct the liver with accurate geometry. However, the original M3C method generates the model with some visual defects because the orientations of the resulting triangles are not consistent. (a) DA, (b) BSHO, (c) Improved M3C, and (d) Original M3C.

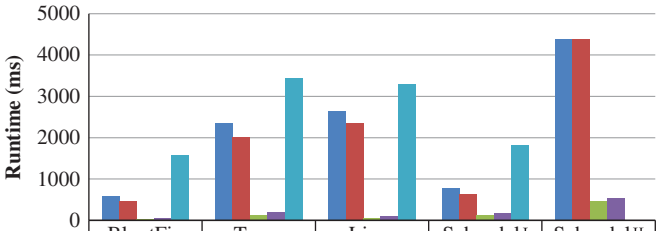
Table 2
The details of datasets.

Dataset	Materials	Data size
BluntFin	3	256 × 128 × 64
Torso	6	199 × 199 × 249
Liver	9	299 × 209 × 175
Schaedel*	10	128 × 128 × 83
Schaedel	10	256 × 256 × 166



	BluntFin	Torso	Liver	Schaedel I	Schaedel II
DA	361181	1065524	856974	542357	2457932
M3C	271580	716788	594754	405489	1831929
BSHO	961350	2212866	1963898	1318086	5958004

Fig. 15. Comparison of the proposed DA, BSHO, and M3C approaches regarding the number of resulting triangles.



	BluntFin	Torso	Liver	Schaedel I	Schaedel II
DA	588.54	2339.36	2649.39	777.45	4383.06
M3C	463.42	2006.38	2346.74	625.05	4375.71
DA*	25.82	135.27	54.45	116.99	454.78
M3C*	48.75	205.41	97.14	171.56	548
BSHO	1592.557	3441.461	3291.703	1818.416	

Fig. 16. Comparison of the runtime for the various approaches.

method. As shown in Fig. 16, the acceleration strategy is effective for both the DA and M3C approaches. Because the interfaces of the liver model contain a comparable number of triangles, both DA* and M3C* obtain a maximum acceleration effect on the liver model. Moreover, the DA* approach proceeds at least 17% faster than the M3C* method. As shown in Fig. 15, because of the difference in primitive voxels, the number of triangles generated by the proposed DA approach is less than one-half of that of the BSHO method (tri-prism (vs.) tetrahedron), but slightly more than that of the M3C method (tri-prism (vs.) cube).

6. Conclusion

In this paper, we proposed a new approach for extracting the boundary surfaces of a heterogeneous volume. We performed our extraction using the tri-prisms. With the help of a carefully designed hierarchical data structure, the boundary surfaces are constructed in a DA manner *i.e.*, separating points, separating line segments, loops, interfaces and boundary surfaces. The information regarding

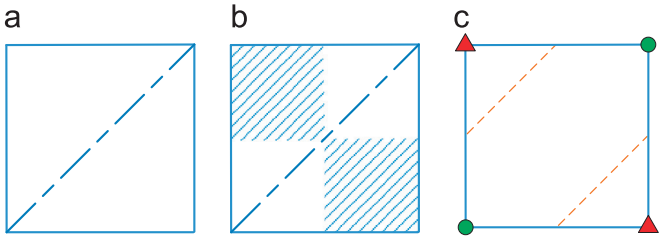


Fig. 17. 2D illustration of the inherent flaw of the proposed DA approach: (a) logical triangular regions; (b) the material distribution; (c) the extracted boundary.

topological intersections among different boundary surfaces is provided explicitly. The proposed DA approach can be verified because the separating patterns of a heterogeneous tri-prism can be exhaustively enumerated by the proposed MOIEM. Furthermore, the strategy for accelerating the DA approach is also effective for the M3C approach.

Compared with the BSHO method, the proposed DA approach has a clear advantage in performance. The M3C technique was improved upon to eliminate the visual artifacts of the raw mesh. Both the DA* and M3C* approaches are 10 times faster than their trivial implementations. Moreover, the DA* approach proceeds at least 17% faster than the M3C* method.

Although it is efficient, the proposed DA approach has an inherent flaw. Let us demonstrate this flaw with a 2D illustration. The boundaries are extracted from the two logical triangular regions, as shown in Fig. 17(a). For the material distribution shown in Fig. 17(b), the boundary construction is related to the manner in which a quadrilateral is subdivided into two logical triangular regions. We name this flaw as "directional bias".

In the future, we will combine the proposed DA approach with a volume rendering technique. A conventional direct volume rendering technique produces a volume classification from volume data, which can be regarded as the input of the proposed DA algorithm [26]. In this sense, the DA approach can be naturally integrated with direct volume rendering.

Acknowledgments

We would like to thank Dr. Yuanmin Cui, Dr. Yuncen Huang, Dr. Yizhi Tang, Dr. Qing Ran, Dr. Wenjun Du and LinLing Zhou for their valuable comments and suggestions. This work is supported by the National Natural Science Foundation of China under Grant nos. 60933007, 61170138, the Program for New Century Excellent Talents in University under Grant no. NCET-10-0728.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cag.2013.10.034.

