3D Finite Element Meshing from Imaging Data [⋆]

Yongjie Zhang   Chandrajit Bajaj   Bong-Soo Sohn

Institute for Computational Engineering and Sciences
Department of Computer Sciences
The University of Texas at Austin

Abstract

This paper describes an algorithm to extract adaptive and quality 3D meshes directly from volumetric imaging data. The extracted tetrahedral and hexahedral meshes are extensively used in the Finite Element Method (FEM). A top-down octree subdivision coupled with the dual contouring method is used to rapidly extract adaptive 3D finite element meshes with correct topology from volumetric imaging data. The edge contraction and smoothing methods are used to improve the mesh quality. The main contribution is extending the dual contouring method to crack-free interval volume 3D meshing with feature sensitive adaptation. Compared to other tetrahedral extraction methods from imaging data, our method generates adaptive and quality 3D meshes without introducing any hanging nodes. The algorithm has been successfully applied to constructing the geometric model of a biomolecule in finite element calculations.

Key words: adaptive and quality mesh, correct topology, feature sensitive adaptation, hanging node.

1 Introduction

The development of finite element simulations in medicine, molecular biology and engineering has increased the need for quality finite element meshes. Although there has been tremendous progresses in the area of surface reconstruction and 3D geometric modelling, it still remains a challenging process to generate 3D meshes directly from imaging data, such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI) and Signed Distance Function (SDF). The imaging data V is given in the form of sampled function values on rectilinear grids, $V = \{F(i, j, k) \mid i, j, k$ are indices of x, y, z coordinates in a rectilinear grid.$\}$. We assume a continuous function F is constructed through the trilinear interpolation of sampled values for each cubic cell in the volume.

For accurate and efficient finite element calculations, it is important to have adaptive and quality geometric models with minimal number of elements. The studied object may have complicated

[⋆] Visit http://www.ices.utexas.edu/∼jessica/meshing
Email addresses: jessica@ices.utexas.edu (Yongjie Zhang), bajaj@cs.utexas.edu (Chandrajit Bajaj), bongbong@cs.utexas.edu (Bong-Soo Sohn).

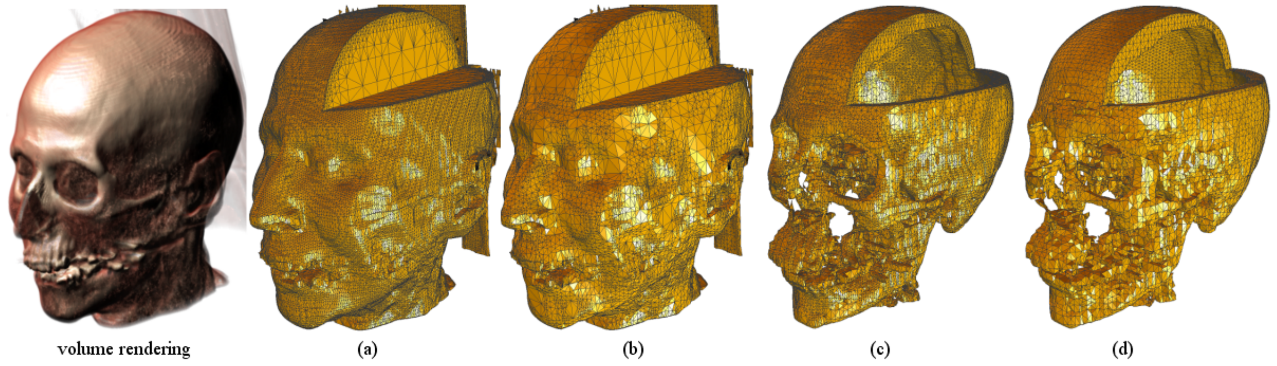


Fig. 1. Adaptive tetrahedral meshes are extracted from a CT-scanned volumetric data(UNC Head). Depending on the selected isovalues, different meshes of a skin and a skull are constructed. The number of tetrahedra can be controlled by choosing a user specified error tolerance (tet# - (b) 935124, (c) 545269, (d) 579834, (e) 166271). Note that the extracted mesh has no crack and no hanging node.

topology. Figure 21 shows an interval volume between two isosurfaces from the SDF volumetric data of a knee. The two surfaces have the same topology in Figure 21(d), while the topology of the inner surface may be different from the topology of the outer one (Figure 21(b)). In this paper, we present a comprehensive approach to extract tetrahedral and hexahedral meshes directly from imaging data.

$$S_F(c) = \{(x,y,z) : F(x,y,z) = c\} \tag{1}$$
$$I_F(\alpha_1,\alpha_2) = \{(x,y,z) : \alpha_1 \leq F(x,y,z) \leq \alpha_2\} \tag{2}$$

Given volumetric imaging data, α_1 and α_2 ($\alpha_1 < \alpha_2$) represent two isovalues, each of which defines a boundary isosurface (Equation (1)) of an interval volume (Equation (2)). The main steps to extract tetrahedral/hexahedral meshes for the interval volume, I_F, between the two isosurfaces are as follows:

(1) Preprocessing – volumetric denoising and isovalue selection.
(2) Adaptive 3D meshing with correct topology and feature preservation.
(3) Quality improvement

Noise often exists in imaging data, especially in CT and MRI data. The bilateral prefiltering coupled with anisotropic diffusion methods [5] is applied to smooth the volumetric data. Accurate gradient estimation can also be obtained. The Contour Spectrum [4] provides quantitative metrics of a volume to help us select two suitable isovalues for the interval volume. If the imaging data has no noise (for example, SDF data) and two isovalues are given to define the interval volume, the preprocessing step can be skipped.

We extend the idea of dual contouring to interval volume tetrahedralization and hexahedralization from volumetric Hermite data (position and normal information). Dual Contouring [30] analyzes those edges that have endpoints lying on different sides of the isosurface, called *sign change edge*. Each sign change edge is shared by four (uniform case) or three (adaptive case) cells, and one minimizer is calculated for each of them by minimizing a predefined Quadratic Error Function

(QEF) [24]. The QEF is defined as follows:

$$QEF[x] = \sum_i (n_i \cdot (x - p_i))^2 \tag{3}$$

where p_i, n_i represent the position and unit normal vectors of the intersection point respectively. For each sign change edge, a quad or a triangle is constructed by connecting the minimizers. These quads and triangles provide an approximation of the isosurface.

Each sign change edge belongs to a boundary cell. We present a systematic way to tetrahedralize the volume in the boundary cell. For uniform grids, it is easy to deal with the interior cells. We only need to decompose each cell into five tetrahedra. The adaptive case is more complicated. In order to avoid introducing hanging nodes, which are strictly prohibited in finite element meshes, we design an algorithm to tetrahedralize the interior cell depending on the resolution levels of all its neighbors. Figure 1 shows an example of adaptive tetrahedral meshes extracted from a scanned CT data. As a byproduct, the uniform hexahedral mesh extraction algorithm is simpler. We analyze each interior vertex (a grid point inside the interval volume) which is shared by eight cells. One minimizer is calculated for each of them, and those eight minimizers construct a hexahedron.

Reconstructing a mesh with correct topology is important for accurate finite element calculations. We guarantee the resulting 3D mesh is topologically equivalent to the real interval volume $I_F(\alpha_1, \alpha_2)$. The topology of the 3D mesh is preserved during the simplification process. Unlike the dual contouring method [30], we use a different error function based on the function difference normalized by gradients. The function approximates the maximum difference between coarse and fine level isosurfaces to decide the level of adaptivity. Using this error measurement and a user specified error tolerance, we can identify octree cells of appropriate levels which satisfy the threshold criteria. The result shows the error function we use yields feature-sensitive adaptation as shown in Figure 13 (d). Since we still use QEF for computing minimizing vertices, we can also preserve sharp edges and corners.

The tetrahedral meshes extracted from volume data can not be used for finite element calculations directly, since some elements may have bad quality. The edge-ratio and Joe-Liu parameter are chosen to measure the mesh quality. The edge contraction method removes tetrahedra with bad edge-ratios, and the smoothing method improves the mesh quality measured by the Joe-Liu parameters. We applied our algorithm to extracting a tetrahedral mesh from the accessibility function of a mouse acetylcholinesterase (mAChE) biomolecule. The extracted meshes have been used for efficient and correct finite element calculation of a diffusion problem [58] [57].

The remainder of this paper is organized as the following: Section 2 summarizes the related work on quality 3D mesh generation; Section 3 reviews the preprocessing step. Section 4 explains the detailed algorithm for extracting tetrahedral/hexaheral meshes from the interval volume; Section 5 describes a subdivision method for preserving correct topology in the reconstructed mesh; Section 6 talks about the feature sensitive error function. Section 7 discusses the mesh quality improvement. Section 8 shows some results and applications. The final section presents our conclusion.

2 Previous Work

Isosurface Extraction In most cases, an isosurface is extracted in the form of piecewise linear approximation for the modeling and rendering purpose. The Marching Cubes algorithm (MC) [38] visits each cell in a volume and performs local triangulation based on the sign configuration of the eight vertices. To avoid visiting unnecessary cells, accelerated algorithms [65] [3] minimizing the time to search for contributing cells are developed. The isosurfaces of a function defined by the trilinear interpolation inside a cubic cell may have a complicated shape and topology which cannot be reconstructed correctly using MC. The function values of face and body saddles in the cell can be used to decide the correct topology and consistent triangulation of an isosurface in the cell [42]. Lopes and Brodlie [37] provided a more accurate triangulation.

Main drawbacks of MC and its variants are (i) the produced mesh is uniform, (ii) badly shaped triangles are generated. and (iii) sharp features in the data are not preserved. An adaptive isosurface can be generated by triangulating cells with different levels. When the adjacent cubes have different resolution levels, the cracking problem will happen. To keep the face compatibility, the gravity center of the coarser triangle is inserted, and a fan of triangles are used to approximate the isosurface [64]. The chain-gang algorithm [32] was presented for isosurface rendering of super adaptive resolution (SAR) and resolves discontinuities in SAR data sets. Progressive multiresolution representation and recursive subdivision are combined effectively, and isosurfaces are constructed and smoothed by applying the edge bisection method [45]. A surface wave-front propagation technique [66] is used to generate multiresolution meshes with good aspect ratio.

The enhanced distance field representation and the extended MC algorithm [31] can detect and reconstruct sharp features in the isosurface. By combining SurfaceNets [27] and the extended Marching Cubes algorithm [31], octree based Dual Contouring [30] can generate adaptive isosurfaces with good aspect ratio and preservation of sharp features. Elements in the extracted mesh often have bad aspect ratio. These elements can not be used for finite element calculations. The grid snapping method reduces the number of elements in an approximated isosurface and also improves the aspect ratio of the elements [41]. [7] studied how to generate triangular meshes with bounded aspect ratios from a planar point set. [40] proposed an algorithm, called QMG, to triangulate a d-dimensional region with a bounded aspect ratio.

