Available online at www.sciencedirect.com
SCIENCE DIRECT®
Comput. Methods Appl. Mech. Engrg. 195 (2006) 942–960
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

# Adaptive and quality quadrilateral/hexahedral meshing from volumetric data ☆

Yongjie Zhang, Chandrajit Bajaj *

*Computational Visualization Center, Department of Computer Sciences, Institute for Computational Engineering and Sciences, The University of Texas at Austin, TX 78712, USA*

Received 27 August 2004; received in revised form 1 February 2005; accepted 14 February 2005

**Abstract**

This paper describes an algorithm to extract adaptive and quality quadrilateral/hexahedral meshes directly from volumetric data. First, a bottom-up surface topology preserving octree-based algorithm is applied to select a starting octree level. Then the dual contouring method is used to extract a preliminary uniform quad/hex mesh, which is decomposed into finer quads/hexes adaptively without introducing any hanging nodes. The positions of all boundary vertices are recalculated to approximate the boundary surface more accurately. Mesh adaptivity can be controlled by a feature sensitive error function, the regions that users are interested in, or finite element calculation results. Finally, a relaxation based technique is deployed to improve mesh quality. Several demonstration examples are provided from a wide variety of application domains. Some extracted meshes have been extensively used in finite element simulations.


*Keywords:* Quadrilateral/hexahedral mesh; Topology preservation; Mesh adaptivity; Mesh quality

## 1. Introduction

Unstructured quadrilateral/hexahedral mesh generation attracts many researchers' interest because of its important applications in finite element simulations. However, it still remains a challenging and open problem to generate adaptive and quality quad/hex meshes directly from volumetric data, such as computed tomography (CT), magnetic resonance imaging (MRI) and signed distance function (SDF) data.

☆ Visit http://www.ices.utexas.edu/cvc/quadhex

* Corresponding author. Tel.: +1 512 471 8870; fax: +1 512 471 8694.
*E-mail addresses:* jessica@ices.utexas.edu (Y. Zhang), bajaj@cs.utexas.edu (C. Bajaj).

0045-7825/$ - see front matter 
doi:10.1016/j.cma.2005.02.016

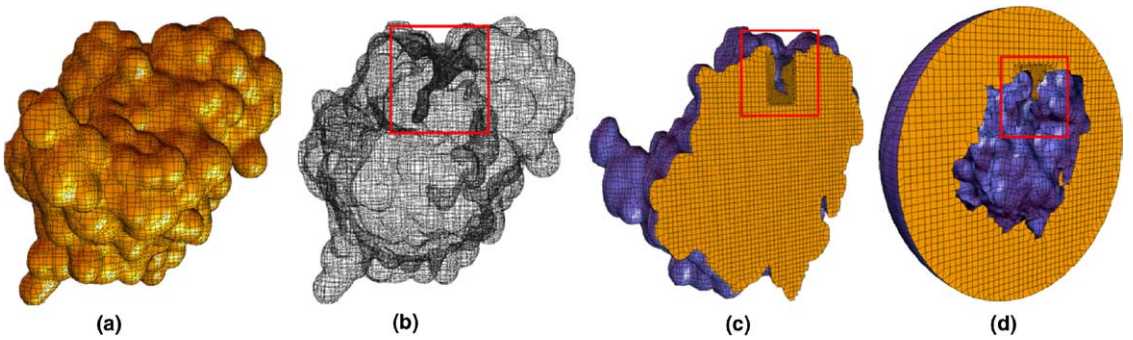


Fig. 1. Adaptive quadrilateral and hexahedral meshes of a biomolecule mAChE: (a) the quadrilateral mesh of the molecular surface; (b) the wireframe of the adaptive quadrilateral mesh of the molecular surface; (c) the adaptive hexahedral mesh of the interior volume; (d) the adaptive hexahedral mesh of the exterior volume between the molecular surface and an outer sphere. Finer meshes are generated in the region of the cavity, while coarser meshes are kept in other areas. The cavity is shown in the red boxes.

The volumetric data $V$ is a sequence of sampled functional values on rectilinear grids, and can be written as $V = \{F(i,j,k)|i,j,k$ are indices in $x$, $y$, $z$ coordinates in a rectilinear grid$\}$. An isosurface or a level set corresponding to the isovalue $\alpha$ is defined as $S_F(\alpha) = \{(x,y,z)|F(x,y,z) = \alpha\}$, and an interval volume between two isosurfaces $S_F(\alpha_1)$, $S_F(\alpha_2)$ is defined as $I_F(\alpha_1,\alpha_2) = \{(x,y,z)|\alpha_1 \leqslant F(x,y,z) \leqslant \alpha_2\}$. In this paper, we present an approach to extract adaptive and quality quadrilateral meshes for an isosurface $S_F(\alpha)$ with correct topology, and hexahedral meshes for an interval volume $I_F(\alpha_1,\alpha_2)$ with isosurfaces as boundaries. In certain finite element simulations, both interior and exterior hexahedral meshes are required, for example, the interior mesh of the volume inside the solvent accessibility surface of the biomolecule mouse acetylcholinesterase (mAChE) [31,30], and the exterior mesh between the solvent accessibility surface and an outer bounding sphere. Since the most important part in the geometric structure of mAChE is the cavity, we need to generate finer mesh for it (Fig. 1). Our approach can also generate adaptive and quality interior and exterior hexahedral meshes.

The main steps to extract adaptive and quality quadrilateral and hexahedral meshes from volumetric data are as follows:

(1) The selection of a starting octree level for uniform mesh generation with correct topology.
(2) Crack-free and adaptive quad/hex meshing without any hanging nodes.
(3) Quality improvement.

In order to generate uniform quadrilateral and hexahedral meshes with correct topology, we select a suitable starting octree level using a bottom-up surface topology preserving octree-based algorithm. An approach provided in [15] is used to check whether a fine isosurface is topologically equivalent to a coarse one or not. Generally correct topology is guaranteed in the uniform mesh.

The dual contouring method [15] proposes an algorithm to extract a uniform quadrilateral mesh for an isosurface by analyzing each *sign change edge*, whose two ending points lie in different sides of the isosurface. In the octree-based data structure, each sign change edge is shared by four octree leaves, and one minimizer point is obtained for each leaf cell by minimizing a pre-defined quadratic error function (QEF) [14]. The QEF is defined as follows:

$$\mathrm{QEF}[x] = \sum_i (n_i \cdot (x - p_i))^2, \tag{1}$$

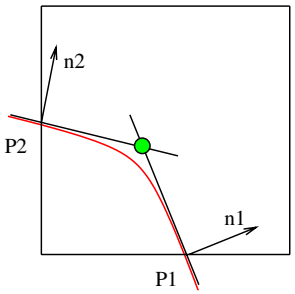


Fig. 2. The quadratic error function (QEF) and the minimizer point in 2D. The red[1] curve is an isocontour, and the green point is the minimizer point calculated in Eq. (1). $(p_1, n_1)$ and $(p_2, n_2)$ represent the position and unit normal vectors of the two intersection points.

where $p_i$, $n_i$ represent the position and unit normal vectors of the intersection point respectively. Fig. 2 shows one 2D example. The four minimizer points construct a quad, and the union of all the generated quads provides an approximation to this isosurface.

Starting from a uniform quadrilateral mesh, we use templates to refine each quad adaptively. The position of each vertex is recalculated by moving it toward the isosurface along its normal direction, which is represented by trilinear interpolation functions within octree leaf cells. The dual contouring isosurface extraction method has been extended to uniform hexahedral mesh generation [38,39]. In this paper, pre-defined three dimensional templates are used to generate adaptive hexahedral meshes.

The mesh adaptivity can be controlled according to various requirements by a feature sensitive error function [38,39], areas that users are interested in, or results from finite element calculations. Users can also design an error function to control the mesh adaptivity according to their specific requirements.

Generally, the extracted quadrilateral and hexahedral meshes can not be used for finite element calculations directly since some elements have poor quality. We choose corresponding metrics to measure the quality of quadrilateral and hexahedral meshes respectively, and deploy a relaxation based technique to improve mesh quality. Several of our generated meshes have been used in finite element simulations.

