Topology Preserving Tetrahedral Decomposition Applied To Trilinear Interval Volume Tetrahedrization

Bong-Soo Sohn
School of Computer Science and Engineering, Chung-Ang University
Seoul, Korea
[e-mail: bongbong@cau.ac.kr]

Received September 22, 2009; revised November 13, 2009; accepted November 16, 2009; published December 31, 2009

Abstract

We describe a method to decompose a cube with trilinear interpolation into a collection of tetrahedra with linear interpolation, where the isosurface topology is preserved for all isovalues during decomposition. Visualization algorithms that require input scalar data to be defined on a tetrahedral grid can utilize our method to process 3D rectilinear data with topological correctness. As one of many possible examples, we apply the decomposition method to topologically accurate tetrahedral mesh extraction of an interval volume from trilinear volumetric imaging data. The topological correctness of the resulting mesh can be critical for accurate simulation and visualization.

Keywords: Graphics, visualization, isosurface, interval volume, mesh extraction

A demo program is downloadable from http://cau.ac.kr/~bongbong/decomp09. A preliminary version of this paper was presented in the proceedings of the International Conference on Computational Sciences (ICCS), pages 350-357, 2007. This version includes a complete implementation of trilinear interval volume tetrahedrization and its experimental results. The author would like to thank Prof. Bajaj for his insightful advice and guidance. This research was supported by the Chung-Ang University Research Grants in 2008.

DOI: 10.3837/tiis.2009.06.006

1. Introduction

Scientific simulations and measurements often generate real-valued volumetric data in the form of function values sampled on a three dimensional (3D) rectilinear grid. Trilinear interpolation is a common way to define a function inside each cell. It is computationally simple and provides a good estimation of a function between sampled points in practice. *Isosurface extraction* is one of the most popular techniques for visualizing the volumetric data. An isosurface is a level set surface defined as $I(w) = \{(x, y, z) \mid F(x, y, z) = w\}$, where F is a function defined from the data and w is an isovalue. The isosurface I is often polygonized for modeling and rendering purposes. We call I a *trilinear isosurface* to distinguish it from a polygonized isosurface when F is a piecewise trilinear function. Similarly, *(trilinear) interval volume*, $V(\alpha, \beta) = \{(x, y, z) \mid \alpha \le F(x, y, z) \le \beta\}$, is defined as a subvolume enclosed by two boundary isosurfaces with isovalues α and β.

Although the rectilinear volumetric data is the most common form, many visualization techniques [1][2][3][4][5][6] require a tetrahedral grid domain due to its simplicity. People generally decompose each cube into a set of tetrahedra where a function is defined by linear interpolation, to apply such techniques to rectilinear volumetric data. The decomposition may significantly distort the original trilinear function in terms of its isosurface topology and geometry. (See [7] for examples.) 2D/3D meshes with undesirable topologies extracted from the distorted function may cause a serious problem of inaccuracy in various simulations, such as Boundary Element Method (BEM) and Finite Element Method (FEM), when the extracted meshes are used as geometric domains for the simulation [6].

The main contributions of this paper are :

- topology preserving tetrahedral decomposition of a cube with trilinear interpolation
- application to trilinear interval volume tetrahedrization

We describe a rule for tetrahedral decomposition of a cube where isosurface topology in the cube is preserved for all isovalues during the decomposition. The main idea of our method is to insert saddle points and connect them to the corner vertices of a cube to generate tetrahedra. In case there is no saddle point, we perform the conventional 6-fold decomposition without inserting additional vertices. The collection of tetrahedra involves a minimal set of vertices that can correctly capture the level set topology of a trilinear function because the level set topology changes only at critical points.

The typical usage of this method is to convert a rectilinear domain of volumetric data into a tetrahedral grid form with topology preservation. Since our method maintains consistent topology during the data conversion, it significantly improves the accuracy of modeling and visualization. Various visualization algorithms that require input scalar data to be defined on a tetrahedral grid can utilize this method to process 3D rectilinear data with topological correctness.

As an example, we apply our method to topologically accurate interval volume tetrahedrization from trilinear volumetric data. We use a well known method [5] to tetrahedrize an interval volume within a tetrahedral cell based on a pre-defined lookup table. For dealing with a rectilinear data, a 5-fold or 6-fold decomposition has been mostly used to convert it into a tetrahedral grid form, which might cause topologically inaccurate results. Since our decomposition method preserves the topology of all the isosurfaces inside a cube, adopting our method instead of the 5-fold or 6-fold decomposition generates a topologically correct interval volume. The topological correctness can be critical for accurate simulation (e.g. FEM), when the resulting meshes are used as geometric domains for the simulation. Besides correct topology, mesh quality that is also an important factor for accurate simulation can be improved using quality improvement methods [8][9].

The remainder of this paper is organized as follows. In section 2, we review related work on tetrahedral decomposition and trilinear isosurfaces. In section 3, we explain the trilinear isosurface and its topology determination. In section 4, we describe topology preserving tetrahedral decomposition of a trilinear cell. In section 5, we apply tetrahedral decomposition methods to trilinear interval volume tetrahedrization. Section 6 reports experimental results. Finally, we conclude this paper in section 7.

2. Related Work

A cube can be decomposed into five or six tetrahedra without inserting additional points, or even more tetrahedra with insertion of cube and face centroids [10]. Carr et al. [7] reported a visualization result and visual artifacts for each decomposition method. Schroeder et al. [11] described a consistent tetrahedral decomposition of cells in any arbitrary adaptive grid. The simplicity of tetrahedral grids generated from those methods was utilized by various visualization applications, such as contour tree construction [2], isosurface triangulation [12], interval volume tetrahedrization [5], volume rendering [13], and particle tracing [14]. However, none of those methods consider preservation of the isosurface topology in a trilinear function during the decomposition.