Tetrahedral Mesh Generation Octree based, advancing front based and Delaunay like techniques were used for tetrahedral mesh generation. The octree technique recursively subdivides the cube containing the geometric model until the desired resolution is reached [51]. Advancing front methods start from a boundary and move a front from the boundary towards empty space within the domain [36] [22] [50]. The Delaunay criterion is called ‘empty sphere’, which means that no node is contained within the circumsphere of any tetrahedra of the mesh. Delaunay refinement is to refine the triangles or tetrahedra locally by inserting new nodes to maintain the Delaunay criterion. Different approaches to define new nodes were studied [16] [52] [14]. Sliver exudation [15] was used to eliminate those slivers. A deterministic algorithm [14] was presented for generating a weighted Delaunay mesh with no poor quality tetrahedra including slivers. Shewchuk [53] solved the problem of enforcing boundary conformity by constrained Delaunay triangulation (CDT). Delaunay refinement [52], edge removal and multi-face removal optimization algorithm [54] were used to improve the tetrahedral quality. Shewchuk [55] provided some valuable conclusions on quality measures for finite element method.

MC was extended to extract tetrahedral meshes between two isosurfaces directly from volume data [23]. A Branch-on-Need octree was used as an auxiliary data structure to accelerate the extraction process . A different algorithm, Marching Tetrahedra (MT), was proposed for interval volume tetrahedralization [43]. A multiresolution framework [69] was generated by combining recursive subdivision and edge-bisection methods. Since many 3D objects are sampled in terms of slices, Bajaj et. al introduced an approach to construct triangular surface meshes from the slice data [1], and tetrahedralize the solid region bounded by planar contours and the surface mesh [2].

Hexahedral Mesh Generation Eppstein [18] started from a tetrahedral mesh to decompose each tetrahedron into four hexahedra. Although this method avoids many difficulties, it increases the number of elements. There are four distinct methods for unstructured all-hex mesh generation: grid-based, medial surface, plastering and whisker weaving. The grid-based approach generates a fitted 3D grid of hex elements on the interior of the volume [48] [49]. Medial surface methods involve an initial decomposition of the volume [46] [47]. Plastering places elements on boundaries first and advances towards the center of the volume [10] [8]. Whisker weaving first construct the spatial twist continuum (STC) or dual of the hex mesh, then the hex elements can be fitted into the volume using the STC as a guide [60].

Quality Improvement Algorithms for mesh improvement can be classified into three categories [61] [44]: local coarsening/refinement by inserting/deleting points, local remeshing by face/edge swapping and mesh smoothing by relocating vertices.

Laplacian smoothing, in its simplest form, relocates the vertex position at the average of the nodes connecting to it. This method generally works quite well for meshes in convex regions. However, it can result in distorted or even inverted elements near concavities in the mesh. [26] weighted the contribution of each neighboring node in the average function. [19] constrained the node movement in order to avoid the creation of invert elements. [39] discretized the Laplacian operator using Voronoi cells and the mixed Finite Element/Finite Volume method. The discretized format was used to solve surface modelling problems using the finite difference method [67]. [33] and [29] developed methods to extend to anisotropic meshes. Instead of relocating vertices based on a heuristic algorithm, people searched a optimization technique to improve the mesh quality. The optimization algorithm measures the quality of the surrounding elements to a node and attempts to optimize it. The algorithm is similar to a minimax technique used to solve circuit design problem [13]. The optimization-based smoothing yields better results while it is more expensive than Laplacian smoothing. Therefore, [11] [21] [20] recommended a combined Laplacian/optimization-based approach, which uses Laplacian smoothing when possible and only uses optimization-based smoothing when necessary. Physically-based simulations are used to reposition nodes [35]. Anisotropic meshes are obtained from bubble equilibrium [56] [9].

3 Preprocessing

Since noise in imaging data influences the accuracy of the extracted meshes, it is important to remove it before the mesh extraction process. The isotropic diffusion method can remove noise, but blurs features such as edges and corners. In order to preserve features during the process of noise smoothing, anisotropic diffusion [63] was proposed by introducing a diffusion tensor. Generally, a Gaussian filter is used to calculate the anisotropic diffusion tensor before smoothing, but it also blurs features. Bilateral filtering [62], which is a nonlinear filter combining domain and range filtering, was introduced to solve this problem. Anisotropic diffusion can be used for fairing out noise in both surface meshes and functions defined on the surface [6] [17]. Here we use the anisotropic diffusion method [5] to smooth noise. In order to obtain more accurate computation of curvature and gradient for anisotropic diffusion tensor, the bilateral prefiltering combining the domain and range filtering together is chosen instead of Gaussian filtering.

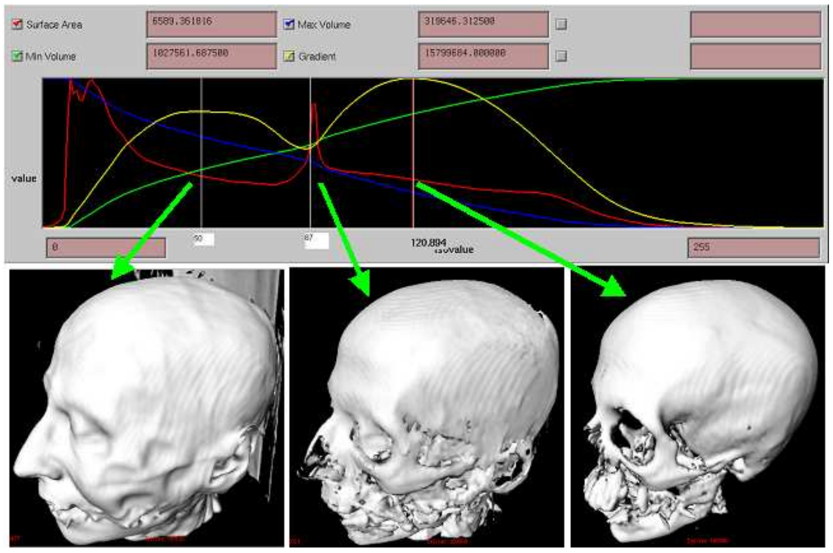

Surface Area 6589.361816
Max Volume 319646.312500
Min Volume 1027561.687500
Gradient 15799684.000000

Fig. 2. The contour spectrum of the UNC human head

Mesh extraction from imaging data requires the selection of suitable boundary isosurfaces. We use a user interface called Contour Spectrum [4] to find significant isosurfaces. The Contour Spectrum computes quantitative properties such as surface area, volume, and gradient integral of contours in real time, and helps to choose suitable isosurfaces by showing the related spectrum in a 2D plane. For instance, we can obtain isosurfaces of a skin and a skull in CT scanned human head data (Figure 2) by taking isovalues with the local maximum of the gradient integral. In the case of the interval volume, the topology of inner and outer isosurfaces may need to be controlled depending on the application. A contour tree [12] can be used to capture the topological information on each isosurface and help choose isosurfaces with desirable topology. For example, we can choose the inner and outer isosurfaces with exactly the same topology by taking isovalues lying on the same edges in a contour tree.

4 3D Mesh Extraction

In this section, our goal is to tetrahedralize or hexahedralize the interval volume between two isosurfaces by using an octree-based data structure. We describe in detail how to extract adaptive tetrahedral meshes from volume data. First, we discuss the triangulation in 2D problems, then we extend it to 3D tetrahedralization. A hexahedral mesh generation algorithm is presented at the end of this section. Here are definitions used in the algorithm description:

Sign Change Edge: A sign change edge is an edge whose one vertex lies inside the interval volume (we call it the interior vertex of this sign change edge), while the other vertex lies outside.

Interior Edge in Boundary Cell: In a boundary cell, those edges with both vertices lying inside the interval volume are called interior edges.

Interior Face in Boundary Cell: In the boundary cell, those faces with all four vertices lying inside the interval volume are called interior faces.

Interior Cell: Different from the boundary cell, all the eight vertices of an interior cell lie interior to the interval volume.

4.1 Uniform Tetrahedral Mesh Extraction

For isosurface extraction, we only need to analyze boundary cells – those cells that contain sign change edges, or those cells that contain the isosurface. There are four neighbor cubes which share the same sign change edge. Dual Contouring generates one minimal vertex for each neighbor cube by minimizing the QEF, then connects them to generate a quad. By marching all the sign change edges, the isosurface is obtained. For tetrahedral mesh extraction, cells inside the interval volume should also be analyzed besides the boundary cells.

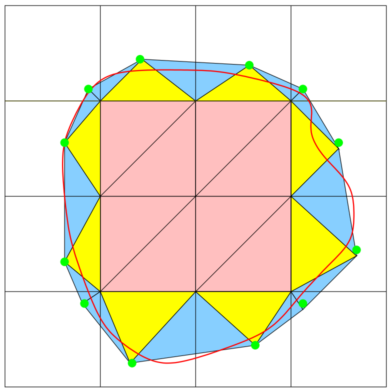


Fig. 3. Uniform triangulation - the red curve represents the isocontour, green points represent minimizers.

4.1.1 *Uniform 2D Triangulation*

Figure 3 is a uniform triangulation example of the area interior to the isocontour in two dimensions. There are three different cases which need to be dealt with separately.

(1) Sign change edge – find the minimizers of the two cells which share the edge, then the two minimizers and the interior vertex of the edge construct a triangle (blue triangles).
(2) Interior edge in boundary cell – find the QEF minimizer of the boundary cell, then the minimizer and this interior edge construct a triangle (yellow triangles).
(3) Interior cell – decompose each interior cell into two triangles (pink triangles).

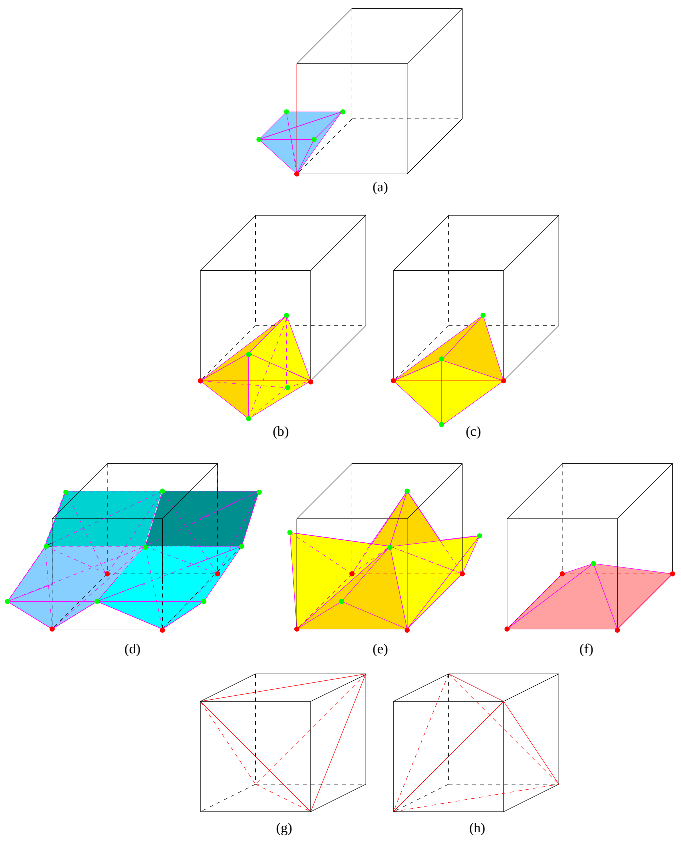


Fig. 4. The case table of uniform tetrahedralization - the red vertex means it lies interior to the interval volume, otherwise, it is outside. Green points represent minimizers. (a) - a sign change edge; (b)(c) - an interior edge in a boundary cell; (d)(e)(f) - an interior face in a boundary cell; (g)(h) - an interior cell.