The remainder of this paper is organized as follows: Section 2 reviews the related work on quadrilateral/hexahedral mesh generation; Section 3 describes how to choose the starting octree level; Sections 4 and 5 explain the detailed algorithm for extracting adaptive quadrilateral and hexahedral meshes; Section 6 talks about three ways to control the mesh adaptivity; Section 7 discusses the mesh quality improvement; Section 8 shows some results and applications; the final section presents our conclusions.

## 2. Previous work

As a structured method, quad/hex mapped meshing [9] generates the most desirable meshes if opposite edges/faces of the domain to be meshed have equal numbers of divisions or the same surface mesh. However, it is always difficult to decompose an arbitrary geometric configuration into mapped meshable regions. In the CUBIT project [1] at Sandia National Labs, a lot of research has been done to automatically recognize features and decompose geometry into mapped meshable areas or volumes.

As reviewed in [23,35], there are indirect and direct methods for unstructured quad/hex mesh generation. The indirect method is to generate triangular/tetrahedral meshes first, then convert them into quads/hexes. The direct method is to generate quads/hexes directly without first going through triangular/tetrahedral meshing.

[1] For interpretation of color in figures, the reader is referred to the Web version of this article.

### 2.1. Unstructured quad mesh generation

The indirect method is to convert triangles into quads by dividing a triangle into three quads, or combining adjacent pairs of triangles to form quads [20].

There are three main categories for unstructured direct quad mesh generation: quad meshing by decomposition, advancing front quad meshing and isosurface extraction. The decomposition technique divides the domain into simpler regions which can be resolved by templates [2,33]. The second category is to utilize a moving front method for direct placement of nodes and elements. Starting with an initial placement of nodes on the boundary, Zhu et al. [40] formed individual elements by projecting edges towards the interior. As a part of CUBIT [1], the paving algorithm places elements starting from the boundary and works in [5]. Different from the decomposition and the advancing front techniques, the dual contouring method [15] extracts uniform quadrilateral meshes from volumetric data to approximate isosurfaces which can be an arbitrary geometry.

### 2.2. Unstructured hex mesh generation

Eppstein [10] started from a tetrahedral mesh to decompose each tetrahedron into four hexahedra. Although this method avoids many difficulties, it rapidly increases the number of elements and tends to introduce poorly shaped elements.

There are five distinct methods for unstructured direct all-hex mesh generation: grid-based, medial surface, plastering, whisker weaving and isosurface extraction. The grid-based approach generates a fitted 3D grid of hex elements on the interior of the volume, and hex elements are added at the boundaries to fill gaps [26,28,29]. The grid-based method is robust, but tends to generate poor quality elements at the boundaries. Medial surface methods decompose the volume into map-meshable regions, and fill the volume with hex elements using templates [24,25]. Plastering places elements on boundaries first and advances towards the center of the volume [6,4]. Whisker weaving first constructs 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 [34]. Medial surface methods, plastering and whisker weaving have successfully generated hex meshes for some geometry, but have not been proven to be robust and reliable for an arbitrary geometric domain. Zhang et al. [38,39] extended the dual contouring isosurface extraction method [15] to uniform hexahedral mesh generation. This method is robust and reliable for an arbitrary geometry, but adaptive meshes are preferable and mesh quality needs to be improved.

### 2.3. Quality improvement

As the simplest and most straight forward method, Laplacian smoothing relocates the vertex position at the average of the nodes connecting to it [11]. There are a variety of other smoothing techniques based on a weighted average of the surrounding nodes and elements. The averaging method may invert or degrade the local quality, but it is simple to implement and in wide use. Instead of relocating vertices based on a heuristic algorithm, people utilized an optimization technique to improve 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 problems [8]. Optimization-based smoothing yields better results but it is more expensive than Laplacian smoothing. Some papers [7,12,13] recommended a combined Laplacian/optimization-based approach.

Staten et al. [32,16] proposed algorithms to improve node valence for quadrilateral meshes. One special case of cleanup in hexahedral meshes for the whisker weaving algorithm is presented in [21]. Schneiders [27] proposed algorithms and a series of templates for quad/hex element decomposition. A recursive subdivision algorithm was proposed for the refinement of hex meshes [3].

## 3. Starting octree level selection

There are three main steps in our adaptive and quality quadrilateral and hexahedral mesh extraction from volumetric data. First, we need to choose a suitable starting octree level to generate the uniform mesh with correct topology. Then pre-defined templates are used to refine the uniform mesh adaptively. The positions of all boundary vertices are recalculated, and the mesh adaptivity can be controlled by an error function designed in multiple ways. Finally, the relaxation based technique is used to improve mesh quality.

The bottom-up surface topology preserving octree-based algorithm is used to select a starting octree level. Suppose the volume data has the dimension of $(2^n + 1)^3$, so the deepest octree level is $n$. For an isosurface, we first compare the surface topology at Level $n$ and Level $(n - 1)$. If the surface topology is equivalent, then we continue comparing the surface topology at Level $(n - 1)$ and Level $(n - 2)$ until we find the surface topology at two neighboring levels, e.g. Level $i$ and Level $(i - 1)$ $(i = n, \ldots, 1)$, is different from each other. Then we will select $i$ as the starting octree level.

We assign a sign to each grid point in the volumetric data. If the function value at a grid point is greater than the isovalue, then the sign is 1, otherwise it is 0. An approach is described in [15] to check whether a fine isocontour is topologically equivalent to a coarse one or not. The fine and coarse isocontour is topologically equivalent with each other if and only if the sign of the middle vertex of a coarse edge/face/cube is the same as the sign of at least one vertex of the edge/face/cube which contains the middle vertex. Generally we guarantee the correct topology for the boundary surfaces by choosing a suitable starting octree level, and correct topology will be preserved in the process of adaptive mesh refinement.

## 4. Quad isosurface extraction

Finite element calculations sometimes require quadrilateral meshes instead of triangular meshes. It is more challenging to generate quadrilateral meshes since not every polygon can be decomposed into quads directly. The uniform quadrilateral mesh extraction algorithm is simpler [15], but adaptive meshes are preferable over uniform ones. There are two main problems in adaptive quadrilateral mesh extraction.

(1) How to decompose a quad into finer quads?
(2) How to calculate the positions of vertices?

### 4.1. Mesh decomposition

#### 4.1.1. Indirect method

In the dual contouring isosurface extraction method [15], an error function is defined to control where we should generate fine meshes, and where we should keep coarse ones. In the adaptive octree data structure, either a sign change edge is shared by three cells resulting in a triangle, or it is shared by four cells and a quad is generated. Therefore, the isosurface is represented by a union of quads and triangles. In order to obtain an all-quad mesh, the indirect method splits each quad into four quads and each triangle into three quads by inserting points at the middle of edges and at the center of the element as shown in Fig. 3. The idea of the indirect method is simple and easy to implement, but the number of elements increases by a factor of 2–3 over the original mesh.

#### 4.1.2. Direct method

At the selected starting octree level, the dual contouring isosurface extraction method [15] generates uniform quadrilateral meshes by analyzing each sign change edge which is shared by four leaf cells. Adaptive

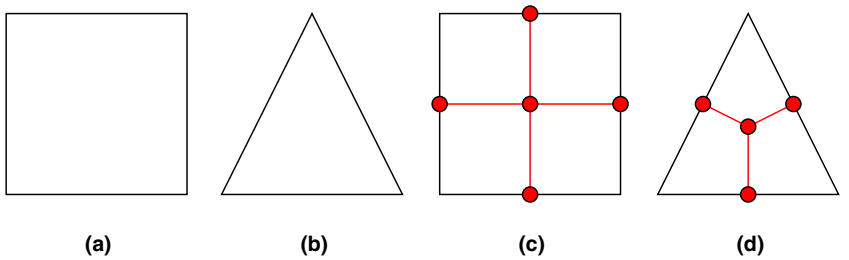


Fig. 3. The templates to decompose a quad or a triangle into quads. Red points are newly inserted at the middle of edges or the element center. (a) A quad before splitting; (b) a triangle before splitting; (c) a quad is split into four quads; (d) a triangle is split into three quads.

quadrilateral meshes can be obtained from the uniform mesh by using some templates. There are multiple ways to define templates for adaptive quadrilateral mesh construction, therefore criteria needs to be set to evaluate them in order to generate meshes with good quality. Here we define some requirements for templates:

(1) All resulting elements are quads.
(2) No hanging nodes exist.
(3) The resulting mesh approximates the object surface accurately.
(4) The resulting elements have good aspect ratio.
(5) The resulting mesh introduces a small number of new elements and vertices.