A trilinear isosurface inside a cube can have complex topology and geometry that requires sophisticated procedures for correct reconstruction. Marching Cubes (MC) [10] is the most popular method to extract a piecewise linear approximation of an isosurface. It visits each cubical cell and performs local triangulation depending on a sign configuration of eight vertices of the cell[1]. Some of the sign configurations have ambiguity in triangulation and may generate an inconsistent triangular mesh with a crack on an isosurface. The ambiguity of contour connectivity on a face can be resolved by taking saddle points, called *face saddle* of the bilinear function on the face and comparing the saddle value to an isovalue [15]. This method generates a consistent triangular mesh of an isosurface but the topology of the extracted isosurface may not be correct due to a topological ambiguity inside a cell. Natarajan [16] used a saddle point inside a cell, called *body saddle*, in a similar manner to resolve the internal ambiguity and compute the correct topology of a trilinear isosurface in the cell. Cignoni et al. [17] applied adaptive refinement of the triangular isosurface mesh generated using Natarajan's method to guarantee user defined geometric accuracy. Chernyaev [18] and Nielson [19] listed every possible trilinear isosurface topology with corresponding conditions. Lopes and Brodlie [20] provided accurate triangulation for each case. These topologically correct isosurfacing methods cannot be directly applied to interval volume tetrahedrization due to the high complexity of the tetrahedrization for a cube. Fujishiro et al. [21] used a set operation to compute polyhedral solid approximation of an interval volume for each cube.

[1] The *sign* of a vertex is positive if the function value on the vertex is greater than an isovalue, and negative in the other case.

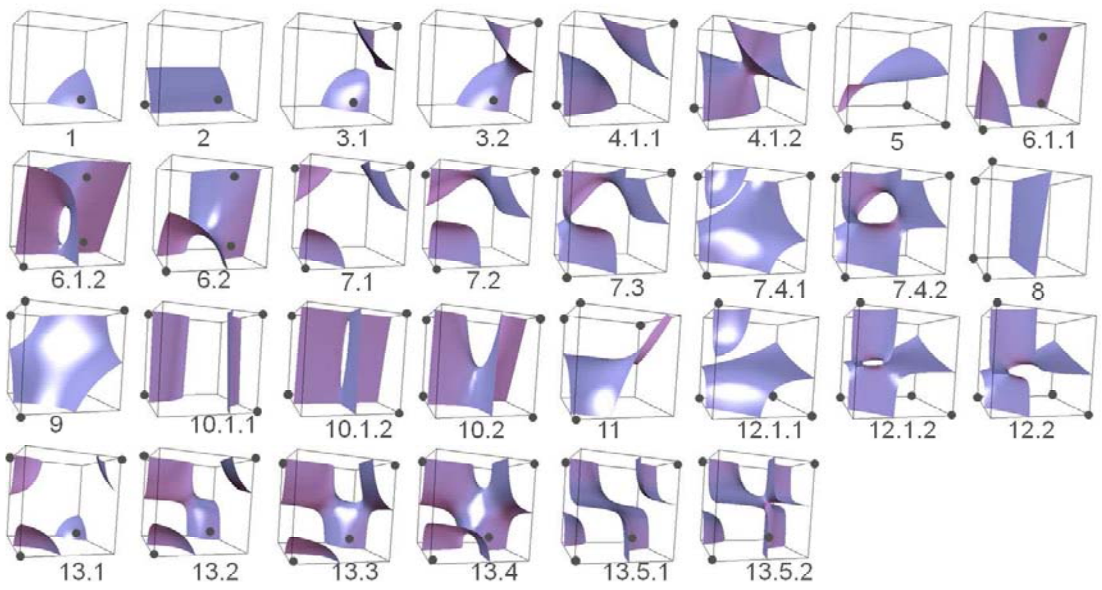


Fig. 1. Possible isosurface topology of trilinear interpolant [20]

3. Trilinear Isosurface Topology

The function inside a cube, F^c, is constructed by trilinear interpolation of values on eight vertices of the cube.

$$\begin{aligned} F^c(x,y,z) = {} & F_{000}(1-x)(1-y)(1-z) + F_{001}(1-x)(1-y)z \\ & + F_{010}(1-x)y(1-z) + F_{011}(1-x)yz \\ & + F_{100}x(1-y)(1-z) + F_{101}x(1-y)z \\ & + F_{110}xy(1-z) + F_{111}xyz \end{aligned}$$

This means the function , F^f , on any face of a cube is a bilinear function computed from four vertices of the face.

$$F^f(x,y) = F_{00}(1-x)(1-y) + F_{01}(1-x)y + F_{10}x(1-y) + F_{11}xy$$

Saddle points, where their first partial derivative for each direction is zero, play important roles in determining the correct topology of a trilinear isosurface inside a cube. The result of computing the location of *face* and *body saddles* that satisfy $F_x^f = F_y^f = 0$ and $F_x^c = F_y^c = F_z^c = 0$ respectively is described in [20]. Saddle points outside the cube are ignored.

It is well known that some of the sign configurations of the eight vertices in a cube have ambiguities in determining contour connectivity. The papers [20] and [16] show that additional sign configurations of the face and body saddle points can disambiguate the correct topology of a trilinear isosurface. **Fig. 2** illustrates a 2D example of the ambiguity problem and its disambiguation through triangular decomposition based on a saddle point. **Fig. 1** shows every possible isosurface connectivity of a trilinear function where symmetric cases are ignored [20]. The correctness of our topology preserving tetrahedral decomposition method is proved by showing that an isosurface defined in the tetrahedral decomposition and a trilinear isosurface inside a cube are topologically equivalent for any sign configurations of saddle points and the eight corner vertices of the cube. The overall idea is presented in **Fig. 3** by showing the topology preserving decomposition rule for the 2D case which is much simpler than 3D case.

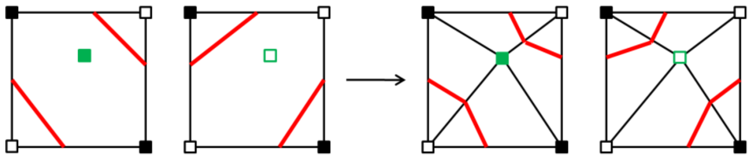


Fig. 2. Triangular decomposition of a face, connecting a face saddle to each edge of the face, resolves an ambiguity in determining contour connectivity and allows correct contour reconstruction.

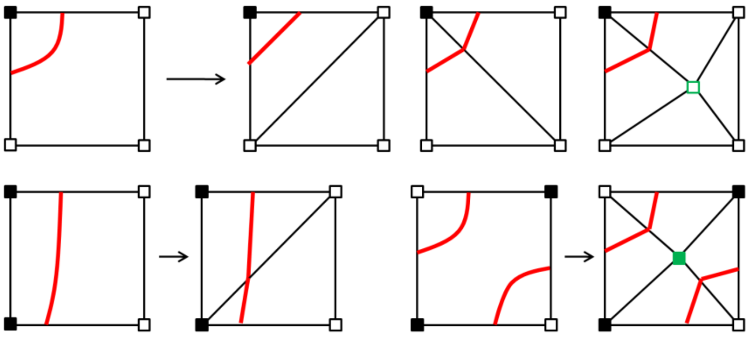


Fig. 3. Triangular subdivision of a rectangle preserves the isosurface topology of a bilinear function for all possible cases.