References

[1] Wang M, Feng J. 2D-manifold boundary surfaces extraction from heterogeneous object on GPU. J Comput Sci Technol 2012;27(4):862–71.
[2] Wang CCL. Direct extraction of surface meshes from implicitly represented heterogeneous volumes. Comput Aided Des 2007;39(1):35–50.
[3] Xu P, Yang Q, Meng X. Marching prisms: rapid section generation in 3D geological model. In: Second international conference on computer modeling and simulation (ICCMS) '10, vol. 2. Sanya, Hainan; 2010. p. 528–32.
[4] Wyvill G, McPheeters C, Wyvill B. Data structure for soft objects. Vis Comput 1986;2:227–34.
[5] Lorensen WE, Cline HE. Marching cubes: a high resolution 3D surface construction algorithm. In: SIGGRAPH '87: proceedings of the 14th annual conference on computer graphics and interactive techniques. New York, NY, USA: ACM; 1987. p. 163–9.

[6] Newman TS, Yi H. A survey of the marching cubes algorithm. Comput Graph 2006;30(5):854–79.
[7] Bloomenthal J. Polygonization of implicit surfaces. Comput Aided Geometric Des 1988;5:341–55.
[8] Nielson G. On marching cubes. In: IEEE transactions on visualization and computer graphics, vol. 9; 2003.
[9] Rosenthal P, Linsen L. Direct isosurface extraction from scattered volume data. In: Eurographics/ IEEE-VGTC symposium on visualization; 2006. p. 99–106.
[10] Poston T, Nguyen HT, Heng P-A, Wong T-T. "Skeleton climbing": fast isosurfaces with fewer triangles. In: Proceedings of the 5th pacific conference on computer graphics and applications, PG '97. Washington, DC, USA: IEEE Computer Society; 1997. p. 117–26.
[11] Poston T, Wong T-T, Heng P-A. Multiresolution isosurface extraction with adaptive skeleton climbing. Comput Graph Forum 1998;17:137–48.
[12] Fujishiro I, Maeda Y, Sato H. Interval volume: a solid fitting technique for volumetric data display and analysis. In: Proceedings of the 6th conference on visualization '95, Washington, DC, USA; 1995. p. 151–8.
[13] Fujishiro I, Maeda Y, Sato H, Takeshima. Volumetric data exploration using interval volume. IEEE Trans Visualization Comput Graph 1996;2(2):144–55.
[14] Nielson GM, Franke R. Computing the separating surface for segmented data. In: VIS '97: proceedings of the 8th conference on visualization '97. Los Alamitos, CA, USA: IEEE Computer Society Press; 1997. p. 229–33.
[15] Wu Z, Sullivan JM. Multiple material marching cubes algorithm. Int J Numer Methods Eng 2003;58(2):189–207.
[16] Feng P, Ju T, Warren J. Piecewise tri-linear contouring for multi-material volumes. In: Advances in geometric modeling and processing, vol. 6130; 2010. p. 43–56.
[17] Dey TK, Janoos F, Levine JA. Meshing interfaces of multi-label data with Delaunay refinement. Eng Comput 2012;28(1):71–82.
[18] Cheng S-W, Dey TK, Ramos EA. Delaunay refinement for piecewise smooth complexes. Discrete Comput Geom 2009;43(1):121–66.
[19] Kou X, Tan S. Heterogeneous object modeling: a review. Comput Aided Des 2007;39:284–301.
[20] Wang CCL. Computing on rays: A parallel approach for surface mesh modeling from multi-material volumetric data. Comput Ind 2011;62:660–71.
[21] Shammaa MH, Suzuki H, Ohtake Y. Extraction of isosurfaces from multi-material CT volumetric data of mechanical parts. In: SPM '08: proceedings of the 2008 ACM symposium on solid and physical modeling. NY, USA: ACM; 2008. p. 213–20.
[22] Durst M. Letters: additional reference to "marching cubes". ACM Comput Graph 1988;22(4):72–3.
[23] Hung C-M, Buning PG. Simulation of blunt-fin-induced shock-wave and turbulent boundary-layer interaction. J Fluid Mech 1985;154:163–85.
[24] MacLeod R, Johnson C, Ershler P. Construction of an inhomegeneous model of the human torso for use in computational electrocardiography. In: IEEE engineering in medicine and biology society 13th annual international conference, vol. 13; 1991. p. 688–9.
[25] Parker SG, Johnson CR. Scirun: a scientific programming environment for computational steering. In: Proceedings of the 1995 ACM/IEEE conference on supercomputing (CDROM). New York, NY, USA: ACM; 1995.
[26] Chen W, Lu A, Ebert DS. Shape-aware volume illustration. Comput Graph Forum 2007;26(3):705–14.