Fig. 4 shows three methods to define templates for adaptive quadrilateral mesh generation starting from a uniform mesh with correct topology. In the uniform case, each sign change edge is shared by four cells and four minimizer points are obtained to construct a quad. In Method 1, if the maximum error function

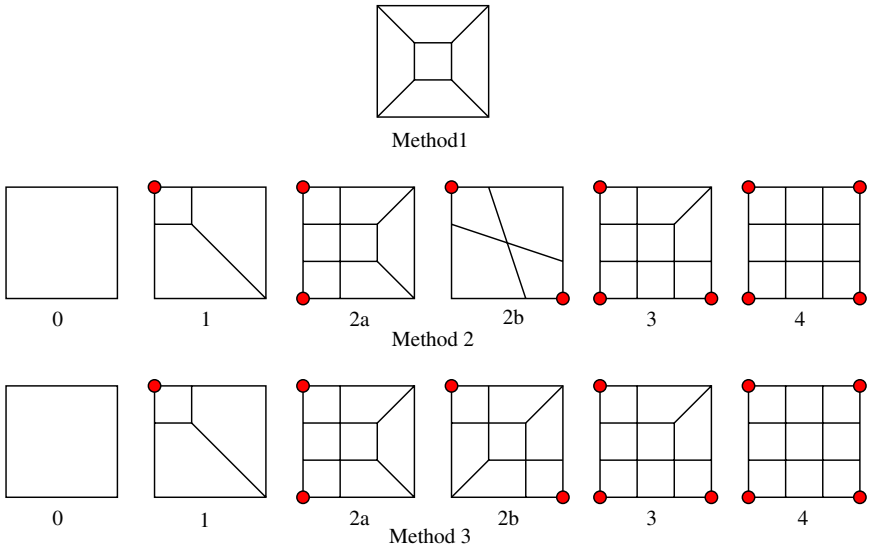


Fig. 4. Three different methods to define templates for adaptive quadrilateral isosurface extraction. In Method 1, the quad needs to be refined; in Methods 2 and 3, octree leaf cells generating red minimizer points need to be refined.

value (for example, the feature sensitive error function defined in [38,39]) of the four cells is greater than a threshold $\varepsilon$, then the four octree cells containing the sign change edge should be subdivided, and the quad generated from this edge should be refined. This method does not consider its neighboring information, each quad is refined independently. If a quad needs to be refined, then the resulting mesh has 5 elements and 4 newly inserted vertices. In Methods 2 [29] and 3, various decomposition methods are chosen according to the cell which generates a quad node and also needs to be refined. Methods 2 and 3 are only different in Case (2b), Method 2 generates less elements and vertices, but the quad quality is worse than in Method 3.

We can use the above five template requirements to compare the three methods in Fig. 4. It is obvious that all the three methods only generate quad elements, and no hanging nodes are introduced. Compared with Method 1, Methods 2 and 3 insert extra nodes on the quad edges as well as inside the quad, so they can approximate the surface more accurately. Comparing the worst aspect ratio of the resulting quad elements in Methods 2 and 3, we can see that Method 3 generates quads with better quality. The number of elements and the number of newly inserted vertices for each template are listed in Fig. 5. Method 3 is preferable by balancing the five criteria.

### 4.2. Vertex position calculation

In the process of mesh refinement, new vertices are inserted according to the pre-defined templates. The next step is to update the positions of existing vertices and calculate the positions of newly inserted vertices.

In Fig. 6, we assume that the leaf cell can be divided into four subcells in the finest resolution level, therefore the real isosurface (the red curve) is represented by a union of three trilinear interpolation functions within the subcells. For each existing minimizer point, first we find the octree leaf cell containing it in

| Method | Number of | 0 | 1 | 2a | 2b | 3 | 4 |
|---|---|---|---|---|---|---|---|
| 2 | elements | 1 | 3 | 7 | 4 | 8 | 9 |
| | vertices | 0 | 3 | 8 | 4 | 10 | 12 |
| 3 | elements | 1 | 3 | 7 | 7 | 8 | 9 |
| | vertices | 0 | 3 | 8 | 8 | 10 | 12 |

Fig. 5. The number of elements and the number of newly inserted vertices for templates in Methods 2 and 3 shown in Fig. 4.

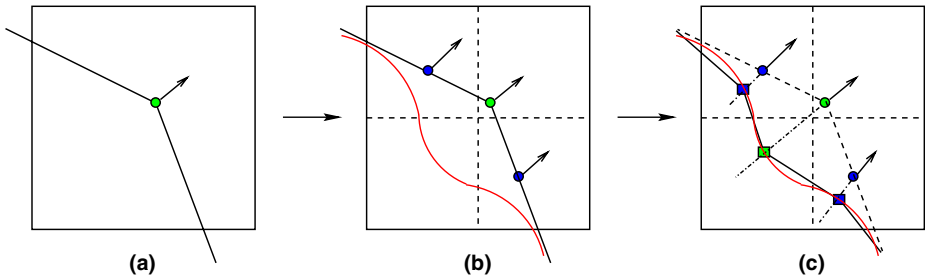


Fig. 6. The calculation of vertex positions: (a) one leaf cell; (b) the leaf cell is subdivided into four subcells, and three minimizer points are obtained; (c) the three minimizer points are moved to the isocontour along their normal directions. The red curve is the real isocontour. The green circle point represents an existing minimizer point of this leaf cell, and blue circle points are two newly inserted vertices. The arrows are their normal vectors, the green and blue box points are the resulting vertices.

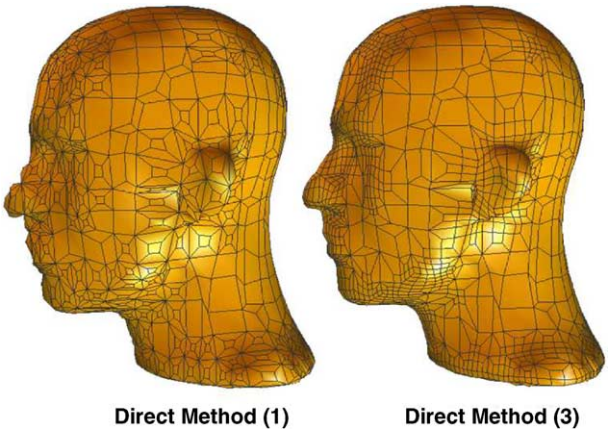


Fig. 7. Adaptive quad meshes generated from two direct methods. A feature sensitive error function [38,39] is chosen for mesh adaptivity, the isovalue $\alpha = 0$, the error tolerance $\varepsilon = 0.4$. Method 1 generates a poor nose, and Method 3 generates a better result.

the current resolution level, then move it toward the isosurface within this leaf cell along its normal direction. The intersection point is more accurate to represent this boundary vertex than the minimizer point. If the calculated intersection point lies outside this cell unfortunately because of bad normal vectors, we will still keep old position and normal vectors.

For those newly inserted vertices, we first calculate their position and normal vectors by linear interpolation of the four vertices of the original quad. Then we will move them toward the isosurfaces in the same way as we update the positions of existing vertices.

Fig. 7 shows adaptive quadrilateral meshes of the human head generated from two direct methods, Method 1 and Method 3 shown in Fig. 4. It is obvious that the original uniform mesh is refined adaptively, and the new vertex positions are closer to the isosurface. Method 1 generates a bad nose, and Method 3 approximates the isosurface more accurately than Method 1 because it introduces extra vertices on the refined edges of each original quad. The mesh adaptivity is controlled by a feature sensitive error function [38,39], which is sensitive to facial features such as the nose, the eyes, the mouth and the ears.

## 5. Hexahedral mesh extraction

The dual contouring method [15] has been extended to uniform hexahedral mesh generation by analyzing each interior vertex (a grid point inside the interval volume) shared by eight different cells, which are either boundary cells or interior cells [38,39]. A minimizer point is calculated for each boundary cell, and the cell center is set as the minimizer point for each interior cell. Those eight minimizer points construct a hexahedron. In this section, we will focus on adaptive hexahedral mesh generation.