4. Topology Preserving Tetrahedral Decomposition (TPTD)

We describe a method to decompose a cube with trilinear interpolation into a collection of tetrahedra with linear interpolation where isosurface topology is preserved for all isovalues during the decomposition. We call this method TPTD. The tetrahedral decomposition is consistent for the entire rectilinear volumetric data in the sense that the tetrahedra are seamlessly matched on a face between any two adjacent cubes.

The main idea of TPTD is to insert the saddle points and connect them to the corner vertices of a cube in order to generate tetrahedra in the same manner as the 2D case shown in **Fig. 3**. Since the isosurface topology changes at critical points, the saddle points must be included in TPTD if there are any inside the cube. TPTD for a 2D rectangle is simple because there are only two possibilities, either a face saddle exists on the rectangle or it does not. However, TPTD for a 3D cube is extremely complex compared to the 2D case because there are two different kinds of saddles, face and body, and there can be zero, one, or two of the latter. For dealing with the high complexity, we separate the different cases based on the analysis of the face and body saddles inside a cube and provide a consistent decomposition rule for each case.

Let s_b and s_f be the number of body saddles and face saddles respectively. There are five cases based on the number of face saddles and body saddles, (i) $s_b = 0$ and $s_f = 0$, (ii) $s_b = 0$ and $1 \le s_f \le 4$, (iii) $s_b = 1$ and $0 \le s_f \le 4$, (iv) $s_b = 0$ and $s_f = 6$, and (v) $1 \le s_b \le 2$ and $s_f = 6$. The most complicated case occurs when there are six face saddles. It requires special treatment. We deal with the case of $s_b = 0$ and $1 \le s_b \le 2$, separately. Note that the number of face saddles cannot be five. The number of body saddles cannot be two, unless the number of face saddles is six. The decomposition rule for each case is as follows :

- case (i) : Decompose a cube into six tetrahedra (6-fold decomposition [7]).
- case (ii) : Choose one face saddle and decompose a cube into five pyramids by connecting the face saddle to the four corner vertices of each face except the one that contains the face saddle. If the rectangular face of a pyramid contains a face saddle, the face is decomposed into four triangles that share the face saddle and the pyramid is decomposed into four tetrahedra. If the face of the pyramid does not contain a face saddle, the pyramid is decomposed into two tetrahedra in a consistent manner. If the number of face saddles is three or four, we need to choose the second biggest face saddle.
- case (iii) : Decompose a cube into six pyramids by connecting a body saddle to the four corner vertices of each face of a cube. As is the case in (ii), if the rectangular face of a pyramid contains a face saddle, the face is decomposed into four triangles and the pyramid is decomposed into four tetrahedra. Otherwise, the pyramid is decomposed into two tetrahedra.
- case (iv) : A diamond is created by connecting the six face saddles. The diamond is decomposed into four tetrahedra. Twelve tetrahedra are created by connecting the two vertices of each of the twelve edges in a cube and the two face saddles on the two faces which share the edge. Eight tetrahedra are created by connecting each of the eight faces in the diamond and a corresponding corner vertex of the cube. This will decompose a cube into twenty four tetrahedra. (**Fig. 4** (c))
- case (v) : **Fig. 5** (a) and (b) show the cell decomposition when there are two body saddles and six face saddles. It generates two pyramids and four prisms, where pyramids and prisms are further decomposed into tetrahedra (**Fig. 5** (c)). We choose any two parallel faces that are connected to body saddles to form pyramids. We classify saddle points as three *small face saddles*, a *small body saddle*, a *big body saddle*, and three *big face saddles* based on increasing order of the saddle values. Let the *small/big corner vertex* be the vertex adjacent to the three faces with small/big face saddles. The two parallel faces with the small and big face saddles are connected to small and big body saddles respectively to form the two pyramids. The four prisms are decomposed into tetrahedra such that the small corner vertex should not be connected to the big body saddle and the big corner vertex should not be connected to the small body saddle. Two types of decomposition of a prism are possible that satisfy this constraint as shown in **Fig. 5** (c). In case $s_b = 1$, we consider as if a small or big body saddle moves to a face saddle of a pyramid that is connected to the body saddle and hence the pyramid is collapsed. In this case, the pair of parallel faces for forming the pyramids is chosen such that the face saddle of the collapsed pyramid should not be the smallest or the biggest one

Fig. 4 shows several examples of applying theTPTD rule to a cube with a different number of body and face saddles. In the Appendix, we give a brief proof of topology preservation during TPTD.

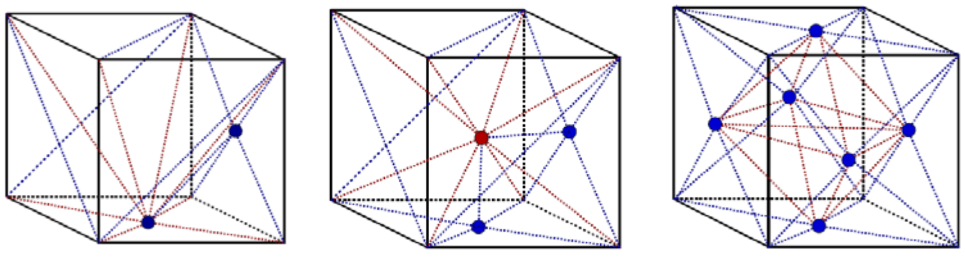
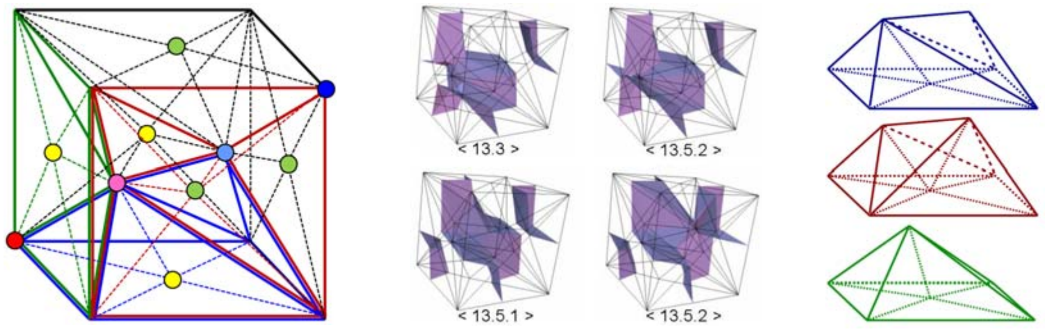

(a) $s_b = 0$, $s_f = 2$ (b) $s_b = 1$, $s_f = 2$ (c) $s_b = 0$, $s_f = 6$

Fig. 4. Example of TPTD for different cases



(a) (b) (c)

Fig. 5. (a) Cell decomposition when two body saddles exist. Dark blue, green, light blue, magenta, brown, and red circles represent small corner vertex, small face and body saddles, big body and face saddles, and big corner vertex, respectively. (b) Isosurface topology for different isovalues in the case of (a). (c) Tetrahedral decomposition of pyramids and prisms.