4.1.2 *Uniform 3D Tetrahedralization*

Compared to 2D triangulation, three dimensional tetrahedral meshing is more complicated. Figure 4 shows the case table of uniform tetrahedralization.

(1) Sign change edge – decompose the quad into two triangles, then each triangle and the interior vertex of this edge construct a tetrahedron. In Figure 4(a), the red line represents the sign change edge, and two blue tetrahedra are constructed.

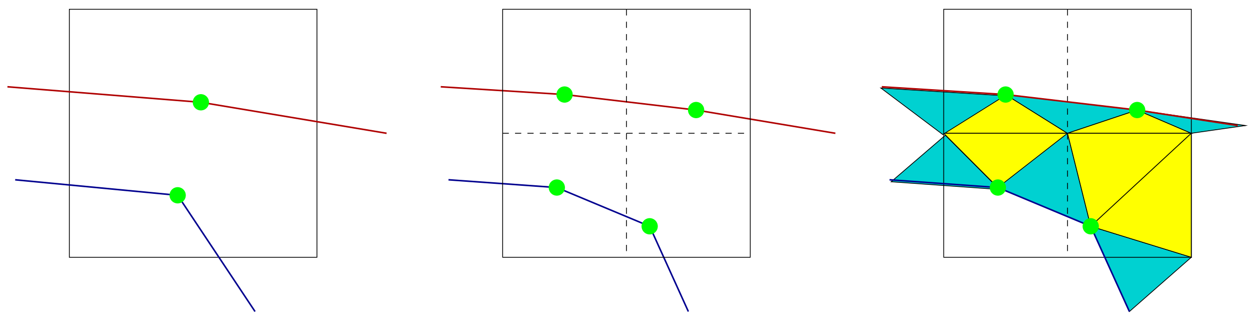


Fig. 5. A two dimensional example of cell subdivision for enforcing each cell to have at most one boundary isocontour. When two boundary isocontours pass through the same cell, the cell is recursively subdivided until each sub-cell contains at most one minimizer.

(2) Interior edge in boundary cell – find the QEF minimizers of the boundary cell and its boundary neighbor cells, then two adjacent minimizers and the interior edge construct a tetrahedron. In Figure 4(b)(c), the red cube edge represents the interior edge. (b) shows four minimizers to construct four edges, each of which construct a tetrahedron with the interior edge, so totally four tetrahedra are constructed. While (c) assumes the cell below this boundary cell is interior to the interval volume, so there is no minimizer for it. Therefore we obtain three minimizers, and only two tetrahedra are constructed.

(3) Interior face in boundary cell – find the QEF minimizer of the boundary cell, then the interior face and the minimizer construct a pyramid, which can be decomposed into two tetrahedra (Figure 4(f)). Figure 4(d)(e)(f) show a sequence of how to generate tetrahedra when there is only one interior face in the boundary cell. (d) analyzes four sign change edges, (e) deals with four interior edges and (f) fills the gap.

(4) Interior cell – decompose the interior cube into five tetrahedra. There are two different decomposition ways (Figure 4(g)(h)). For two adjacent cells, we choose a different decomposition method to avoid the diagonal choosing conflict problem.

Our meshing algorithm assumes that there is only one minimizer point in a cell. This means two different boundary isosurfaces of an interval volume can not pass through the same cell. Therefore, we enforce every cell to have at most one boundary isosurface before actual meshing of cells. If a cell contains two boundary isosurfaces, we recursively subdivide the cell into eight identical sub-cells until each sub-cell contains at most one boundary isosurface as shown in Figure 5. The detailed subdivision algorithm is described in Section 5.

4.2 Adaptive Tetrahedral Mesh Extraction

Uniform tetrahedralization usually generates an over-sampled mesh. Adaptive tetrahedral meshing is an effective way to minimize the number of elements while preserving the accuracy requirement. First, we split the volume data by using the octree data structure to obtain denser cells along the boundary, and coarser cells inside the interval volume (Figure 7). The QEF value is calculated for each octree cell. When the error tolerance ε is selected, the octree is traversed in a bottom-up manner to obtain the cells which satisfy the error tolerance requirement. We call this set of cells to be leaf cells of the octree assuming we pruned unnecessary nodes from the tree.

Each leaf cell may have neighbors at different levels. An edge in a leaf cell may be divided into several edges in its neighbor cells. Therefore it is important to decide which edge should be analyzed. The Dual Contouring method provides a good rule to follow – we always choose the minimal edge. Minimal edges are those edges of leaf cubes that do not properly contain an edge of a neighboring leaf.

Similar to uniform tetrahedral mesh extraction, we need to analyze the sign change edge, the interior edge/face in the boundary cell, and the interior cell. When we analyze boundary cells, only minimal edges/faces are analyzed. Compared to the uniform case, the only difference is how to decompose interior cells into tetrahedra without hanging nodes.

Hanging Node: a hanging node is one that is attached to the corner of one triangle but does not attach to the corners of the adjacent triangles. Generally, a hanging node is a point that is a vertex for some elements (e.g., triangle, quad, tetra, hexa), but it is not for its other neighbor elements that share it. It lies on one edge or one face of its neighbors, for example, a T-Vertex.

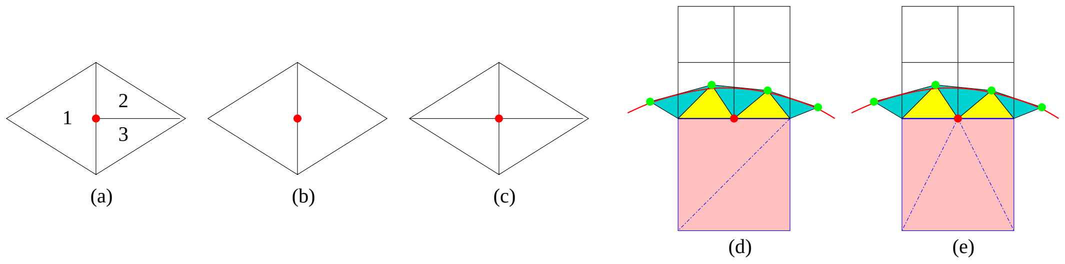


Fig. 6. Hanging node removal - the red point is a hanging node. (a) - T-Vertex; (b) - merging two triangles; (c) - splitting method. (d) and (e) show an example of hanging node removal by re-triangulating the interior cell (pink). The red curve is the real isocontour, and green points are minimizer points for boundary cells. (d) - the triangulation of the interior cell without considering the hanging node; (e) - the re-triangulation of the interior cell to remove the hanging node.

Figure 6 shows two methods to remove hanging nodes - splitting and merging. In the T-Vertex example (Figure 6a), there is a hanging node (red point). Only the right two triangles (Number 2 and 3) need to be modified if we use the merging method (Figure 6b); while only the left one (Number 1) needs to be modified if we intend to split the mesh (Figure 6c). In order to maintain the accuracy, we adopt the splitting method in our algorithm.

Lemma: Only the interior cell needs to be modified if the splitting method is adopted.

Proof: All the leaf cells can be divided into two groups: the boundary cell and the interior cell.

(1) Interior cell – Since its neighbor cells may have higher resolution levels, hanging nodes are unavoidable. In Figure 6d, there is a hanging node if we triangulate the interior cell as in the uniform case. Figure 6e shows a re-triangulating method to remove the hanging node.
(2) Boundary cell – There are two rules for the sign change edge and the interior edge/face in boundary cells. The two rules guarantee that no hanging nodes need to be removed for the boundary cell if only the splitting method is chosen.

- Minimal Edge/Face Rule - only minimal edges/faces are chosen. This rule keeps the analyzed edges/faces owning the highest resolution level compared with their neighbors.
- Only one minimizer is generated for each leaf cell.

4.2.1 Adaptive 2D Triangulation

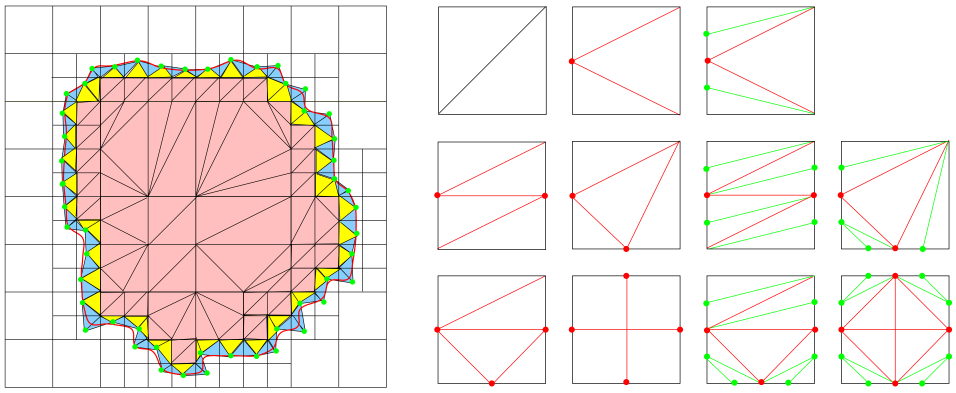


Fig. 7. Left: adaptive triangulation. The red curve represents the isocontour, green points represent minimizers. Right: The case table for decomposing the interior cell into triangles in 2D. Suppose the resolution level of this cell is κ, and middle points appear on the shared edges if its neighbors have higher level than κ. Red points and red lines mean its neighbors have level $(\kappa+1)$; green points and green lines mean its neighbors have higher level than $(\kappa+1)$. The case table can be easily generalized to any other adaptive cases.

Figure 7(left) shows an example of how to triangulate the interior area of an isocontour. Similarly, we need to analyze the following three problems:

(1) Sign change edge – if the edge is minimal, deal with it as in the uniform case (blue triangles).
(2) Interior edge in the boundary cell – if the edge is minimal, deal with it as in the uniform case (yellow triangles).
(3) Interior cell – Figure 7(Right) lists all the main cases of how to decompose an interior cell into triangles. The case table can be easily generalized to any other adaptive cases. In order to obtain triangles with good aspect ratio, we restrict the neighboring level difference to be ≤ 2.

Compared to the uniform case, the triangulation of interior cells is more complicated. All neighbors of an interior cell need to be checked because the neighbor cells are used to decide if there are any middle points on the shared edge. Suppose the resolution level of this cell is κ. We group into five cases according to the number of edges whose level is greater than κ. The i^{th} group means there are number i edges whose level is greater than κ, where $i = 0, \ldots, 4$. For each subdivided edge, it may be subdivided more than once, or the neighbor cell may have higher level than $(\kappa+1)$. So we need to search all the middle points on this edge by looking at the resolution levels of the neighbor cells. Figure 7(right) lists all the main cases of how to decompose the interior cell into triangles according to its neighbors' resolution levels. If all the four edges have already been subdivided, then we can use the recursive method to march each of the four sub-cells with the same algorithm. In this way, hanging nodes are removed effectively.