### *5.1. 2D mesh decomposition*

In 2D, the uniform quadrilateral mesh can be constructed by analyzing each interior grid point, which is shared by four cells. One minimizer point is calculated for each cell, therefore four minimizer points are obtained and they construct a quad. All the templates defined in Fig. 4 can be used here for adaptive 2D mesh generation. Fig. 8 shows an example of adaptive quadrilateral mesh extraction using Method 3. When we analyze each cell to calculate the minimizer point, we compare the feature sensitive error

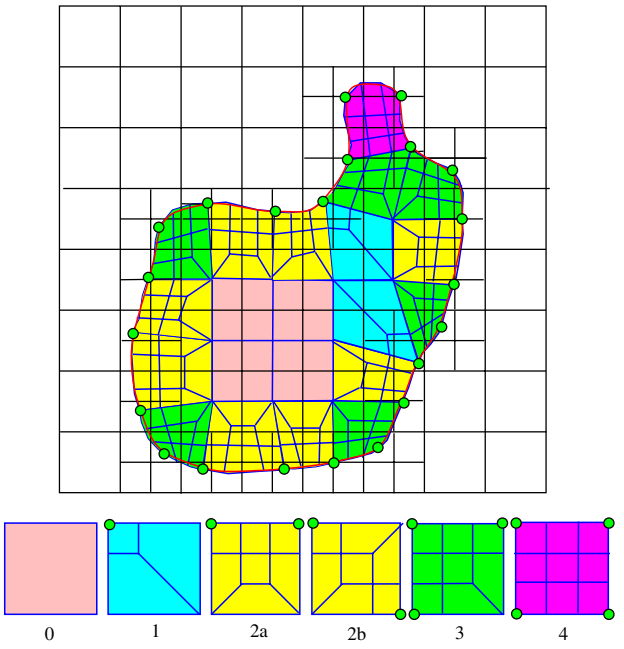


Fig. 8. Top row—an example of adaptive quad mesh generation in 2D. Each green point represents a minimizer point of a cell to be refined, and the red curve represents the real isocontour. Bottom row—the decomposition templates of Method 3 shown in Fig. 4.

function of this cell with a threshold $\varepsilon$. If the error function value of a cell is greater than $\varepsilon$, then this cell needs to be subdivided. An interior grid point is shared by four cells, therefore there are a total of $2^4 = 16$ configurations. Due to the symmetry, there are six basic templates for the quad refinement. A uniform quadrilateral mesh can be refined adaptively by using those templates.

### 5.2. 3D mesh decomposition

#### 5.2.1. Indirect method

Adaptive and quality tetrahedral meshes have been generated from volumetric imaging data [38,39], therefore we can obtain hexahedral meshes by decomposing each tetrahedron into four hexahedra.

#### 5.2.2. Direct method

Not all the direct methods for adaptive 2D mesh generation shown in Fig. 4 can be extended to 3D. There are two main methods for adaptive hexahedral mesh generation, one is extended from the first 2D direct method and the other one is derived from part of the third 2D direct method.

Extended from the first 2D direct method in Fig. 4, Method 1 refines each hexahedron independently as shown in Fig. 9. It first splits each hexahedron into seven hexahedra by inserting one small hex in its center, and each face of the original hex is contained in a hex independently. If one face needs to be refined, then the hex containing it will be refined as shown in the right picture of Fig. 9. If there are $i$ ($i = 1, \ldots, 6$) faces that need to be refined for a hexahedron, then the resulting mesh has $(6i + (6 - i) + 1 = 5i + 7)$ elements and $8(i + 1)$ newly inserted vertices.

Method 2 is derived from part of the third 2D direct method shown in Fig. 4. In the process of refinement, this method considers whether the error function value of each cell is greater than a threshold $\varepsilon$ or not. One hexahedron has a total of eight vertices, so there are ($2^8 = 256$) configurations. Due to symmetry,

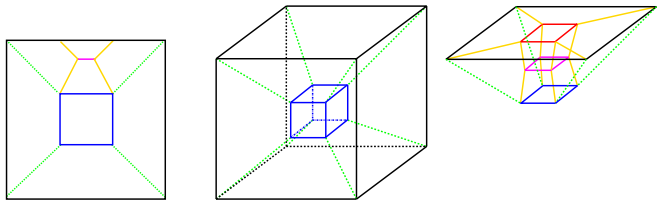

Fig. 9. Adaptive hexahedral mesh decomposition (Method 1): left—a 2D example; middle—a small hexahedron is inserted; right—the top face of the original hexahedron needs to be refined, 6 hex and 8 extra vertices are generated.

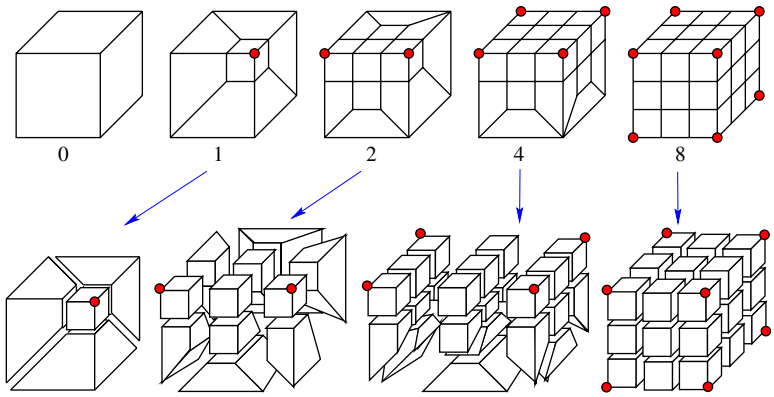


Fig. 10. Templates of adaptive hexahedral mesh decomposition (Method 2) according to the cells to be refined from which red minimizer points are generated. The bottom row shows the detailed decomposition format.

| Method | Number of | 0 | 1 | 2 | 4 | 8 |
|---|---|---|---|---|---|---|
| 2 | elements | 1 | 4 | 11 | 22 | 27 |
| | vertices | 0 | 7 | 19 | 39 | 56 |

Fig. 11. The element number and the newly inserted vertex number of Method 2 within refined hexahedra shown in Fig. 10.

there are only 22 unique templates [36], but only five templates are useful out of them because not all the templates can be decomposed into hexahedra. Fig. 10 shows the five templates for adaptive hexahedral decomposition together with a detailed view [29], which are much more complicated than the templates for 2D quadrilateral decomposition. Fig. 11 lists the number of elements and the number of newly inserted vertices for each template.

We set a sign for each leaf cell at the uniform starting octree level indicating if this cell needs to be refined or not. For each leaf cell, the feature sensitive error function is calculated and compared with a threshold $\varepsilon$. If the function value is greater than $\varepsilon$, then the sign is set to be 1, otherwise it is 0. For each hexahedron extracted from the uniform level, we check if it belongs to one of the templates shown in Fig. 10. If not, we need to convert it by looking up the table shown in Fig. 12. We keep updating the sign for each leaf cell until no sign changes, at this time all the generated hexahedra in the uniform level are in the format of the five templates shown in Fig. 10, then we can construct an adaptive hexahedral mesh using the corresponding templates.

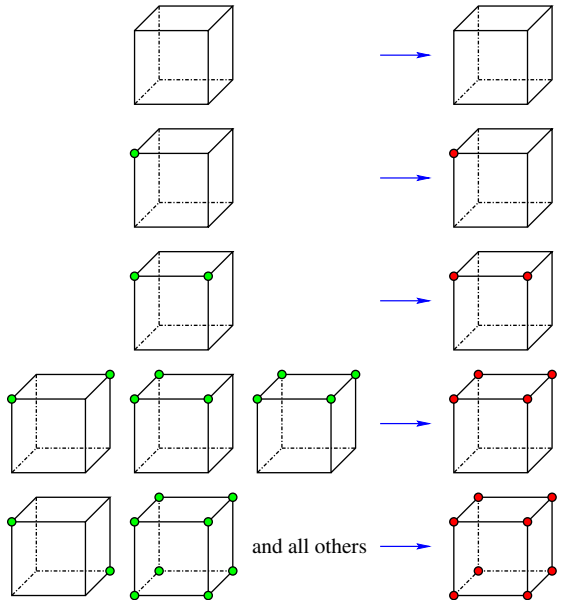


Fig. 12. The Look-Up table for converting an arbitrary configuration to one of the five templates in Fig. 10. Each green node represents the cell from which the minimizer point is generated needs to be refined. The sign of the cell generating a red node is 1, otherwise the sign is 0.