5. Trilinear Interval Volume Tetrahedrization

5.1 Basic Method Using TPTD

Interval volume [21], $V(\alpha, \beta) = \{(x, y, z) \mid \alpha \leq F(x, y, z) \leq \beta\}$, is a generalized form of an isosurface that is defined as a subvolume between two boundary isosurfaces. Tetrahedral meshes that approximate an interval volume are widely used for scientific simulations, such as FEMethod [9], as well as for visualization [21]. Topologically and geometrically accurate tetrahedral meshing is crucial for accurate simulation and visual exploration of the data.

Nielson and Sung [5] constructed a lookup table to extract tetrahedral meshes for an interval volume in a tetrahedron. If data are defined on a 3D rectilinear grid, they convert the data into a tetrahedral form by performing a decomposition method that generates five tetrahedra from each cube. By using TPTD instead of the 5-fold decomposition, we can extract a topologically correct tetrahedral mesh of a trilinear interval volume.

Fig. 7 shows that even the most complex interval volume inside a cube can be correctly tetrahedrized. Two boundary trilinear isosurfaces are visualized. After applying tetrahedrization for each tetrahedral cell generated by TPTD, we could extract topologically correct tetrahedral meshes for the interval volume.

5.2 Efficiency Enhancement Using Simplified Tetrahedral Decomposition

We can enhance the efficiency using a simplified tetrahedral decomposition (STD) method when a cubic cell contains one of no boundary isosurfaces of an interval volume. STD generates a smaller number of tetrahedra from a cube than TPTD does. For example, if a cube is completely inside an interval volume, STD generates six tetrahedra, whilst TPTD may generate between six and thirty tetrahedra depending on the sign configurations of the vertices in the cube. Unlike TPTD, STD is designed to preserve isosurface topology for one selected isovalue during decomposition. Therefore, performing interval volume tetrahedrization from STD of a cube gives a topologically correct result if the cubic cell contains at most one boundary isosurface. In this section, we describe STD and a means to combine STD and TPTD for efficient and topologically correct tetrahedrization of a general interval volume.

Simplified Tetrahedral Decomposition (STD) The goal of STD is to provide tetrahedral decomposition of a cube, where isosurface topology is preserved for one selected isovalue. This is done by removing facial and internal ambiguities for isosurface reconstruction. A facial ambiguity is resolved by decomposing a face into four triangles that share a face saddle point. Likewise, internal ambiguity is resolved by decomposing a cube into six pyramids that share a body saddle point. If there is an internal ambiguity, and the isosurface does not contain a tunnel shape (a.k.a. neck), a body saddle point is not necessary in the cell decomposition for correct reconstruction.

There are four cases : (i) no face ambiguity and no tunnel, (ii) face ambiguity (less than six) and no tunnel, (iii) a tunnel, and (iv) six face ambiguities and no tunnel.

- case (i) : perform 6-fold decomposition.
- case (ii) : choose a face saddle on a face with ambiguity[2]. Next, decompose a cube into five pyramids by connecting the face saddle into the four corner vertices of each face, except the face that contains the face saddle.
- case (iii) : decompose a cube into six pyramids by connecting a body saddle that is involved with a tunnel to the four corner vertices of each face of the cube.
- case (iv) : perform the same decomposition as for case (iv) in section 4. (13(a) in **Fig. 6**)

If the rectangular face of a pyramid generated from case (ii) and (iii) has a face ambiguity, the face is decomposed into four triangles that share the face saddle so that the pyramid is decomposed into four tetrahedra. Otherwise, the face is decomposed into two triangles by connecting the two diagonal vertices so that the pyramid is decomposed into two tetrahedra.

Isosurface configurations 1, 2, 5, 8, 9, and 11 in **Fig. 1** are the only possible cases that do not have an ambiguity in a cube. 6-fold decomposition is sufficient for such cases. The configuration 4.1.1 is an exceptional case that has an internal ambiguity with no face ambiguity and no tunnel. This case can be handled by type-4(a) decomposition in **Fig. 6**.

We applied STD to each case in **Fig. 1**, and extracted a triangular isosurface. **Fig. 6** shows the cell decomposition and its triangulation for every possible configuration of a trilinear isosurface. The comparison of **Fig. 1** and **Fig. 6** confirms that the triangular isosurface generated from STD is topologically equivalent to a trilinear isosurface.

[2] If there are three faces with ambiguities, we need to choose the third ambiguous face where the order is based on the function value of the face saddle. This is necessary to correctly reconstruct configuration 7.2 and 7.3.