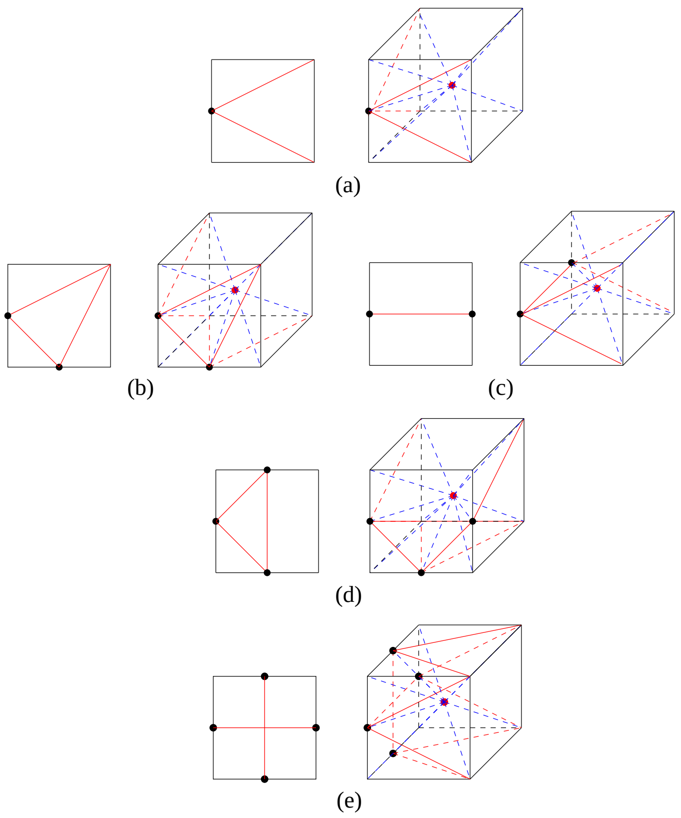


Fig. 8. The case table for decomposing the interior cell into tetrahedra when the neighboring level difference is one – in (a $\sim$ e), the left picture shows the triangulation format of one face according to Figure 7; the right one shows how to decompose the cell into tetrahedra without hanging nodes. (a) - one subdivided edge; (b)(c) - two subdivided edges; (d) - three subdivided edges; (e) - four subdivided edges. Any other adaptive cases are easily generalized using the case table in Figure 7.

4.2.2 Adaptive 3D Tetrahedralization

For three dimensional adaptive tetrahedralization, we use the similar algorithm with the uniform case when we deal with the boundary cell.

(1) Sign change edge – if the edge is minimal, deal with it as in the uniform case.
(2) Interior edge in the boundary cell – if the edge is minimal, deal with it as in the uniform case.
(3) Interior face in the boundary cell – identify all the middle points on the four edges, and decompose the face into triangles by applying the same algorithm as in the adaptive 2D case, then calculate the minimizer of this cell, each triangle and this minimizer construct a tetrahedron.
(4) Interior cell – decompose each face of the cube into triangles, just as how to deal with the interior cell for the adaptive 2D triangulation (Figure 7(right)), then insert a Steiner point at the cell center. Each triangle and the Steiner point construct a tetrahedron. Figure 8 shows how to construct tetrahedra when the neighboring level difference is one. Sometimes pyramids are constructed. In order to avoid the diagonal choosing conflict, we decide which diagonal is chosen to decompose one pyramid into two tetrahedra according to the odd-even property of the cell index.

By using the above algorithm, we extract tetrahedral meshes from volumetric imaging data successfully. Figure 9 (a) and (b) show the tetrahedral mesh of the human head model extracted from 65^3 volumetric data. The volume inside the skin isosurface is tetrahedralized.

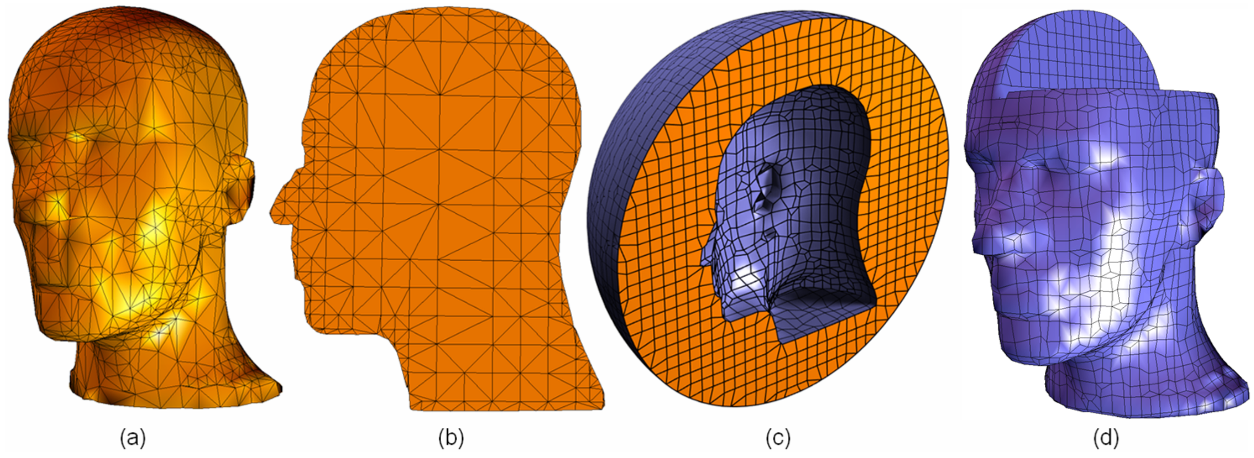


Fig. 9. Tetrahedral/hexahedral meshes of the human head model - (a) and (b) show adaptive tetrahedral meshes, (c) and (d) show the hexahedral meshes. (a): the tetrahedralization of the volume inside the head; (b): a cross section of (a); (c): the hexahedralization of the volume between the human head and a sphere boundary; (d): the hexahedralization of the volume inside the head.

4.3 Hexahedral Mesh Extraction

Finite element calculations sometimes require hexahedral meshes instead of tetrahedral meshes. Each hexahedron has eight points. In the tetrahedralization process, we deal with edges shared by at most four cells. This means that we can not get eight minimizers for each edge. However, each vertex within the interval volume is shared by eight cells. We can calculate a minimizer for each of them. In the case of interior cells, we set the center point as the minimizer. These eight minimizers can then be used to construct a hexahedron. Figure 9 (c) and (d) show two hexahedral meshes for the head model, which are used to solve electromagnetic scattering simulations in finite element calculations.

5 Mesh Topology

Constructing an adaptive 3D mesh with correct topology plays an important role in accurate and efficient finite element calculations. Our goals in this section are (i) meshing with correct topology and (ii) topology preserving adaptive meshing. The topology of a mesh defined by an interval volume depends on the topology of two boundary isosurfaces enclosing the interval volume. Therefore we focus on the topology of an isosurface. We use the term ‘dual contour’ as a polygonized isosurface extracted using the dual contouring method [30]. ‘Correct topology’ means that an extracted dual contour is topologically equivalent to the real isosurface defined from an original

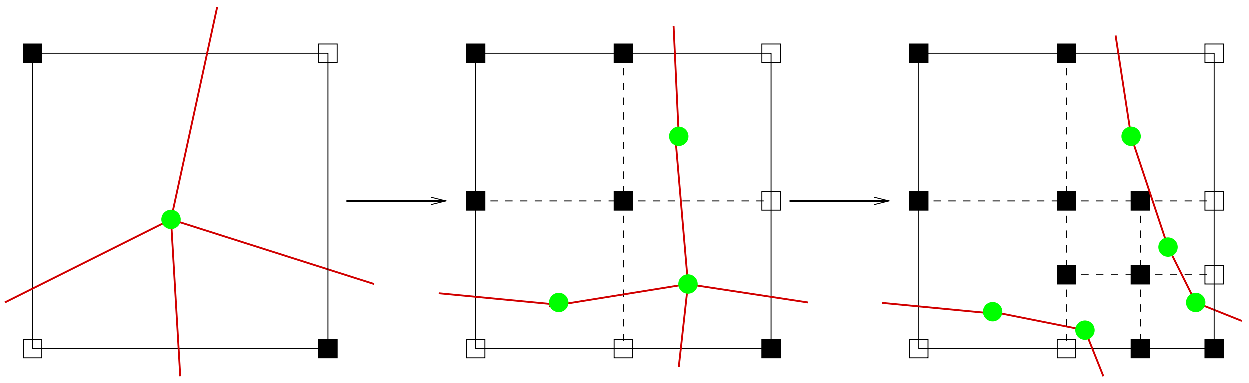
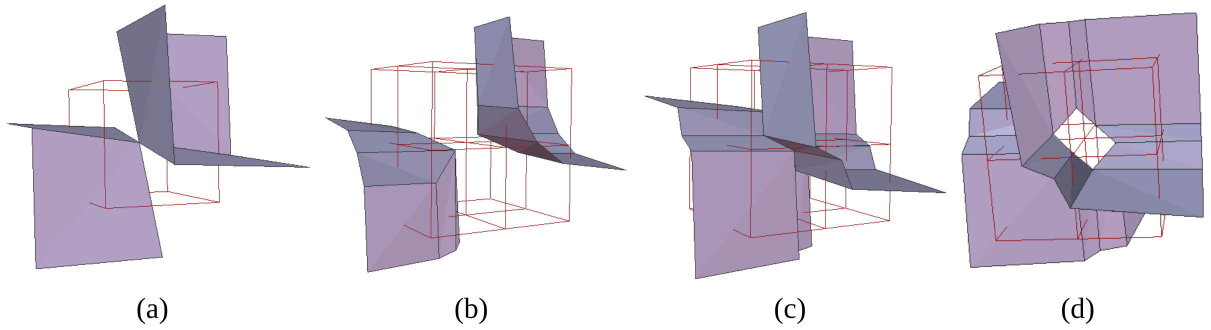


Fig. 10. A two dimensional example on the recursive subdivision of a cubic cell in the finest level for reconstructing a dual contour with correct topology.



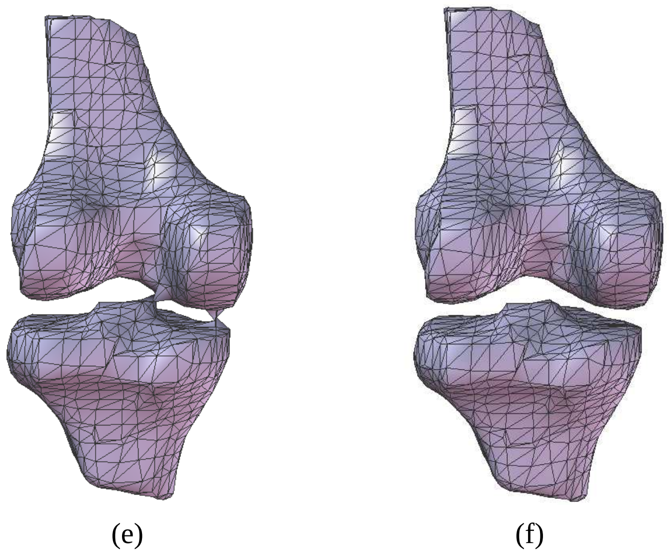


Fig. 11. (a)-(d): An example of an ambiguous case for a finest-level cell. Both the front-right-up and back-left-down vertices have the positive sign, all the other vertices have the negative sign. (a): A non-manifold dual contour is generated in a cube with an ambiguous case. In this case, the real isosurface in the cube is topologically equivalent to either two simple disks (b) or a tunnel (c) [37]. The topologically correct dual contours, (b) and (c), can be constructed by the simple subdivision of a cell in (a). (d) is displayed in a different view angle from (c) to emphasize that the tunnel shape is correctly reconstructed. (e)-(f): A real example (the human knee) on topologically correct reconstruction of a dual contour. (e): before subdivision, (f): after subdivision. Note that non-manifolds in (e) are removed in (f).

input function. ‘Topology preserving’ means the topology of a dual contour is preserved during simplification.