Each hexahedron is constructed by eight minimizer points, which are calculated from leaf cells in the uniform octree level. The error function of the cell generating a minimizer point is either greater than the threshold $\varepsilon$ or $\leqslant\varepsilon$, therefore there are a total of $2^8 = 256$ configurations for a hexahedron. Fig. 12 shows the Look-Up table for converting an arbitrary configuration to the five templates shown in Fig. 10. The green node means the error function of the cell generating this minimizer point is greater than the threshold $\varepsilon$. The red node means the sign of the cell generating this node is set to be 1, otherwise the sign is 0.

In the process of adaptive hexahedral mesh generation, we need to insert extra vertices and detect if they lie on the boundary or not. If a vertex lies on a boundary edge or a boundary face, then it is a boundary vertex. Otherwise it lies interior to the interval volume. There is a special case, of which we need to be careful. It occurs in cases where a vertex lying on an edge whose two end points are on the boundary, or lying on a face whose four points are on the boundary, may not be on the boundary. For those extra vertices lying inside the interval volume, we choose the linear interpolation of the eight vertices of the original hexahedron. For those existing and newly inserted vertices lying on the boundary isosurface, we first compute their positions from the linear interpolation, then move them toward the isosurface as we do for adaptive quadrilateral isosurface extraction.

Fig. 13 compares adaptive hexahedral meshes of the human head generated from Method 1 and Method 2. It is obvious that Method 2 constructs a better nose than Method 1 because it introduces extra vertices on edges of refined hexes resulting in a more accurate approximation, and Method 2 tends to generate meshes with better quality than Method 1. The extracted surface mesh from Method 2 is a little different from the result of the third method shown in Fig. 7, since only templates 0, 1, 2*a* and 4 of the third method in Fig. 4 are adopted, while templates 2*b* and 3 are not used. Since we still use QEF (Eq. (1)) for computing minimizing vertices, we can also preserve sharp edges and corners (Fig. 14).

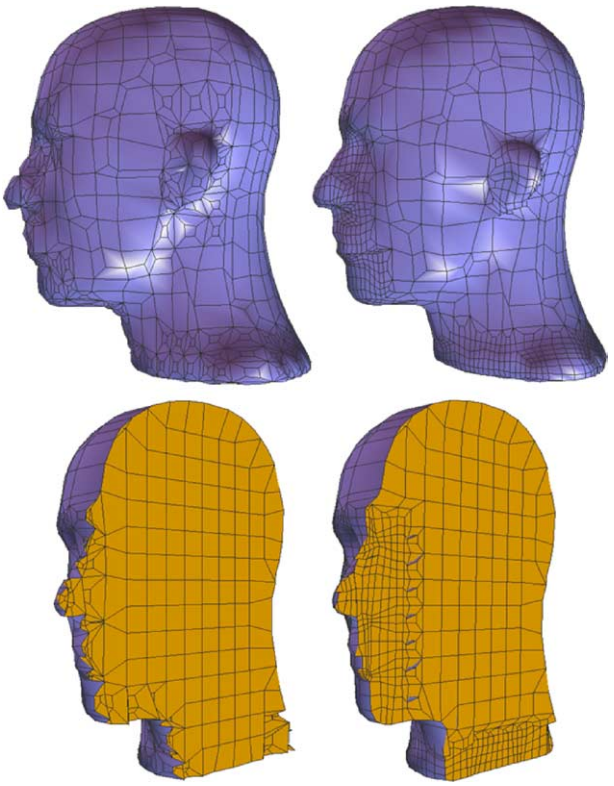

Fig. 13. Adaptive hexahedral meshes from Method 1 (left) and Method 2 (right) for the human head. Top row shows the boundary isosurfaces, it is obvious that Method 1 generates a poorly-shaped nose as was shown in Fig. 7. Bottom row shows cross sections, the right part of elements are removed.

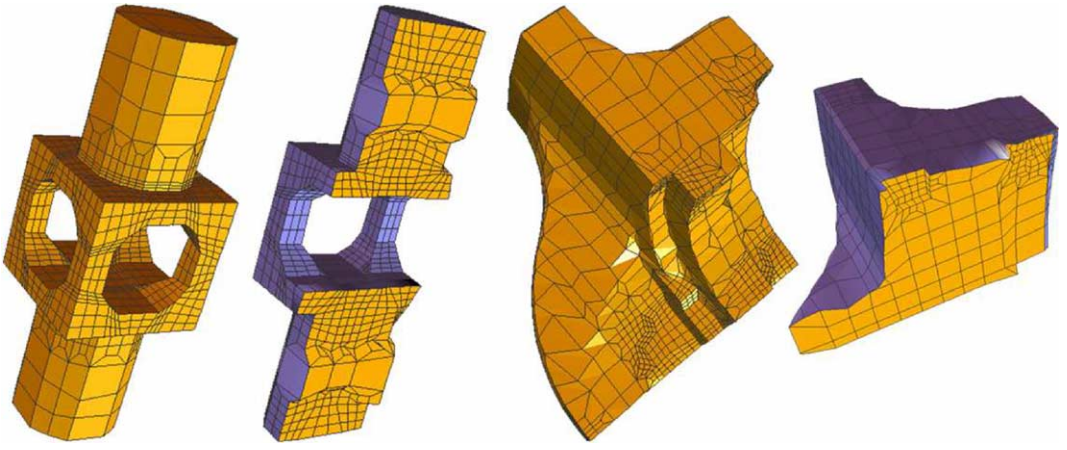

Fig. 14. Sharp features are preserved. From left to right: an adaptive quad mesh of a mechanical part, an adaptive hex mesh of a mechanical part, an adaptive quad mesh of a fandisk, and an adaptive hex mesh of a fandisk.

## 6. Mesh adaptivity

In order to generate accurate meshes with the minimal number of elements and vertices, it is important to choose a good error metric to decide where we should generate a finer mesh and where a coarser mesh should be kept. There are three main ways to control the mesh adaptivity. Users can also design an error function based on their specific requirements.

- Feature sensitive error function.
- Areas that users are interested in.
- Finite element calculation results.

The feature sensitive error function [38,39] is defined as the difference of trilinear interpolation functions between coarse and fine octree levels normalized by the gradient magnitude. It is sensitive to areas of large geometric features since it directly measures the surface difference between coarse and fine levels, for example, the facial features (nose, eyes, mouth and ears) in the head model as shown in Figs. 13 and 18.

Sometimes, people are interested in some special areas based on their physical or biological applications. For example, there is a cavity in the structure of the biomolecule called mouse acetylcholinesterase (mAChE) [31,30]. A finer mesh is required around the cavity area while a coarse mesh needs to be kept in other regions. In this situation, the error function should be defined by regions. Fig. 1 shows the adaptive quadrilateral and hexahedral meshes for the biomolecule mAChE, and it is obvious that the mesh adaptivity is controlled by regions.

In finite element simulations, we first need to construct meshes to represent the analyzed geometric domain, then solve ordinary/partial differential equations over it using the finite element method. For accurate and efficient finite element analysis, adaptive meshes are preferable. The mesh adaptivity can be controlled directly by finite element solutions to balance the error of finite element solutions over each element. Fig. 21 shows quad meshes of a bubble model. The mesh adaptivity is controlled by its deformation obtained from the finite element analysis.

## 7. Quality improvement

Quality improvement is a necessary step for finite element mesh generation. First we need to choose corresponding quality metrics to measure the quality of quadrilateral and hexahedral meshes. Here we select the scaled Jacobian, the condition number of the Jacobian matrix and Oddy metric [22] as our metrics [17–19].