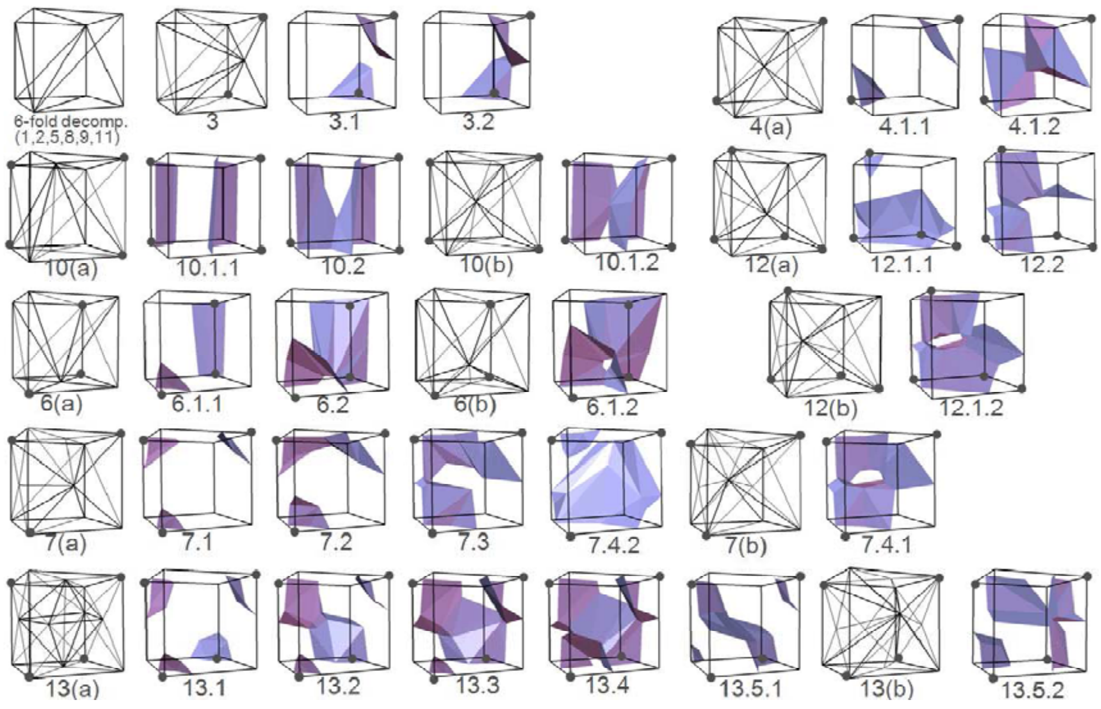


Fig. 6. Simplified tetrahedral decomposition (STD) and triangulation for each case.

Efficient Tetrahedrization of Trilinear Interval Volume By Combining STD and TPTD (CTD) STD preserves the topology of only one boundary isosurface. If there are two boundary isosurfaces of an interval volume inside a cube, tetrahedral meshes extracted from the decomposition may not be topologically correct. For efficient and topologically correct reconstruction of the general interval volume in each cube, we apply TPTD to a cube that contains two boundary isosurfaces and apply STD to a cube that contains one or no boundary isosurfaces. However, the mesh may not be consistent on a face between two neighboring cubes, where one cube is decomposed using TPTD and the other is decomposed using STD. Note that this inconsistency occurs only when STD contains two triangles and TPTD contains four triangles on the shared face, respectively. In such a case, we choose a pyramid of STD that contains the shared face and replace the pyramid with four tetrahedra that can be generated by connecting the top point of the pyramid to each of the four triangles on the face. We term this method CTD, which combines STD and TPTD in a consistent manner. CTD coupled with the Nielson and Sung's method guarantees to generate topologically correct and consistent tetrahedral meshes for a trilinear interval volume.

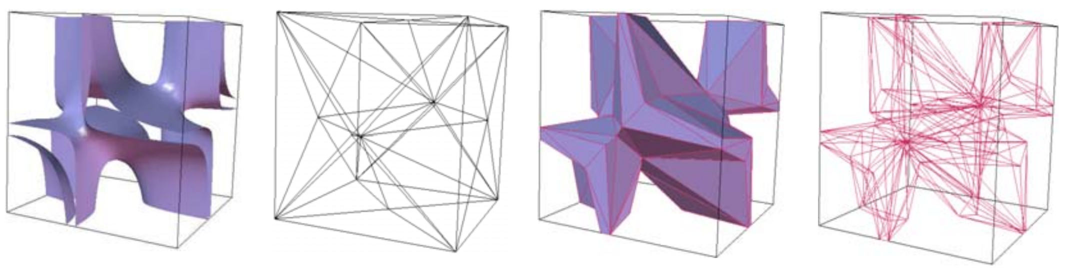
6. Results

We implemented the TPTD, STD, CTD and 6-fold decomposition methods. We used them to convert the rectilinear grid of volumetric data into a tetrahedral grid. Applying Nielson and Sung's method [5] to each tetrahedron of the data, we extracted tetrahedral meshes that approximate an interval volume. The extracted meshes were topologically equivalent to the ideal trilinear interval volume when TPTD or CTD was applied. The result of STD is equivalent to that of CTD unless there is a cubic cell that contains two boundary isosurfaces. We measured the performance of each method by counting the number of tetrahedra for the extracted interval volume and the time for decomposition and interval volume tetrahedrization. We tested several volumetric data sets defined on rectilinear grids, a signed distance function (SDF) dataset and real datasets generated by Magnetic Resonance Imaging (MRI) and Computed Tomography Angiography (CTA). Since the original MRI dataset was too large, we tested two subregions, tumor and breast. The experimental results are summarized in **Table 1**. Our results were computed on a desktop PC equipped with an Intel Pentium Dual Core CPU (2.40GHz) and 2GB main memory.

As shown in **Fig. 8** (b) and (c), TPTD correctly reconstructs trilinear interval volume, whilst the result applying 6-fold decomposition has topological artifacts. **Fig. 8** (d) and **Fig. 9** (d) show the TPTD and CTD correctly reconstruct a very thin interval volume. CTD is generally more efficient than TPTD in terms of the resulting interval volume mesh size, whilst the accuracy is equivalent to that of TPTD. However, the usage of CTD is limited to trilinear interval volume tetrahedrization. TPTD can be applied to general visualization techniques that require tetrahedral decomposition. There are only small performance differences in the results of SDF(knee) and MRI(tumor) because the data sets are topologically simple. On the other hand, the MRI(breast) data is reasonably complex, so the result clearly shows the performance difference. It seems that the timing results do not have significant differences among the methods. The markd regions of **Fig. 9** (b) and (c) show that TPTD generates more mesh elements without achieving better geometric or topological accuracy than CTD.

Table 1. Results on the number of mesh elements and the time for decomposition and extraction.

Data	Resolution	α , β	TPTD	STD	CTD	6-fold
SDF	32x32x32	-4.5,-0.9	22862 (31ms)	22303 (31ms)	22303 (31ms)	22239 (32ms)
SDF(thin)	32x32x32	-2.0,-0.9	24442 (31ms)	24114 (31ms)	24242 (32ms)	24049 (32ms)
MRI(tumor)	32x32x32	80.1, ∞	12988 (16ms)	12092 (15ms)	12092 (16ms)	12092 (16ms)
MRI(breast)	169x171x90	30.1, ∞	6028469 (6218ms)	5448297 (6157ms)	5448297 (6490ms)	5384223 (5985ms)
MRI(thin)	32x32x32	80.1,89.5	15574 (16ms)	15058 (15ms)	15270 (15ms)	15058 (15ms)
CTA(aorta)	128x128x128	168, ∞	2002507 (2339ms)	1676844 (2118ms)	1676844 (2497ms)	1663861 (1975ms)



(a) Bounday isosurfaces (b) Cell Decomposition (c) Interval Volume (d) Wireframe

Fig. 7. Interval volume extraction for two boundary isosurfaces with complex geometry and topology inside a cube that has two body saddles and six face saddles. The tetrahedral meshes (c) extracted from TPTD (b) correctly approximate trilinear interval volume (a).