Assume the function F within a cubic cell is defined by the trilinear interpolation of the eight vertex values. The sign of a vertex is defined to be positive when its value is greater than or equal to an isovalue, and negative when its value is less than the isovalue. An isosurface within a cube, defined as $F(x,y,z) = \alpha$, may have different local topology depending on the configuration of signs at the eight vertices and the isovalue. There are 2^8 sign configurations which can be reduced into 14 cases using symmetry [38]. Several cases may have more than one local topology, and are termed ambiguous. We refer to [37] for all the cases of different local topology of an isosurface in a cube. When a case is ambiguous, the standard dual contouring method causes a non-manifold at the minimizing vertex in the cube. This makes the topology of the dual contour different from that of the real isosurface. On the other hand, if a cube has no ambiguity, then the real isosurface is topologically equivalent to the dual contour, which is always a simple manifold. Whether a case in a cube is ambiguous or not can be checked by collapsing each edge which has two vertices with the same sign into a vertex [30]. If the cube can be collapsed into an edge, then the cube has no ambiguity and the topologically correct dual contour is generated in the cube. An example of a cell with an ambiguous case is shown in Figure 11 (a) where the non-manifold dual contour is generated using a naive approach.

We use the recursive cell subdivision in the finest level to reconstruct a dual contour with correct topology when the cell contains a non-manifold dual contour. The subdivision algorithm is very similar with what we used for enforcing each cell to have at most one boundary isosurface. As a first step, all boundary cells in the finest level are identified. If a cell contains a non-manifold dual contour, we subdivide the cube into eight identical sub-cubes. The function values defined on newly generated vertices are calculated by the trilinear interpolation of the values at the eight cube vertices. We recursively repeat this process for the sub-cubes containing a non-manifold dual contour until each sub-cube contains a manifold dual contour. Figure 10 shows a 2D example.

We justify the correctness of the algorithm as follows. The function defined in a sub-cube is exactly same as the original function because we use the trilinear interpolation. If a sub-cube has no ambiguity and hence contains a manifold dual contour, then the topology of the dual contour in the sub-cube is correct in the sense that the dual contour is topologically equivalent to the real isosurface. Therefore, if every sub-cube contains a manifold dual contour, then the dual contour in each sub-cube has correct topology. Since we recursively subdivide a cube until every sub-cube has a manifold dual contour, the resulting dual contour within the finest cubic cell has correct topology.

In this way, we can obtain a dual contour with correct topology from every finest cell in an octree structure. Then we traverse the octree in a bottom-up manner to get an adaptively simplified dual contour. During the traverse from children cells to a parent cell, the topology of a dual contour can change. This may not be desirable. The paper [30] described an algorithm to check whether the fine dual contour is topologically equivalent to the coarse one or not. The sign of the middle vertex of a coarse edge/face/cube must be the same as the sign of at least one vertex of the edge/face/cube which contains the middle vertex. We restrict the octree simplification process to preserve the topology by using their algorithm.

Figure 11 shows an example of a finest-level cell with an ambiguous case. A non-manifold dual contour is generated in the cell (a). However, the real isosurface in the cell can have either two disks (b) or a tunnel shape (c)(d) depending on an isovalue. (b) and (c) show two results of a dual contouring after the cell subdivision on (a). The tunnel shape is correctly reconstructed as shown in (d). (e) and (f) show a real example on topologically correct reconstruction of a knee surface.

6 Feature Sensitive Adaptation

Finite element applications require a minimal number of elements while preserving important features on boundary surfaces for efficient and accurate calculations. For a given precision requirement, uniform meshes are always over-sampled with unnecessary small elements. Adaptive meshes are therefore preferable.

The level adaptivity can be controlled manually by regions or automatically by using an error function and a tolerance. For example, in the calculation of ligand binding rate constants on mAChE data (Figure 18), the geometric accuracy on the cavity area mostly affects the accuracy of the calculation. Therefore, we refine the cavity area as much as possible, while keep coarse meshes in other regions. In most cases, we want to control the level adaptivity with an error tolerance ε. For this purpose, we use an error function (Equation 5) which approximates the difference between isosurfaces defined from two neighboring levels. Large geometric change of surfaces is considered as features. The error is measured as large in a region which contains important features, therefore the features are not easily lost during the process of adaptive simplification. Similar error metric is used in [28]. The details of the difference approximation can be found in [59].

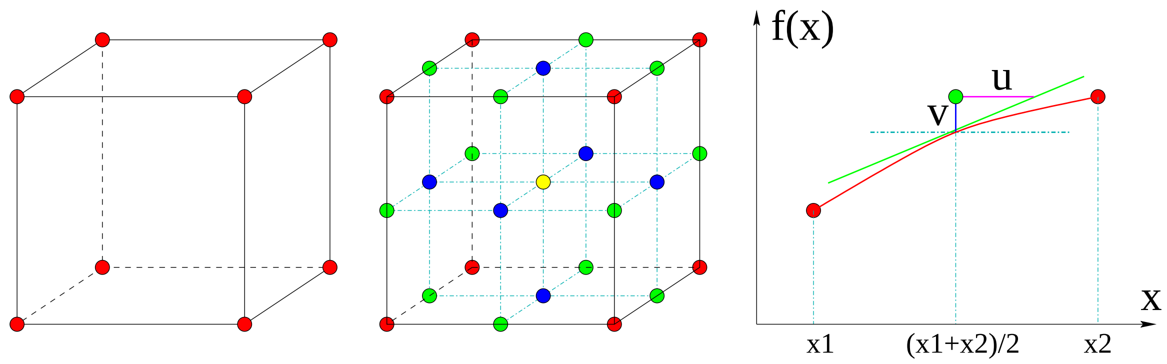


Fig. 12. Sampling points for the feature sensitive error function (Left - Level (i); Middle - Level (i+1).) and the isosurface error calculation in 1D (Right). In the right picture, the red curve represents the trilinear function in Level (i) (it becomes to a straight line in 1D), and the green straight line represents the tangent line of the trilinear function at the middle point.

In Figure 12, the left picture shows eight red vertices for resolution level (i), and the right picture shows twelve green edge middle points, six blue face middle points and one yellow center point besides those eight red vertices for level (i+1). For level (i), the eight red vertices' function values are given, and a trilinear function is defined in Equation (4), from which the function values of 12 edge middle points (green), 6 face middle points (blue) and 1 center point (yellow) can be obtained. For level (i+1), the function values of all vertices and all middle points are given. We want to estimate the difference of the isosurface between the two neighboring levels.

$$\begin{aligned} f^i(x,y,z) = {} & f_{000}(1-x)(1-y)(1-z) + f_{001}(1-x)(1-y)z \\ & + f_{010}(1-x)y(1-z) + f_{011}(1-x)yz \\ & + f_{100}x(1-y)(1-z) + f_{101}x(1-y)z \\ & + f_{110}xy(1-z) + f_{111}xyz \end{aligned} \tag{4}$$

$$Error = \sum \frac{|f^{i+1} - f^i|}{|\nabla f^i|} \tag{5}$$

The error function is defined in Equation (5). The right picture of Figure 12 shows the calculation of the isosurface error U at the green middle point in 1D. The blue line segment V represents the difference of the function values at the middle point, or $V = |f^{i+1} - f^i|$. The isosurface error U can be computed by $U = V/k$, where k is the slope of the green tangent line. In higher dimensions, the slope k becomes the magnitude of the gradient. In Equation (5), we only need to sum the isosurface error over all the middle points, including edge middle points, face middle points and the center point, since the function values at the eight vertices are the same for the two neighboring levels.

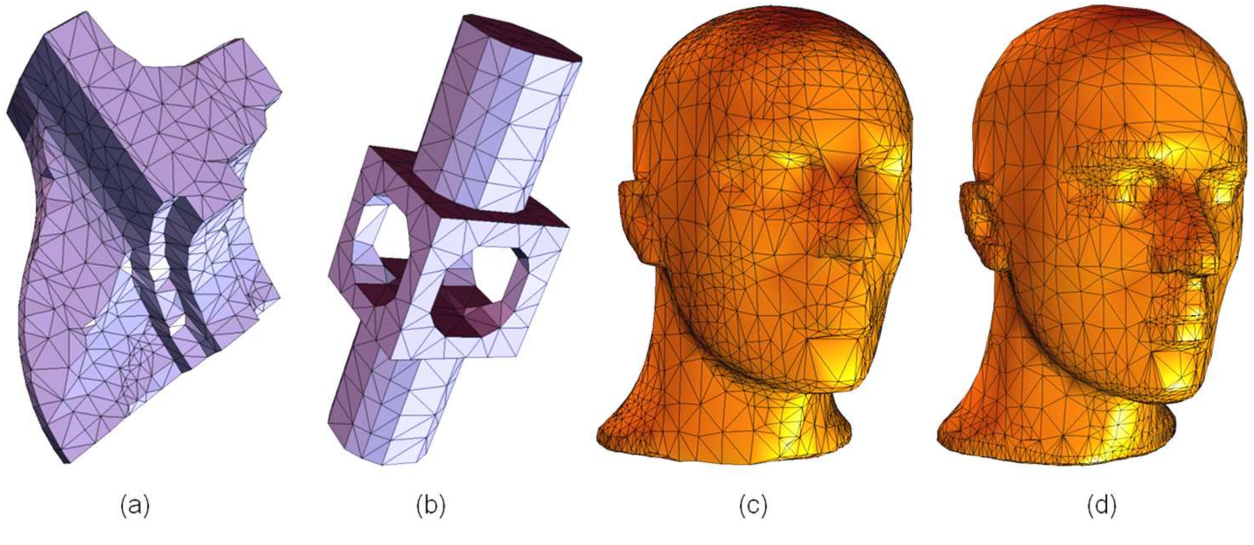


Fig. 13. (a) , (b) : Tetrahedral meshes of (a) fandisk and (b) mechanical part. Note that sharp edges and corners are accurately reconstructed ; (c) , (d) : the comparison of QEF ((c), 2952 triangles) and the feature sensitive error function ((d), 2734 triangles). The facial features are better refined in (d).