Assume $x \in \Re^3$ is the position vector of a vertex in a quad or a hex, and $x_i \in \Re^3$ for $i = 1, \ldots, m$ are its neighboring vertices, where $m = 2$ for a quad and $m = 3$ for a hex. Edge vectors are defined as $e_i = x_i - x$ with $i = 1, \ldots, m$, and the Jacobian matrix is $J = [e_1, \ldots, e_m]$. The determinant of the Jacobian matrix is called *Jacobian*, or *scaled Jacobian* if edge vectors are normalized. An element is said to be *inverted* if one of its Jacobians $\leqslant 0$. We use the *Frobenius norm* as a matrix norm, $|J| = (\mathrm{tr}(J^{\mathrm{T}}J)^{1/2})$. The condition number of the Jacobian matrix is defined as $\kappa(J) = |J||J^{-1}|$, where $|J^{-1}| = \frac{|J|}{\det(J)}$. Therefore, the three quality metrics for a vertex $x$ in a quad or a hex are defined as follows:

$$\mathrm{Jacobian}(x) = \det(J), \tag{2}$$

$$\kappa(x) = \frac{1}{m}|J^{-1}||J|, \tag{3}$$

$$\mathrm{Oddy}(x) = \frac{\left(|J^{\mathrm{T}}J|^2 - \frac{1}{m}|J|^4\right)}{\det(J)^{\frac{4}{m}}}, \tag{4}$$

where $m = 2$ for quadrilateral meshes and $m = 3$ for hexahedral meshes.

In the process of mesh quality improvement, our goal is to remove inverted elements and improve the worst condition number of the Jacobian matrix. First the averaging method is used to remove inverted elements. We calculate the scaled Jacobian for a vertex in each element, and relocate this vertex by the average of all its neighbors if the Jacobian is negative. Then we calculate the condition number of the Jacobian matrix for a vertex in each quad or hex, and find the vertex with the maximum value. We compute the new position for this vertex using the conjugated gradient method with the condition number (Eq. (3)) as objective.

If the relocated vertex is an interior node, then we replace the location of this vertex with the calculated new position. If this vertex lies on the boundary, then we calculate its new position and move it toward the isosurface along its normal direction. We keep reducing the maximum condition number for quad or hex meshes until we arrive a given threshold. In this way, we can improve the worst condition number of the

| Type | DataSet | MeshSize (Vertex#, Elem#) | Scaled Jacobian (best, aver., worst) | Condition Number (best, aver., worst) | Oddy Metric (best, aver., worst) | Inverted Elem# |
|---|---|---|---|---|---|---|
| quad | Bubble[1] | (208, 206) | (1.0, 0.92, 0.36) | (1.0, 1.12, 2.77) | (0.0, 0.61, 13.37) | 0 |
| | Bubble[2] | - | (1.0, 0.94, 0.62) | (1.0, 1.07, 1.60) | (0.0, 0.34, 3.13) | 0 |
| | Head[1] | (714, 712) | (1.0, 0.92, 0.06) | (1.0, 1.13, 17.41) | (0.0, 0.98, 604.24) | 0 |
| | Head[2] | - | (1.0, 0.92, 0.37) | (1.0, 1.10, 2.73) | (0.0, 0.48, 12.93) | 0 |
| | mAChE[1] | (19998, 20013) | (1.0, 0.90, 0.04) | (1.0, 1.17, 27.63) | (0.0, 1.29, 1524.67) | 0 |
| | mAChE[2] | - | (1.0, 0.90, 0.16) | (1.0, 1.15, 6.26) | (0.0, 0.87, 76.28) | 0 |
| hex | Head[1] | (1210, 812) | (1.0, 0.85, 1.9e-3) | (1.0, 2.62, 519.74) | (0.0, 12.88, 6.95e-3) | 1 |
| | Head[2] | - | (1.0, 0.85, 0.02) | (1.0, 1.98, 46.34) | (0.0, 5.03, 638.83) | 0 |
| | mAChE[1] | (81233, 70966) | (1.0, 0.94, 5.2e-5) | (1.0, 2.07, 1.92e-4) | (0.0, 18.35, 1.58e-6) | 5 |
| | mAChE[2] | - | (1.0, 0.94, 0.01) | (1.0, 1.40, 74.73) | (0.0, 2.37, 1379.81) | 0 |

Fig. 15. The comparison of the three quality criteria (the scaled Jacobian, the condition number and Oddy metric) before/after the quality improvement for quadrilateral meshes of bubble, head and mAChE. DATA[1]—before quality improvement; DATA[2]—after quality improvement.

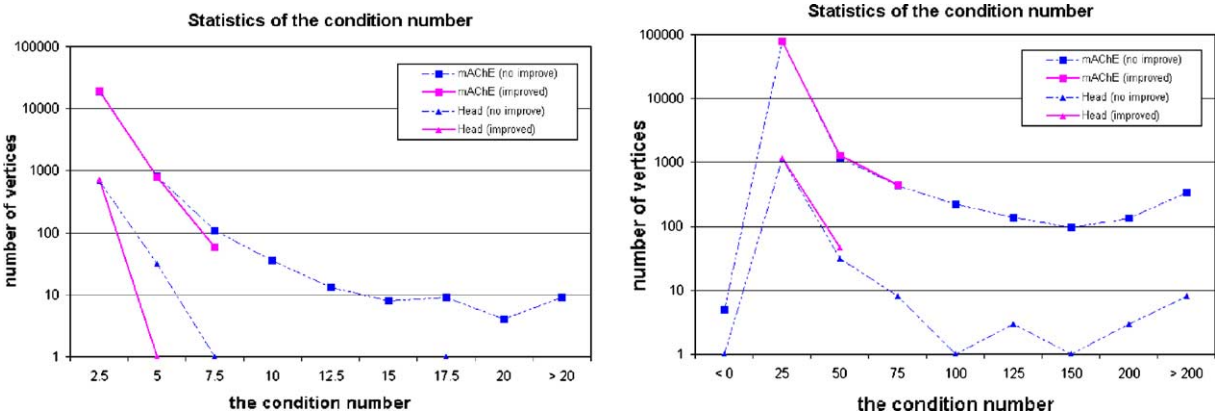


Fig. 16. The histogram of the condition number for quadrilateral (left) and hexahedral (right) meshes of mAChE and the human head.

Jacobian matrix, as well as improving the other two metrics, the scaled Jacobian and Oddy metric. However, it is possible to produce an invalid mesh containing inverted elements. We choose a 'smart' smoothing method [12], which relocates the point only if the mesh quality is improved.

Fig. 15 shows the improvement of the worst values of the scaled Jacobian, the condition number and Oddy metric. The histograms of the condition number (Fig. 16) show the overall quality of quad and hex meshes for the human head model and a biomolecule mAChE. By Comparing the three quality metrics before and after quality improvement, we can see that the worst parameters are improved significantly.

## 8. Results and applications

We have developed an interactive program for adaptive and quality quadrilateral/hexahedral mesh extraction and rendering from volumetric data, and plugged it into our LBIE-Mesh software (Level Set Boundary and Interior–Exterior Mesher), which can generate adaptive and quality 2D (triangular/quadrilateral) and 3D (tetrahedral/hexahedral) meshes from volume data. The algorithm of tetrahedral mesh gen-

| DataSet | Type | Dimension | Number of Elements (Extraction Time (unit: ms)) | | | |
|---|---|---|---|---|---|---|
| | | | (a) | (b) | (c) | (d) |
| Bubble | SDF | $65^3$ | 206 (172) | 1478 (329) | 1854 (344) | – |
| Head | SDF | $65^3$ | 1942 (594) | 812 (375) | 4049 (750) | 17905 (3267) |
| Knee | SDF | $65^3$ | 4058 (735) | 1386 (453) | 7111 (797) | 36207 (1516) |
| Skull | CT | $129^3$ | – | – | 20416 (9893) | 10827 (9205) |
| Skin | CT | $129^3$ | 20999 (9955) | 61244 (14565) | – | – |
| mAChE | Given | $257^3$ | 20013 (6080) | – | 70966 (11690) | 38939 (7955) |

Fig. 17. Data sets and test results. The CT data sets are re-sampled to fit into the octree representation. Rendering results for each case are shown in Figs. 21, 18–20 and 1. Skull and skin are extracted from the UNC Head model.