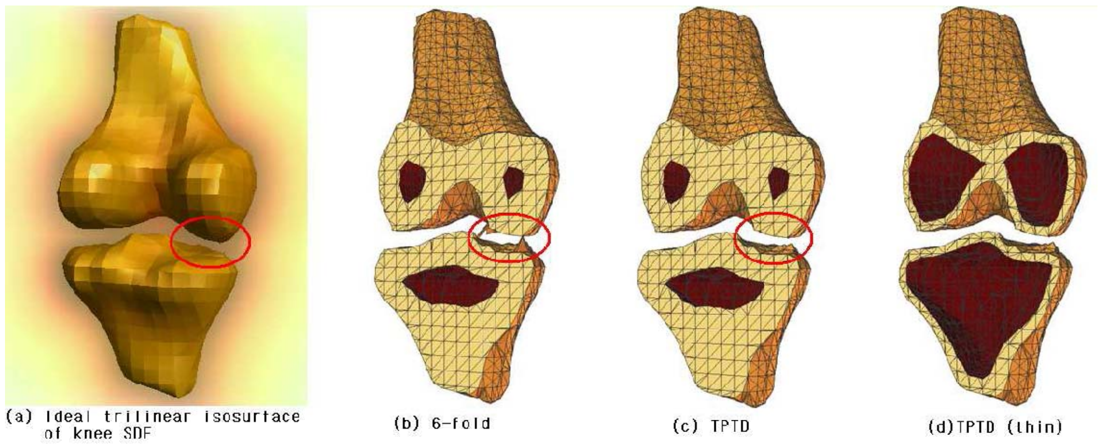


Fig. 8. Trilinear interval volume tetrahedrization of a knee SDF data. The comparison of regions marked with red circles shows that our method (TPTD) correctly reconstructs the boundary isosurfaces of a trilinear interval volume while 6-fold decomposition cannot.

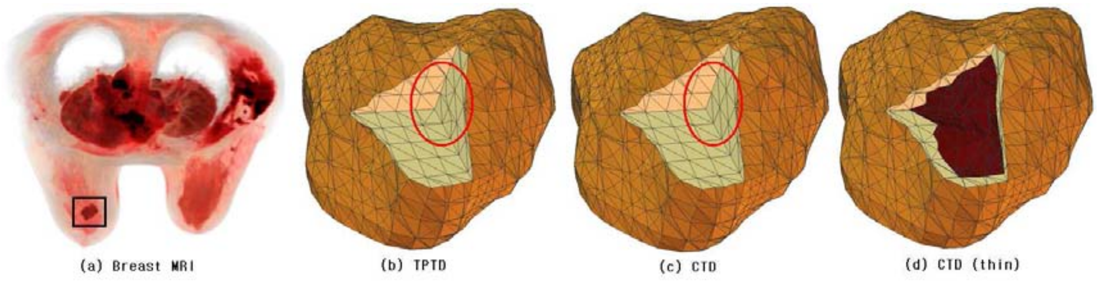


Fig. 9. Trilinear interval volume tetrahedrization of a breast lesion (MRI). The comparison of regions marked with red circles shows that CTD allows more efficient interval volume tetrahedrization than TPTD does while it is as accurate as TPTD even for extracting thin interval volume.

We also tested our method with the CTA dataset which is more complex than the other datasets. As shown in **Fig. 10**, the dataset contains a spine and an abdominal aorta, the boundary surfaces of which should be separated. Boundary isosurfaces of interval volumes extracted from (e) TPTD are topologically equivalent to (c) ideal trilinear isosurfaces. However, the boundary isosurfaces for spine and aorta are connected when (d) 6-fold decomposition is applied. The wrong topology in the extracted meshes may give rise to critical inaccuracy especially when used in a computional simulation.

Table 2 and **Table 3** provide an additional comparion between TPTD and CTD based on the number and relative ratio of cubic cells that belong to each case. We counted only the cells that intersected the interval volume. The result confirms that CTD is generally more efficient than TPTD in terms of the generated mesh size. For example, in CTD of CTA dataset, the number of cells that belong to case (i) is 182788 (99.5%), which means that almost all cubic cells are decomposed with the simple 6-fold method. On the other hand, the number of cells for TPTD is 132800 (72.3%) and TPTD requires more complex decompositions for 27.7% of cubic cells.