We still use QEF to calculate minimizer points, which are connected to approximate the isosurfaces, so we can also preserve sharp edges and corners as shown in Figure 13 (a) and (b). QEF and the error function in Equation (5) are compared in controlling the level adaptivity for the human head model (Figure 13, (c) and (d)), and Equation (5) yields more sensitive adaptivity for facial features, like the areas of nose, eyes, mouth and ears. The feature sensitive adaptivity is important for finite element meshes to identify and preserve necessary geometric and topological properties of the object while minimizing the number of elements.

7 Quality Improvement

Poorly shaped elements influence the convergence and stability of the numerical solutions. Since the extracted meshes may have such undesirable elements, we need an additional step for mesh quality improvement. First we need to define criteria to measure the mesh quality. Various functions have been chosen to measure the quality of a mesh element. For example, Freitag [21] defined poor quality tetrahedra using dihedral angles. George [25] chose the ratio of the element diameter such as the longest edge over the in-radius. We use the edge-ratio, the Joe-Liu parameter [34], and a minimum volume bound. These quality metrics are used to detect slivers and sharp elements which need to be removed. With these measures, the mesh quality can be judged by observing the worst element quality, and the distribution of elements in terms of their quality values.

- **Edge-ratio:** The edge-ratio of a tetrahedron is the ratio of the longest edge length over the shortest edge length.
- **Joe-Liu parameter:** Let $|s|$ denote the volume (which may be negative) of a given d-simplex 's', $v_i (i = 0, \ldots, d)$ denote the vertices of s, and e_{ij} denote the edges of s that connect v_i to v_j. We compute the Joe-Liu parameter of a d-simplex s as:

$$F(s,d) = \frac{f(s,d)}{g(s,d)} = \frac{2^{2(1-\frac{1}{d})} \times 3^{\frac{d-1}{2}} \times |s|^{\frac{2}{d}}}{\sum_{0 \leq i < j \leq 3} |e_{ij}|^2} \tag{6}$$

  For tetrahedral meshes, d is set to be 3. $F(s,3)$ is normalized to yield maximal value of 1 for the unit tetrahedron (with volume 1/6).
- **Minimum volume bound:** Calculate the volume for each tetrahedron and find the minimum volume of the extracted mesh. This volume parameter needs to be improved if the minimum volume is less than a threshold (e.g. 10^{-4}).

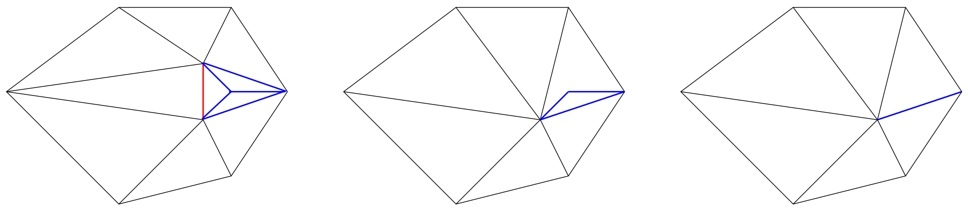


Fig. 14. A special case for the edge-contraction method. Left - the original mesh, the red edge is to be contracted; Middle - the red edge is contracted; Right - the additional triangles are removed.

We perform iterative edge contractions to improve the worst edge-ratio. For each iteration, we remove the tetrahedron with the maximum edge-ratio by contracting the shortest edge. We keep removing the tetrahedron with the maximum edge-ratio until the maximum edge-ratio is below the given threshold. During the edge contraction, we merge an interior vertex to a boundary vertex, an interior vertex to another interior vertex, or a boundary vertex to another boundary vertex. A special case may occur when we contract edges as shown in Figure 14, where two blue triangles (left picture) degenerate into the same triangle, and which should also be removed from the mesh. The special case can be detected by checking the number of elements sharing the same edge (2D) or face (3D). If there are more than two elements sharing the same edge/face, then the special case occurs. This can also be removed by contracting one shared edge. If edge contraction is not enough to arrive the threshold, then the longest edge bisection method is used to continue reducing the largest edge-ratio. But the number of vertices and the number of elements will increase.

In the mesh extraction process, a pyramid/diamond may be generated when we analyze the sign change edge/the interior edge in boundary cells. There are two or three ways to decompose a pyramid/diamond into tetrahedra, we compare the worst Joe-Liu parameters for different splitting ways and choose the better one. This process is executed when we extract meshes, therefore our mesh generation method tends to produce meshes with good overall quality.

Finally smoothing techniques are used to improve the Joe-Liu parameter and the minimum volume. The simplest discretization of the Laplacian operator for an interior node is the average of all its neighbors. The new position of a boundary vertex is the average of all its boundary neighbors derived from the discretized Laplace-Beltrami operator [39] [67]. Laplacian smoothing is an efficient heuristic, but it is possible to produce an invalid mesh containing inverted elements or elements with negative volume. We choose a ‘smart’ Laplacian smoothing [21], which relocates the point only if the quality of the local mesh is improved.

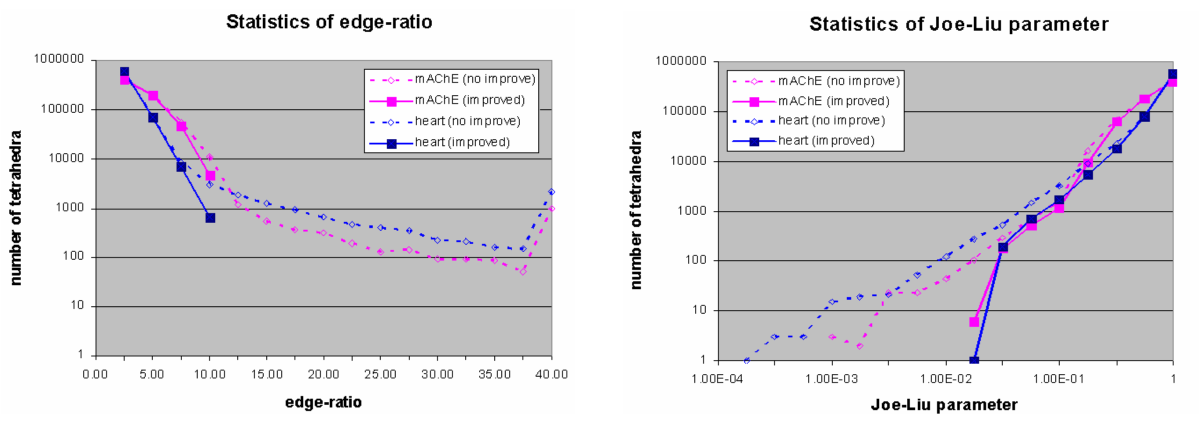


Fig. 15. The histogram of edge-ratios (left) and Joe-Liu parameters (right) for mAChE and the human heart. The number of tetrahedra at edge-ratio 40.0 represents the number of all the elements, whose edge-ratios are greater than 40.0.

The histogram of edge-ratios and Joe-Liu paramters (Figure 15) shows the overall quality of extracted tetrahedral meshes for a biomolecule mAChE (Figure 18) and the heart model (Figure 19). By comparing the two quality metrics before and after applying techniques of quality improvement, we can see the worst parameters are improved significantly. Figure 16 shows the improvement of the worst values of the edge-ratio, the Joe-Liu parameter and the minimum volume.

	Vertex Number	Tetra Number	Edge-ratio (best, worst)	Joe-Liu (best, worst)	Volume (minimal, maximal)
mAChE[b]	124180	670642	$(1.03, 1.19\times10^{4})$	$(1.0, 6.24\times10^{-4})$	$(2.43\times10^{-7}, 9.40\times10^{6})$
mAChE[a]	121670	656823	(1.03, 8.5)	$(1.0, 1.73\times10^{-2})$	$(1.40\times10^{-4}, 9.40\times10^{6})$
Heart[b]	140425	689020	$(1.02, 1.20\times10^{5})$	$(1.0, 1.23\times10^{-4})$	$(4.88\times10^{-8}, 2.56\times10^{2})$
Heart[a]	138072	676494	(1.02, 8.5)	$(1.0, 1.99\times10^{-2})$	$(4.39\times10^{-4}, 4.31\times10^{2})$

Fig. 16. The comparison of the three quality criteria (the edge-ratio, the Joe-Liu parameter and the minimal volume) before/after the quality improvement for mAChE and the human heart model. DATA[b] – before quality improvement; DATA[a] – after quality improvement.

8 Results and Applications

We have developed an interactive program for 3D mesh extraction and rendering from volume data. In the program, error tolerances and isovalues can be changed interactively. Our results were computed on a PC equipped with a Pentium III 800 MHz processor and 1 GB main memory.

Our algorithm has been used to generate tetrahedral meshes for a molecular dataset (mAChE) and the human heart model in two projects. We also tested our algorithm on volumetric data from CT scans, the *UNC Human Head* and *Poly (heart valve)*, and signed distance volumes generated from the polygonal surfaces of a human head and a knee. Figure 17, 18, 19, 1, 20, 21 and 22 provide information about datasets and test results. The results consist of the number of tetrahedra, extraction time, and corresponding images with respect to different isovalues and error tolerances. As a preprocessing, we calculate min/max values for each octree cell to visit only cells contributing to mesh extraction and to compute QEF values only in those cells at run time. Extraction time in the table includes octree traversal, QEF computation and actual mesh extraction, given isovalues and error tolerance values for inner and outer surfaces as run time parameters. If we fix isovalues, and change error tolerance interactively, the computed QEF is reused and thus the whole extraction process is accelerated. The results show that the mesh extraction time scales linearly with the number of elements in the extracted mesh.

Figure 18 shows the extracted adaptive tetrahedral mesh of mAChE, which has been used as the geometric model in solving the steady-state Smoluchowski equation to calculate ligand binding rate constants using the finite element method (FEM) [58] [57]. The most important part in the geometric structure of mAChE is the cavity, where fine meshes are required. We first find the position of the cavity, then control the adaptivity according to it. The area of the cavity is kept the finest level, while coarser meshes are obtained everywhere else. After improving the mesh quality, convergent results have been obtained in the finite element calculation, and they match with some experimental results well.

A good geometric model of the human heart is important for the simulation of the human cardiovascular system, which is very useful for the predictive medicine applications in cardiovascular

Dataset	Type	Size	Number of Tetrahedra (Extraction Time (unit : ms))			
			(a)	(b)	(c)	(d)
mAChE	Given	257^3	670642 (10891)	–	–	–
Heart	SDF	257^3	689020 (11110)	–	–	–
Skin	CT	129^3	935124 (17406)	545269 (10468)	–	–
Skull	CT	129^3	–	–	579834 (10203)	166271 (3063)
Poly	CT	257^3	–	276388 (5640)	63325 (1672)	14204 (672)
Head	SDF	65^3	143912 (2547)	76218 (1391)	40913 (766)	10696 (203)
Knee	SDF	65^3	70768 (1360)	94586 (1782)	93330 (1750)	72366 (1406)

Fig. 17. Datasets and Test Results. The CT data sets are re-sampled to fit into the octree representation. Rendering results for each case is shown in Figure 18, 19, 1, 20, 21 and 22. the Skull and Skin isosurfaces are extracted from the UNC Head model.