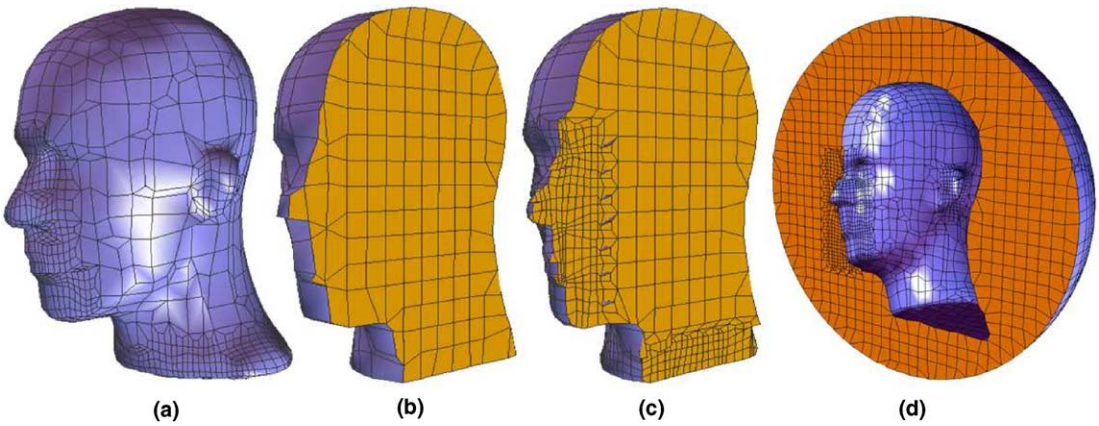


Fig. 18. Quadrilateral and hexahedral meshes of the human head: (a) an adaptive quadrilateral mesh; (b) the uniform hexahedral mesh at a chosen starting level; (c) an adaptive interior hexahedral mesh controlled by the feature sensitive error function; (d) an adaptive exterior hexahedral mesh controlled by the feature sensitive error function.

eration is described in [38,39]. In this software, 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 quadrilateral and hexahedral meshes for some signed distance function data such as the bubble (Fig. 21), the human head (Fig. 18) and the knee model (Fig. 19). We have also extracted meshes for the skin and the skull from a CT scanned data (the UNChead, Fig. 20), and tested the algorithm on biomolecular data (mAChE, Fig. 1). Fig. 17 shows the information for each dataset and results. The results consist of the number of elements, the extraction time and images with respect to different isovalues and error tolerances. Extraction time includes octree traversal, QEF computation and mesh extraction.

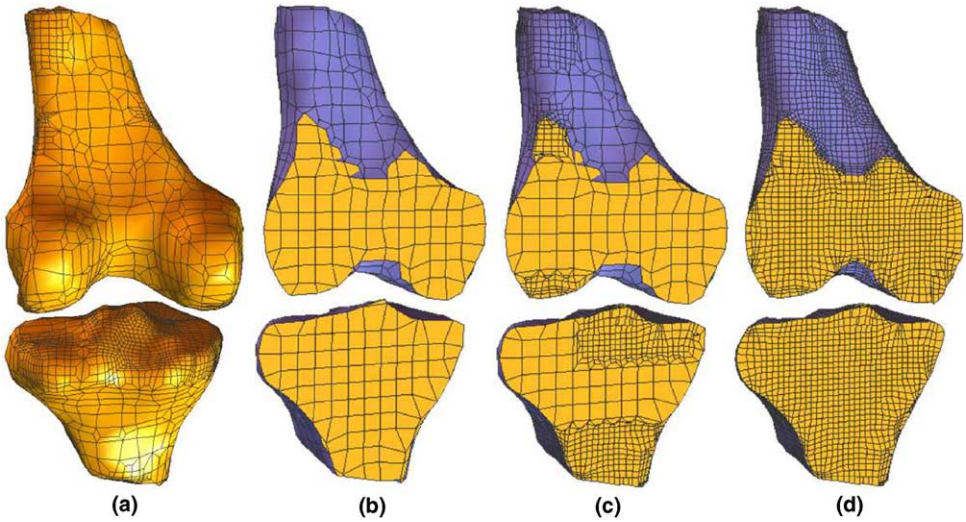


Fig. 19. Quadrilateral and hexahedral meshes of the knee: (a) an adaptive quadrilateral mesh; (b) the uniform hexahedral mesh at a chosen starting level; (c) an adaptive hex mesh controlled by the feature sensitive error function; (d) all the hexahedral elements in (b) are refined.

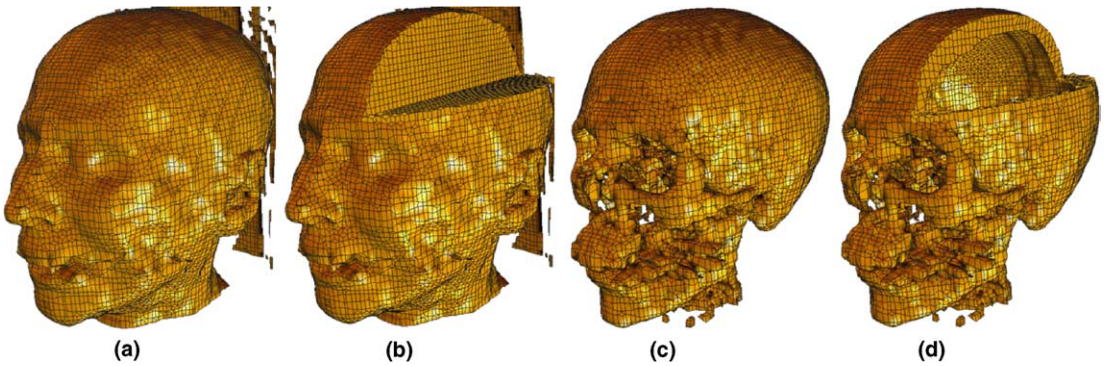


Fig. 20. Quadrilateral and hexahedral meshes are extracted from a CT-scanned volumetric data (UNC head): (a) the quadrilateral mesh of the skin; (b) the hexahedral mesh of the volume inside the skin; (c) the quadrilateral mesh of the skull isosurface; (d) the hexahedral mesh of the skull.

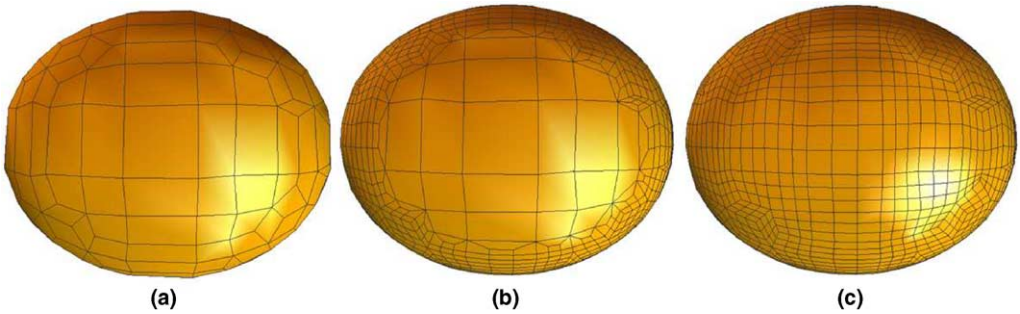


Fig. 21. Quadrilateral meshes of a bubble model: (a) the uniform mesh at a chosen starting level; (b) an adaptive mesh controlled by finite element solutions (deformation); (c) a mesh generated by refining all the elements in (a).

Fig. 21 shows the extracted quadrilateral meshes for a bubble, which has been used in the simulation of drop deformation using the finite element method. First, we generate a uniform quad mesh for the original state of the bubble. Then we get finite element solutions such as the deformation from finite element analysis, and use the error of the deformation over each element to control the mesh adaptivity. Finally we can provide an adaptive and quality quad mesh to limit the maximum error of finite element solutions within a threshold.

Some physically-based simulations need both interior and exterior hexahedral meshes. For example, when people are analyzing the electromagnetic scattering over the human head, hex meshes of the volume interior to the head surface and hex meshes exterior to the head surface but inside an outer sphere are needed at the same time. Fig. 18 shows the extracted interior and exterior meshes for a head model. The facial features such as nose, eyes, mouth and ears are kept, and fine meshes are generated in those regions. Fig. 1 shows another example of interior and exterior hexahedral meshes, the biomolecule mAChE. The mesh adaptivity is controlled by regions, fine meshes are generated in the area of cavity.