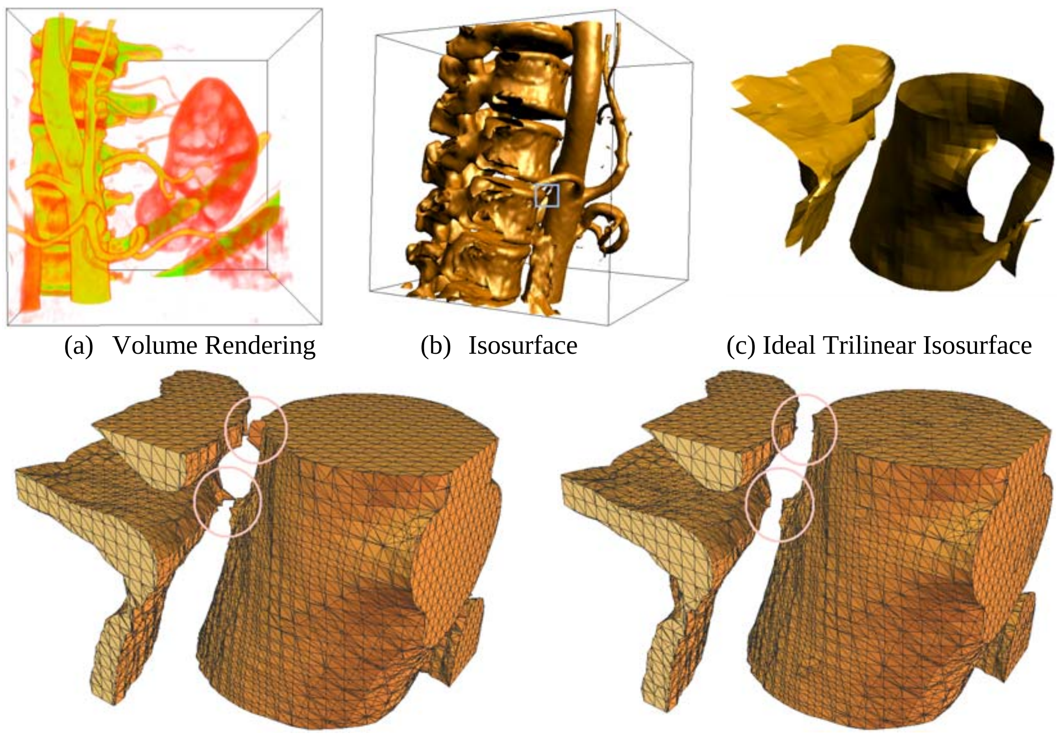


Fig. 10. Trilinear interval volume tetrahedrization of CT Angiography (CTA) data. (α =168, β $=\infty$). The region marked with a blue box in (b) is magnified in (c)~(e).

Table 2. The number and ratio of cubic cells for each case when TPTD is applied to each dataset. Only cells that intersected the interval volume specified by α and β were considered.

Data	α , β	**Number and Ratio of Cubic Cells for TPTD**					
		case (i)	case (ii)	case (iii)	case (iv)	case (v)	Total
SDF (32x32x32)	$-\infty$,-0.9	2055 (94.6%)	114 (5.2%)	3 (0.1%)	0 (0.0%)	0 (0.0%)	2172 (100%)
MRI (169x171x90)	30.1, ∞	514929 (83.9%)	88648 (14.4%)	10340 (1.7%)	1 (0.0%)	0 (0.0%)	613918 (100%)
CTA (128x128x128)	168, ∞	132800 (72.3%)	47214 (25.7%)	3559 (1.9%)	61 (0.03%)	80 (0.04%)	183714 (100%)

Table 3. The number and ratio of cubic cells for each case when CTD is applied to each dataset. Only cells that intersected the interval volume specified by α and β were considered.

Data	α , β	**CTD : Number and Ratio of Cubic Cells**				
		case (i)	case (ii)	case (iii)	case (iv)	Total
SDF (32x32x32)	$-\infty$,-0.9	2162 (99.5%)	10 (0.5%)	0 (0.0%)	0 (0.0%)	2172 (100%)
MRI (169x171x90)	30.1, ∞	609796 (99.3%)	3997 (0.7%)	125 (0.0%)	0 (0.0%)	613918 (100%)
CTA (128x128x128)	168, ∞	182788 (99.5%)	888 (0.5%)	38 (0.0%)	0 (0.0%)	183714 (100%)

7. Conclusion

We described a method for tetrahedral decomposition of a cube where isosurface topology is preserved for all isovalues. We applied our method to topologically correct tetrahedrization of an interval volume from trilinear volumetric data. We envision that many visualization algorithms, which can take only tetrahedral grid data, can utilize our method to deal with trilinear volumetric data, instead of using simple 5-fold or 6-fold tetrahedral decomposition methods that may significantly distort the isosurface topology and geometry.

References

[1] C. L. Bajaj, V. Pascucci, and D. R. Schikore. “The contour spectrum”. in *IEEE Visualization Conference*, pages 167–173, 1997.

[2] H. Carr, J. Snoeyink, and U. Axen. “Computing contour trees in all dimensions”. *Computational Geometry: Theory and Applications*, 24(2):75–94, 2003.

[3] H. Carr, J. Snoeyink, and M. van de Panne, “Flexible isosurfaces: Simplifying and displaying scalar topology using the contour tree”, *Computational Geometry: Theory and Applications*, 43(1), pages 42-58, 2010

[4] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci. “Morse complexes for piecewise linear 3-manifolds”. in *Proceeding of the 19-th ACM Symposium on Computational Geometry (SoCG).* Pages 361-370, 2003

[5] G. M. Nielson and J. Sung. “Interval volume tetrahedrization”. in *IEEE Visualization Conference*, pages 221–228, 1997.

[6] M. J. van Kreveld, R. van Oostrum, C. L. Bajaj, V. Pascucci, and D. Schikore. “Contour trees and small seed sets for isosurface traversal”. in *ACM Symposium on Computational Geometry*, pages 212–220, 1997.

[7] H. Carr, T. Moeller, and J. Snoeyink. “Artifacts caused by simplicial subdivision”. *IEEE Transactions on Visualization and Computer Graphics*, 12(2), pages 231–242, 2006.

[8] C. A. Dietrich, C.E. Scheidegger, J. Schreiner, J. Comba, L. P. Nedel, and C. T. Silva, “Edge Transformations for Improving Mesh Quality of Marching Cubes”, *IEEE Transactions on Visualization and Computer Graphics*, 15(1): 150-159, 2009

[9] J. Zhang, C. Bajaj, and B.-S. Sohn. “3D finite element meshing from imaging data”. in *the special issue of Computer Methods in Applied Mechanics and Engineering (CMAME) on Unstructured Mesh generation*, 194(48-49), 2005.

[10] G. Albertelli and R. A. Crawfis. “Efficient subdivision of finite-element datasets into consistent tetrahedral”. in *IEEE Visualizaton Conference*, pages 213–219, 1997.

[11] W. J. Schroeder, B. Geveci, and M.Malaterre. “Compatible triangulations of spatial decompositions”. in *IEEE Visualization Conference*, pages 211–218, 2004.

[12] P. Ning and J. Bloomenthal. “An evaluation of implicit surface tillers”. *IEEE Computer Graphics and Applications*, pages 33–41, 1993.

[13] P. Shirley and A. Tuchman. “A polygonal approximation to direct scalar volume rendering”. *Computer Graphics*, 24(5):63,70, 1990.

[14] D. N. Kenwright and D. A. Lane. “Interactive timedependent particle tracing using tetrahedral decomposition”. *IEEE Ttransactions on Visualization and Computer Graphics*, 2(2):120–129, 1996.

[15] G. M. Nielson and B. Hamman. “The asymptotic decider: resolving the ambiguity in marching cubes”. in *Proceedings of IEEE Visualization Conference*, pages 83–91, 1991.

[16] B. K. Natarajan. “On generating topologically consistent isosurfaces from uniform samples”. *The Visual Computer*, 11(1):52–62, 1994.