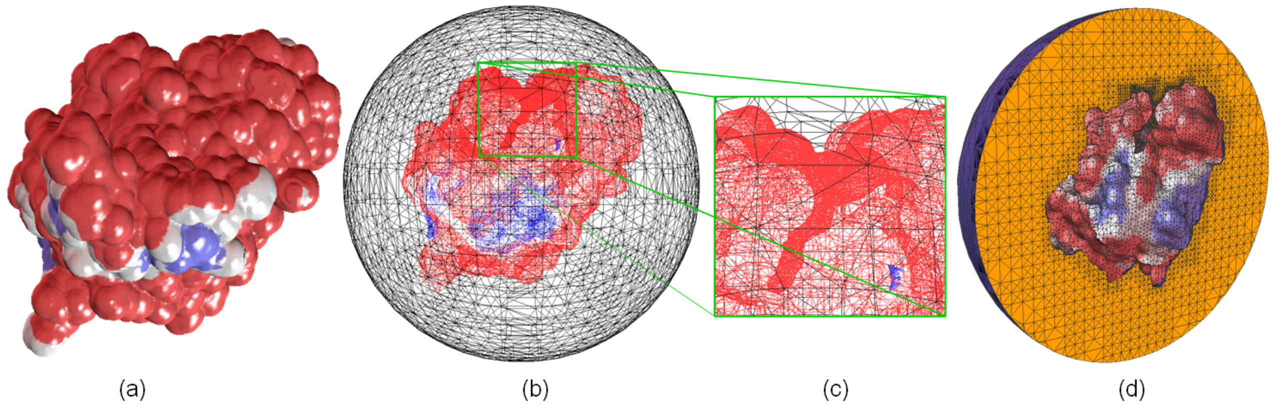


Fig. 18. mAChE – (a): the isosurface at 0.5 from the accessibility function; (b): the wire frame of the isosurface and an outer sphere; (c): a zoomed-in picture to show the refined cavity; (d): the adaptive tetrahedral mesh of the volume between the isosurface and an outer sphere. The color on the isosurface represents the distribution of the potential function, the color map is: $(-\infty, -0.5)$ - red; $[-0.5, 0.5]$ - white; $(0.5, +\infty)$ - blue. Note the region around the cavity has fine meshes, while other areas have relatively coarse meshes.

surgery. To extract 3D meshes from the surface heart model provided by New York University, we computed the signed distance function from the surface data and performed the mesh extraction (Figure 19). An adaptive and quality tetrahedral mesh is extracted with correct topology and feature sensitive adaptation. Meshes are refined in the areas of heart valves, while coarse meshes are kept for other regions.

Figure 20 and 21 show the resulting tetrahedral meshes extracted from two signed distance function data with the size of 65^3. The results from CT data are shown in Figure 1 (skull, skin) and 22 (heart valve). The number of elements in the extracted mesh is controlled by changing error tolerance. It is clear that adaptive tetrahedral meshes are extracted from the interval volume, and facial features are identified sensitively and preserved (Figure 20). In Figure 21, the sequence of

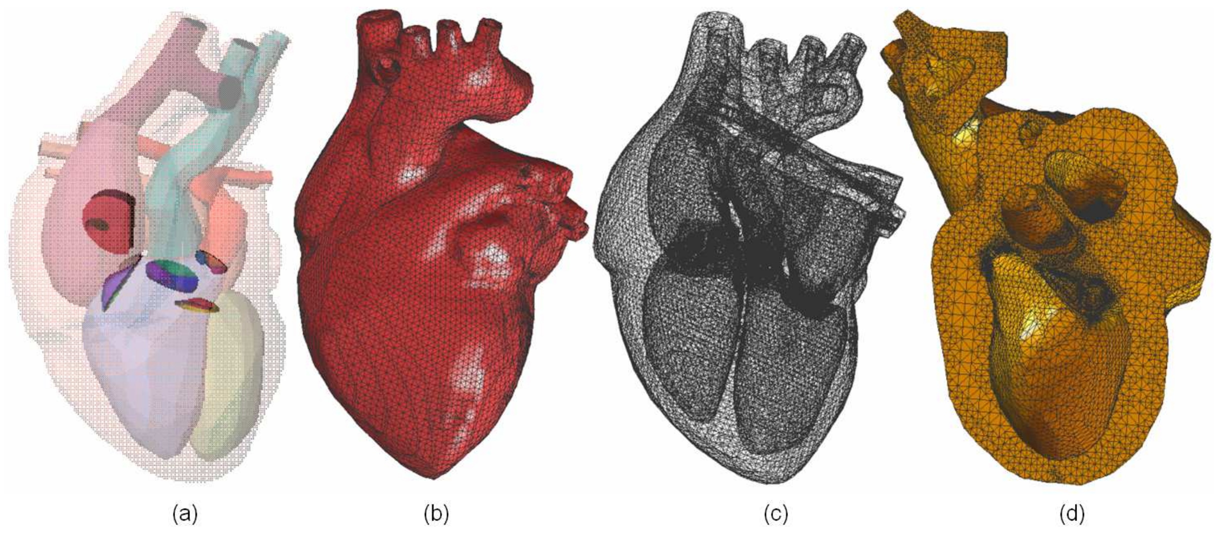


Fig. 19. Heart (SDF) – (a): the triangular surface of a human heart with valves (data courtesy from New York University (NYU)); (b): The tetrahedral mesh extracted from SDF of the heart surface. The smooth surface and the wire frame on the mesh is rendered; (c): the wire frame of the extracted heart model. Note that the region of heart valves are refined; (d): Cross-section of the adaptive tetrahedral mesh.

images are generated by changing the isovalue of the inner isosurface. The topology of the inner isosurface can change arbitrarily.

9 Conclusion

We have presented an algorithm to extract adaptive and quality 3D meshes directly from volumetric imaging data. By extending the dual contouring method described in [30], our method can generate 3D meshes with good properties such as no hanging nodes, sharp feature preservation and good aspect ratio. Using an error metric which is normalized by the function gradient, the resolution of the extracted mesh is adapted to the features sensitively. The resulting meshes are useful for efficient and accurate finite element calculations.

Acknowledgments

An early version of this paper appeared in *ACM* SM03 [68]. We thank Qiu Wu, and Anthony Thane for the use of their volume diffusion and rendering program, Jianguang Sun for the system management, Zeyun Yu and Julio Castrillon for useful discussions, Shashank G. Khandelwal for proofreading, and Nathan Baker, NYU, UNC for providing access to the molecule mAChE, the surface mesh of the heart model (With permission of New York University, Copyright 1994-2003), CT volume dataset of a human head respectively. This work was supported in part by NSF grants ACI-9982297, CCR-9988357, a DOE-ASCI grant BD4485-MOID from LLNL/SNL and from grant UCSD 1018140 as part of NSF-NPACI, Interactive Environments Thrust.

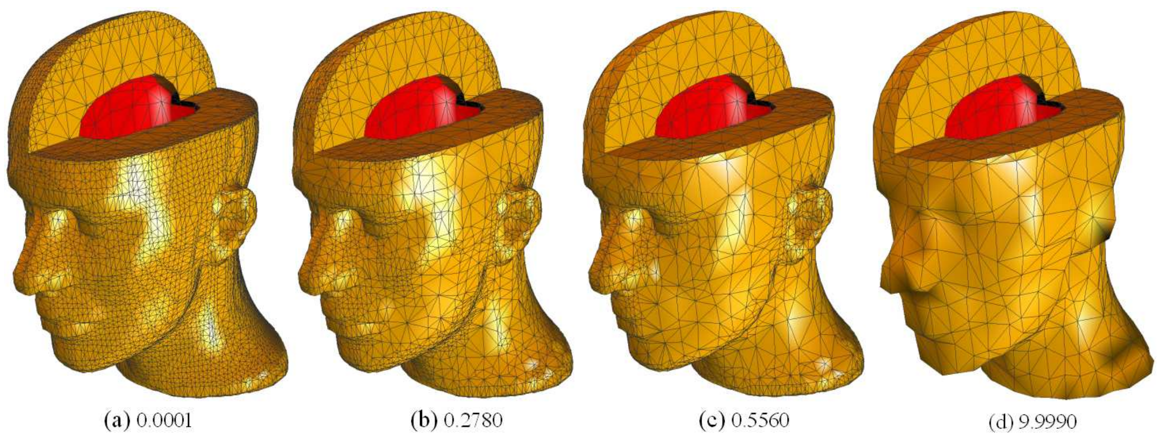


Fig. 20. Head (SDF) – Isovalues α_{in} = -9,17464, α_{out} = 0.0001; error tolerances ε_{in} = 1.7, ε_{out} are listed below each picture.

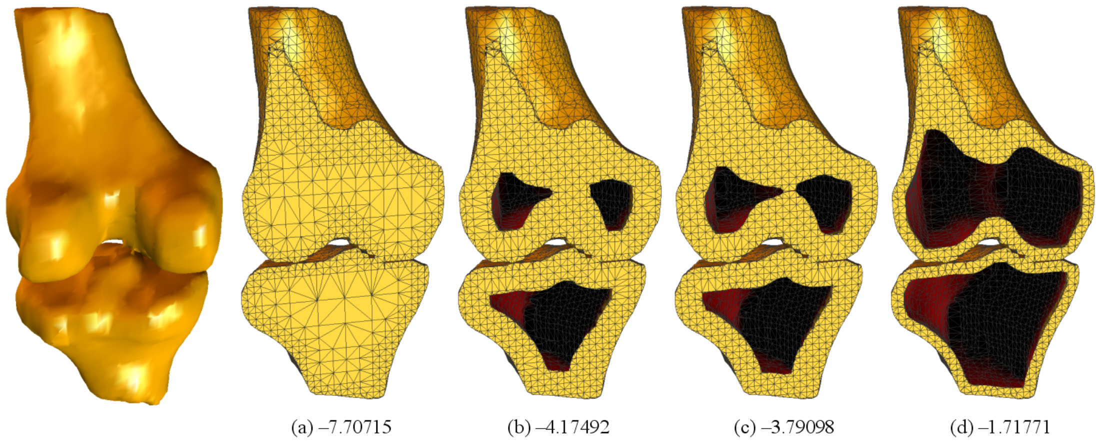


Fig. 21. Knee (SDF) – Error tolerances $\varepsilon_{in} = \varepsilon_{out}$ = 0.0001; isovalues α_{out} = -0.02838, α_{in} are listed below each picture.

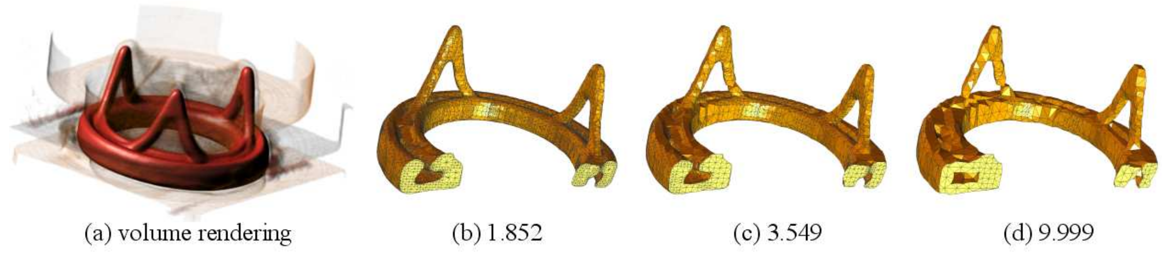


Fig. 22. Heart Valve (Poly) – Isovalues α_{in} = 1000.0, α_{out} = 75.0; error tolerances ε_{in} = 0.0001, ε_{out} are listed below each picture.