## 9. Conclusions

We have presented an algorithm to extract adaptive and quality quadrilateral and hexahedral meshes directly from volumetric data. First, a bottom-up surface topology preserving octree-based algorithm is used to select a starting octree level, at which we extract uniform meshes with correct topology using the dual contouring isosurface extraction method [15,38,39]. Then we extended it to adaptive quadrilateral and hexahedral mesh generation using some pre-defined templates without introducing any hanging nodes. The position of each boundary vertex is recalculated to approximate the isosurface more accurately. The mesh adaptivity can be controlled in three ways, the feature sensitive error function [38,39], the areas that users are interested in and finite element solutions. Users can also design their own error function to control the mesh adaptivity according to their specific requirements. Finally, three various quality metrics are selected to measure the mesh quality, and the relaxation based technique is used to improve it. The resulting meshes are extensively used for efficient and accurate finite element calculations. Some of them have been used successfully.

## Acknowledgements

An early version of this paper appeared in 13th International Meshing Roundtable conference [37]. We thank Bong-Soo Sohn for several useful discussions, Jianguang Sun for our system management,

Dr. Gregory Rodin for finite element solutions of drop deformation, Dr. Nathan Baker for providing access to the accessibility volume of biomolecule mAChE and UNC for the CT volume dataset of a human head respectively. This work was supported in part by NSF grants ACI-0220037, EIA-0325550, and NIH grant P20 RR02647-01.

## References

[1] Cubit mesh generation toolkit. Available from: <http://www.sass1693.sandia.gov/cubit>.
[2] P. Baehmann, S. Wittchen, M. Shephard, K. Grice, M. Yerry, Robust geometrically based, automatic two-dimensional mesh generation, Int. J. Numer. Methods Engrg. 24 (1987) 1043–1078.
[3] C. Bajaj, J. Warren, G. Xu, A subdivision scheme for hexahedral meshes, Visual Comput. 18 (5–6) (2002) 343–356.
[4] T. Blacker, R. Myers, Seams and wedges in plastering: a 3d hexahedral mesh generation algorithm, Engrg. Comput. 2 (1993) 83–93.
[5] T. Blacker, M. Stephenson, Paving: a new approach to automated quadrilateral mesh generation, Int. J. Numer. Methods Engrg. 32 (1991) 811–847.
[6] S. Canann, Plastering and optismoothing: new approaches to automated, 3d hexahedral mesh generation and mesh smoothing. Ph.D. Dissertation, Brigham Young University, Provo, UT, 1991.
[7] S. Canann, J. Tristano, M. Staten, An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral and quad-dominant meshes, in: 7th International Meshing Roundtable, 1998, pp. 479–494.
[8] C. Charalambous, A. Conn, An efficient method to solve the minimax problem directly, SIAM J. Numer. Anal. 15 (1) (1978) 162–187.
[9] W.A. Cook, W.R. Oakes, Mapping methods for generating three-dimensional meshes, Comput. Mech. Engrg. (1982) 67–72.
[10] David Eppstein, Linear complexity hexahedral mesh generation, in: Symposium on Computational Geometry, 1996, pp. 58–67.
[11] D. Field, Laplacian smoothing and Delaunay triangulations, Commun. Appl. Numer. Methods 4 (1988) 709–712.
[12] L. Freitag, On combining Laplacian and optimization-based mesh smoothing techniques, Trends in Unstructured Mesh Generation, AMD-Vol. 220, 1997, pp. 37–43.
[13] L. Freitag, C. Ollivier-Gooch, Tetrahedral mesh improvement using swapping and smoothing, Int. J. Numer. Methods Engrg. 40 (1997) 3979–4002.
[14] M. Garland, P. Heckbert, Simplifying surfaces with color and texture using quadric error metrics, in: IEEE Visualization, 1998, pp. 263–270.
[15] T. Ju, F. Losasso, S. Schaefer, J. Warren, Dual contouring of Hermite data, in: Proceedings of SIGGRAPH, 2002, pp. 339–346.
[16] P. Kinney, Cleanup: improving quadrilateral finite element meshes, in: 6th International Meshing Roundtable, 1997, pp. 437–447.
[17] P. Knupp, Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part I: A framework for surface mesh optimization, Int. J. Numer. Methods Engrg. 48 (2000) 401–420.
[18] P. Knupp, Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities. Part II: A framework for volume mesh optimization and the condition number of the Jacobian matrix, Int. J. Numer. Methods Engrg. 48 (2000) 1165–1185.
[19] C. Kober, M. Matthias, Hexahedral mesh generation for the simulation of the human mandible, in: 9th International Meshing Roundtable, 2000, pp. 423–434.
[20] C. Lee, S. Lo, A new scheme for the generation of a graded quadrilateral mesh, Comput. Struct. 52 (5) (1994) 847–857.
[21] S. Mitchell, T. Tautges, Pillowing doublets: refining a mesh to ensure that faces share at most one edge, in: 4th International Meshing Roundtable, 1995, pp. 231–240.
[22] A. Oddy, J. Goldak, M. McDill, M. Bibby, A distortion metric for isoparametric finite elements, Transactions of CSME, No. 38-CSME-32, Accession No. 2161, 1988.
[23] S. Owen, A survey of unstructured mesh generation technology, in: 7th International Meshing Roundtable, 1998, pp. 26–28.
[24] M. Price, C. Armstrong, Hexahedral mesh generation by medial surface subdivision: Part I, Int. J. Numer. Methods Engrg. 38 (19) (1995) 3335–3359.
[25] M. Price, C. Armstrong, Hexahedral mesh generation by medial surface subdivision: Part II, Int. J. Numer. Methods Engrg. 40 (1997) 111–136.
[26] R. Schneiders, A grid-based algorithm for the generation of hexahedral element meshes, Engrg. Comput. 12 (1996) 168–177.
[27] R. Schneiders, Refining quadrilateral and hexahedral element meshes, in: 5th International Conference on Grid Generation in Computational Field Simulations, 1996, pp. 679–688.
[28] R. Schneiders, An algorithm for the generation of hexahedral element meshes based on an octree technique, in: 6th International Meshing Roundtable, 1997, pp. 195–196.

[29] R. Schneiders, R. Schindler, F. Weiler, Octree-based generation of hexahedral element meshes, in: 5th International Meshing Roundtable, 1996, pp. 205–216.
[30] Y. Song, Y. Zhang, C.L. Bajaj, N.A. Baker, Continuum diffusion reaction rate calculations of wild type and mutant mouse acetylcholinesterase: adaptive finite element analysis, Biophys. J. 87 (3) (2004) 1558–1566.
[31] Y. Song, Y. Zhang, T. Shen, C.L. Bajaj, J.A. McCammon, N.A. Baker, Finite element solution of the steady-state Smoluchowski equation for rate constant calculations, Biophys. J. 86 (4) (2004) 2017–2029.
[32] M. Staten, S. Canann, Post refinement element shape improvement for quadrilateral meshes, Trends in Unstructured Mesh Generation, AMD-Vol. 220, 1997, pp. 9–16.
[33] J. Talbert, A. Parkinson, Development of an automatic, two dimensional finite element mesh generator using quadrilateral elements and Bezier curve boundary definitions, Int. J. Numer. Methods Engrg. 29 (1991) 1551–1567.
[34] T. Tautges, T. Blacker, S. Mitchell, The whisker-weaving algorithm: a connectivity based method for constructing all-hexahedral finite element meshes, Int. J. Numer. Methods Engrg. 39 (1996) 3327–3349.
[35] S.-H. Teng, C.W. Wong, Unstructured mesh generation: theory, practice, perspectives, Int. J. Comput. Geometry Appl. 10 (3) (2000) 227–266.
[36] F. Weiler, R. Schindler, R. schneiders, Automatic geometry-adaptive generation of quadrilateral and hexahedral element meshes for the FEM, Numerical Grid Generation in Computational Field Simulations, 1996, pp. 689–697.
[37] Y. Zhang, C. Bajaj, Adaptive and quality quadrilateral/hexahedral meshing from volumetric data, in: 13th International Meshing Roundtable, 2004, pp. 365–376.
[38] Y. Zhang, C. Bajaj, B.-S. Sohn, Adaptive and quality 3d meshing from imaging data, in: ACM Symposium on Solid Modeling and Applications, 2003, pp. 286–291.
[39] Y. Zhang, C. Bajaj, B.-S. Sohn, 3D finite element meshing from imaging data, Comput. Methods Appl. Mech. Engrg., in press, doi:10.1016/j.cma.2004.11.026.
[40] J. Zhu, O. Zienkiewicz, E. Hinton, J. Wu, A new approach to the development of automatic quadrilateral mesh generation, Int. J. Numer. Methods Engrg. 32 (1991) 849–866.