[17] P. Cignoni, F. Ganovelli, C. Montani, and R. Scopigno. “Reconstruction of topologically correct and adaptive trilinear isosurfaces”. *Computers and Graphics*, 24(3):399–418, 2000.

[18] E. V. Chernyaev. “Marching cubes 33 : Construction of topologically correct isosurfaces”.

Technical report, Technical Report CN/95-17, CERN, 1995.

[19] G. M. Nielson. “On marching cubes”. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):283–297, 2003.

[20] A. Lopes and K. Brodlie. “Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing”. *IEEE Transactions on Visualization and Computer Graphics*, pages 19–26, 2003.

[21] I. Fujishiro, Y. Maeda, H. Sato, and Y. Takeshima. “Volumetric data exploration using interval volume”. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):144–155, 1996.

Appendix : Proof of Topology Preservation During TPTD

Consider a cube where a trilinear interpolation and TPTD are applied. The proof is done by showing that, for any isovalue inside a cube, (i) the PL(piecewise linear) isosurface is topologically equivalent to the trilinear isosurface on each face of the cube, and (ii) the topologies are equivalent inside the cube. Note that a PL isosurface inside a cube should always be manifold (except for a degenerate case). There can be no closed PL isosurface component inside a cube.

First, it is easy to see that the decomposition preserves trilinear isosurface topology on each face. For each face of a cube, TPTD decomposes it into four triangles when there is a face saddle, otherwise it is decomposed into two triangles. Three contour connectivities are possible, except for the case of no contour on a face, where symmetric cases are ignored. **Fig. 3** shows that each isocontour connectivity of a bilinear function on a face is preserved for any possible triangular mesh generated from our decomposition rule.

Second, we show that TPTD preserves the topology of the trilinear isosurface inside a cube. We classify each of the corner vertices and saddle points of TPTD into either *up-vertex* or *down-vertex* based on whether the function value on it is greater than an isovalue or not. The connected components of either the up-vertices or down-vertices uniquely represent isosurface components. We consider only the case where the connected components of the up-vertices uniquely represent isosurface components. We ignore the other case, since it is symmetric, as shown in **Fig. 1**, where the up-vertices among the corner vertices are marked with black solid circles. If the connected components of the trilinear isosurface are separated inside a cube, the connected components of the corresponding up-vertices should be also separated, and vice versa. Consider cases (i), (ii), and (iii). If there is no hole, the connected components of the up-vertices on the faces are separated from each other inside a cube in order to form a simple sheet (disk) for each connected component, which is consistent with the actual trilinear isosurface topology. If there is a hole, the connected components of up-vertices on faces are connected through a body saddle point inside a cube to form a tunnel isosurface component. The reason why we choose the second biggest face saddle in case (ii) with three or four face saddles and (v) with one body saddle is to avoid connecting components of the up-vertices inside a cube that need to be separated. For example, if we choose the smallest or the biggest face saddle in configuration 7.2, two components of the up-vertices on the faces of a cube can be connected through an edge. Hence, two separate isosurface components would be connected by a tunnel.

Fig. 11 is an example showing that TPTD preserves the topology of the trilinear isosurface for configuration 6.1.1 and 6.1.2 when a cube has one body saddle and one face saddle. Since TPTD performs topology preserving triangular decomposition for each face (as shown in **Fig. 3**), the PL isocontour on each face is topologically correct. The connected components of the up-vertices on the faces uniquely represent isocontour components (closed red contours in (a)).

The topology of the trilinear isosurface inside the cube depends on the sign of the body saddle, which is mathematically shown in [16]. If the sign is positive, TPTD combines the two connected components of the up-vertices on the faces into one component inside the cube, resulting in a single surface with a neck (**Fig. 11** (c)). Otherwise, TPTD separates the two components of the up-vertices inside the cube (**Fig. 11** (b)). We would like to emphasize that TPTD correctly reconstructs the trilinear isosurface topology by appropriately connecting the saddle points to the corner vertices of a cube in order to merge the components of the up-vertices inside a cube when the trilinear isosurface has a neck (tunnel), and to separate them when the trilinear isosurface contains separate surface sheets.

In cases (iv) and (v), where there are six face saddles, the configurations, except for 13.5.1 and 13.5.2, are proved in a similar manner to the cases of (i), (ii), and (iii). The configurations 13.5.1 and 13.5.2 can be proved by removing tetrahedra that contribute to the small isosurface component and apply the same proof of (i), (ii), and (iii) to the remaining isosurfaces for topological correctness.

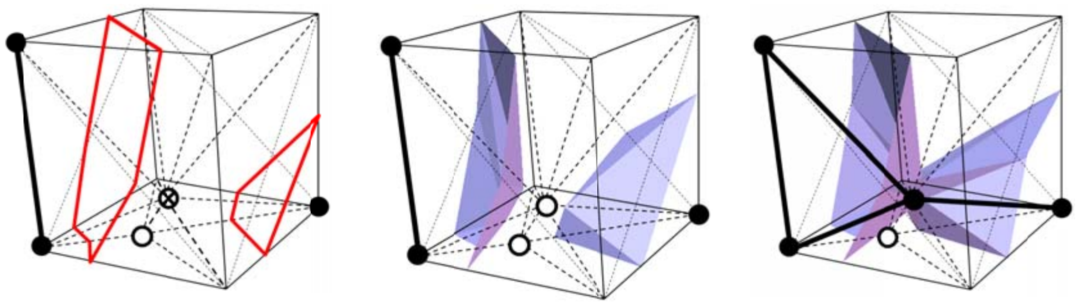


Fig. 11. An example (a cube with $s_b = 1$, $s_f = 1$) that shows TPTD preserves the topology of a trilinear isosurface (i) on each face as shown in (a), and (ii) inside a cube using saddle points, as shown in (b) and (c). (⊗ represents a body saddle in (a). ● represents an up-vertex.)

Bong-Soo Sohn is an Assistant Professor in the School of Computer Science and Engineering at Chung-Ang University, Seoul, Korea. Prior to joining Chung-Ang University, he was a Full-Time Lecturer in the Department of Computer Engineering at Kyungpook National University, Daegu, Korea. He received his M.S. and Ph.D. degrees in Computer Sciences from the University of Texas at Austin, in 2001 and 2005 respectively. He received his B.S. degree in Computer Science from Seoul National University, Seoul, Korea. His research interests are in the areas of graphics, visualization, 3D image processing, and medical applications.