References

[1] C. Bajaj, E. Coyle, and K-N. Lin. Arbitrary topology shape reconstruction from planar cross sections. *Graphical Modeling and Image Processing*, 58(6):524–543, 1996.

[2] C. Bajaj, E. Coyle, and K-N. Lin. Tetrahedral meshes from planar cross sections. *Computer Methods*

in Applied Mechanics and Engineering, 179:31–52, 1999.

[3] C. Bajaj, V. Pascucci, and D. Schikore. Fast isocontouring for improved interactivity. In *Proceedings of the IEEE Symposium on Volume Visualization*, pages 39–46, 1996.

[4] C. Bajaj, V. Pascucci, and D. Schikore. The contour spectrum. In *IEEE Visualization*, pages 167–174, 1997.

[5] C. Bajaj, Q. Wu, and G. Xu. Level set based volume anisotropic diffusion. In *ICES Technical Report 301, The Univ. of Texas at Austin*, 2002.

[6] C. Bajaj and G. Xu. Anisotropic diffusion of subdivision surfaces and functions on surfaces. *ACM Transactions on Graphics*, 22(1):4–32, 2003.

[7] M. W. Bern, D. Eppstein, and J. R. Gilbert. Provably good mesh generation. In *IEEE Symposium on Foundations of Computer Science*, pages 231–241, 1990.

[8] T. D. Blacker and R. J. Myers. Seams and wedges in plastering: a 3d hexahedral mesh generation algorithm. *Engineering With Computers*, 2:83–93, 1993.

[9] F. J. Bossen and P. S. Heckbert. A pliant method for anisotropic mesh generation. In *5th International Meshing Roundtable*, pages 63–76, 1996.

[10] S. A. Canann. Plastering and optismoothing: new approaches to automated, 3d hexahedral mesh generation and mesh smoothing. *Ph.D. Dissertation, Brigham Young University, Provo, UT*, 1991.

[11] S. A. Canann, J. R. Tristano, and M. L. Staten. An approach to combined laplacian and optimization-based smoothing for triangular, quadrilateral and quad-dominant meshes. In *7th International Meshing Roundtable*, pages 479–494, 1998.

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

[13] C. Charalambous and A. Conn. An efficient method to solve the minimax problem directly. *SIAM Journal of Numerical Analysis*, 15(1):162–187, 1978.

[14] S.-W. Cheng and T. K. Dey. Quality meshing with weighted delaunay refinement. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms (SODA)*, pages 137–146, 2002.

[15] S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S. Teng. Sliver exudation. *Proc. Journal of ACM*, 47:883–904, 2000.

[16] L. P. Chew. Guaranteed-quality delaunay meshing in 3d (short version). In *13th ACM Symposium on Computational Geometry*, pages 391–393, 1997.

[17] U. Clarenz, U. Diewald, and M. Rumpf. Nonlinear anisotropic diffusion in surface processing. In *Proceedings of the 11th IEEE Visualization*, pages 397–405, 2000.

[18] D. Eppstein. Linear complexity hexahedral mesh generation. In *Symposium on Computational Geometry*, pages 58–67, 1996.

[19] D. A. Field. Laplacian smoothing and delaunay triangulations. *Communications in Applied Numerical Methods*, 4:709–712, 1988.

[20] L. Freitag and C. Ollivier-Gooch. Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering*, 40:3979–4002, 1997.

[21] L. A. Freitag. On combining laplacian and optimization-based mesh smoothing techniqes. *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, pages 37–43, 1997.

[22] P. J. Frey, H. Borouchaki, and P.-L. George. Delaunay tetrahedralization using an advancing-front approach. In *5th International Meshing Roundtable*, pages 31–48, 1996.

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

[24] M. Garland and P. S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization*, pages 263–270, 1998.

[25] P. L. George. Tet meshing: Construction, optimization and adaptation. In *8th International Meshing Roundtable*, pages 133–141, 1999.

[26] P. L. George and H. Borouchaki. *Delaunay triangulation and meshing, application to finite elements*, pages 230–234, 1998.

[27] S. F. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *IEEE Visualization*, pages 23–30, 1998.

[28] B. Gregorski, M. Duchaineau, P. Lindstrom, V. Pascucci, and K. I. Joy. Interactive view-dependent extraction of large isosurfaces. In *IEEE Visualization*, pages 475–482, 2002.

[29] P. Hansbo. Generalized laplacian smoothing of unstructured grids. *Communications in Numerical Methods in Engineering*, 11:455–464, 1995.

[30] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *Proceedings of SIGGRAPH*, pages 339–346, 2002.

[31] L. Kobbelt, M. Botsch, U. Schwanecke, and H. Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of SIGGRAPH*, pages 57–66, 2001.

[32] R. Laramee and R. Bergeron. An isosurface continuity algorithm for super adaptive resolution data. *Advances in Modelling, Animation, and Rendering: Computer Graphics International (CGI)*, pages 215–237, 2002.

[33] E. B. De l'Isle and P. L. George. Optimization of tetrahedral meshes. *IMA Volumes in Mathematics and its Applications*, 75:97–128, 1995.

[34] A. Liu and B. Joe. Relationship between tetrahedron shape measures. *BIT*, 34(2):268–287, 1994.

[35] R. Lohner, K. Morgan, and O. C. Zienkiewicz. Adaptive grid refinement for compressible euler equations. *Accuracy Estimates and Adaptive Refinements in Finite Element Computations, I. Babuska et. al. eds. Wiley*, pages 281–297, 1986.

[36] R. Lohner and P. Parikh. Three dimensional grid generation by the advancing-front method. *International Journal for Numerical Methods in Fluids*, 8:1135–1149, 1988.

[37] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. In *IEEE Transactions on Visualization and Computer Graphics*, pages 19–26, 2000.

[38] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of SIGGRAPH*, pages 163–169, 1987.

[39] M. Meyer, M. Desbrun, P. Schroder, and A. H. Burr. Discrete differential-geometry operators for triangulated 2-manifolds. *VisMath'02, Berlin (Germany)*, 2002.

[40] S. A. Mitchell and S. A. Vavasis. Quality mesh generation in higher dimensions. *SIAM Journal on Computing*, 29(4):1334–1370, 2000.

[41] D. Moore. Compact isocontours from sampled data. *Graphics Gems III*, pages 23–28, 1992.

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

[43] G. M. Nielson and J. Sung. Interval volume tetrahedrization. In *IEEE Visualization*, pages 221–228, 1997.

[44] S. Owen. A survey of unstructured mesh generation technology. In *7th International Meshing Roundtable*, pages 26–28, 1998.

[45] V. Pascucci and C. Bajaj. Time critical adaptive refinement and smoothing. In *Proceedings of the ACM/IEEE Volume Visualization and Graphics Symposium*, pages 33–42, 2000.

[46] M. A. Price and C. G. Armstrong. Hexahedral mesh generation by medial surface subdivision: Part i. *International Journal for Numerical Methods in Engineering*, 38(19):3335–3359, 1995.

[47] M. A. Price and C. G. Armstrong. Hexahedral mesh generation by medial surface subdivision: Part ii. *International Journal for Numerical Methods in Engineering*, 40:111–136, 1997.

[48] R. Schneiders. A grid-based algorithm for the generation of hexahedral element meshes. *Engineering With Computers*, 12:168–177, 1996.

[49] R. Schneiders. An algorithm for the generation of hexahedral element meshes based on an octree technique. In *6th International Meshing Roundtable*, pages 195–196, 1997.

[50] E. Seveno. Towards an adaptive advancing front method. In *6th International Meshing Roundtable*, pages 349–362, 1997.

[51] M. S. Shephard and M. K. Georges. Three-dimensional mesh generation by finite octree technique. *International Journal for Numerical Methods in Engineering*, 32:709–749, 1991.

[52] J. R. Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *Proceedings of the 4th Annual Symposium on Computational Geometry, Association for Computational Machinery*, pages 86–95, 1998.

[53] J. R. Shewchuk. Constrained delaunay tetrahedrizations and provably good boundary recovery. In *11th International Meshing Roundtable*, pages 193–204, 2002.

[54] J. R. Shewchuk. Two discrete optimization algorithms for the topological improvement of tetrahedral meshes. In *Unpublished manuscript*, 2002.

[55] J. R. Shewchuk. What is a good linear element? interpolation, conditioning, and quality measures. In *11th International Meshing Roundtable*, pages 115–126, 2002.

[56] K. Shimada, A. Yamada, and T. Itoh. Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles. In *6th International Meshing Roundtable*, pages 375–390, 1997.

[57] Y. Song, Y. Zhang, C. L. Bajaj, and N. A. Baker. Continuum diffusion reaction rate calculations of wild type and mutant mouse acetylcholinesterase: Adaptive finite element analysis. *Accepted in Biophysical Journal*, 2004.

[58] Y. Song, Y. Zhang, T. Shen, C. L. Bajaj, J. A. McCammon, and N. A. Baker. Finite element solution of the steady-state smoluchowski equation for rate constant calculations. *Biophysical Journal*, 86(4):1–13, 2004.

[59] Gabriel Taubin. Distance approximations for rasterizing implicit curves. *ACM Transactions on Graphics*, 14(3–42), 1994.

[60] T. J. Tautges, T. Blacker, and S. Mitchell. The whisker-weaving algorithm: a connectivity based method for constructing all-hexahedral finite element meshes. *International Journal for Numerical Methods in Engineering*, 39:3327–3349, 1996.

[61] S.-H. Teng and C. W. Wong. Unstructured mesh generation: Theory, practice, and perspectives. *International Journal of Computational Geometry and Applications*, 10(3):227–266, 2000.

[62] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proceedings of IEEE International Conference on Computer Vision*, pages 839–846, 1998.

[63] J. Weickert. Anisotropic diffusion in image processing. Teubner Verlag, Stuttagart, 1998.

[64] R. Westermann, L. Kobbelt, and T. Ertl. Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, 15(2):100–111, 1999.

[65] J. Wilhelm and A. Van Gelder. Octrees for faster isosurface generation. In *ACM Transactions on Graphics*, pages 57–62, 1992.

[66] Z. Wood, M. Desbrun, P. Schroder, and D.E. Breen. Semi-regular mesh extraction from volumes. In *Visualization Conference Proceedings*, pages 275–282, 2000.

[67] G. Xu, Q. Pan, and C. Bajaj. Discrete surface modeling using geometric flows. *Technical Report TR-03-36, Department of Computer Sciences, University of Texas at Austin*, 2003.

[68] Y. Zhang, C. Bajaj, and B-S Sohn. Adaptive and quality 3d meshing from imaging data. In *ACM Symposium on Solid Modeling and Applications*, pages 286–291, 2003.

[69] Y. Zhou, B. Chen, and A. Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In *Proceedings of the 8th IEEE Visualization*, pages 135–142, 1997.