Direct extraction of surface meshes from implicitly represented heterogeneous volumes

Charlie C. L. Wang

Department of Automation and Computer-Aided Engineering, Chinese University of Hong Kong,

Shatin, N.T., Hong Kong

E-mail: cwang@acae.cuhk.edu.hk

Abstract

This paper describes a novel algorithm to extract surface meshes directly from implicitly represented heterogeneous models made of different constituent materials. Our approach can directly convert implicitly represented heterogeneous objects into a surface model separating homogeneous material regions, where every homogeneous region in a heterogeneous structure is enclosed by a set of two-manifold surface meshes. Unlike other discretization techniques of implicitly represented heterogeneous objects, the intermediate surfaces between two constituent materials can be directly extracted by our algorithm. Therefore, it is more convenient to adopt the surface meshes from our approach in the *boundary element method* (BEM) or as a starting model to generate volumetric meshes preserving intermediate surfaces for the *finite element method* (FEM). The algorithm consists of three major steps: firstly, a set of assembled two-manifold surface patches coarsely approximating the interfaces between homogeneous regions are extracted and segmented; secondly, signed distance-fields are constructed that each field expresses the Euclidean distance from points to the surface of one homogeneous material region; and finally, coarse patches generated in the first step is dynamically optimized to give adaptive and high-quality surface meshes. The manifold topology is preserved on each surface patch.

Keywords: surface mesh, manifold preserved, implicit representation, heterogeneous models, remeshing.

1. Introduction

The modeling methods of heterogeneous objects have been widely studied in the past decade. A heterogeneous object is actually a solid model made of different materials, where each material occupies portion of the solid. Recently, more and more interests are shown to model heterogeneous structure using implicitly represented volumes [1-5]. The implicit models provide compact and intuitive mathematical representation for complex heterogeneous objects, supporting the operations from set theory and other operations such as offsetting, blending, and sweeping. Finite element method (FEM) and boundary element method (BEM) nowadays take the role as the most powerful analysis tools in engineering. However, the implicit represented models cannot be directly applied in them since these numerical methods usually need discrete models (volume or surface meshes) of geometric objects. Although the meshfree analysis and simulation methods [6, 7] can be employed, it is still important to have the surface representation in many computational engineering applications. Here comes the purpose of our research: *to develop a method for extracting surface meshes which discretely represent the implicit heterogeneous volumes*. After getting well-defined surface meshes, the procedure to generate a volumetric mesh by them is standard (see [8, 9] and the references therein).

Benefited from the compact and intuitive mathematical representation, a lot of approaches conducted the implicit representation to optimize shape and topology of a given structure (e.g., [2, 10]). The approaches in [10]

considered about single material; therefore, the resulting surface can be easily determined by an isosurface extraction method (e.g., the optimized Marching Cubes method [11]). The technique introduced in [2] extends the topology and structure optimization to the domain of multi-material composition, so the intermediate surfaces are required to be generated. For an intermediate surface between the regions filled with two materials u_1 and u_2, if the isosurfaces are individually extracted on the region covered by u_1 and the portion occupied by u_2, the consistency between them cannot be guaranteed. In order to achieve a model with high accuracy it is expected that only one intermediate surface between u_1 and u_2 is constructed, hereby no gap or overlap is generated between two regions. Besides the structure optimization, similar problem arises when extracting a boundary surface of a human body part from CT or MRI data (ref. [12]). In summary, the problem addressed in this paper has many applications in both the mechanical engineering and the biomedical engineering.

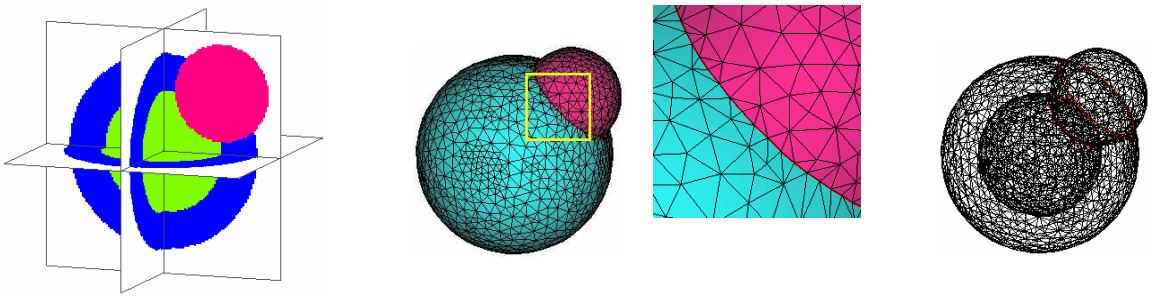
Problem statement: Following the representation method in [2], without loss of general, an implicit heterogeneous object H is a solid in a given domain $\Psi \subseteq \Re^3$ defined by a function $F(p)$ with $p \in \Psi$. If there are total n materials involved in H, the value of $F(p)$ is an integral index of material class between 0 and n (*zero* represents no material), which indicates the material type in H at p. Suppose that a region filled with the material class i is denoted by Ω_i, we have

$$H = \bigcup_{i=1}^{n} \Omega_i . \tag{1}$$

The heterogeneous object H is partitioned by sub-regions with unique material. Thus, we have $\Omega_i \bigcap_{i \neq j} \Omega_j = \Gamma_{ij}$ and $\Gamma_i = \partial\Omega_i \cap H$, where each material region has a meaningful boundary surface Γ_i and the interface between two material regions Ω_i and Ω_j is denoted by $\Gamma_{ij} = \Gamma_{ji} = \Gamma_i \cap \Gamma_j$ ($\forall i \neq j$). When considering the region has no material filled as material class *zero*, all the intermediate surfaces in H can be defined as

$$\Gamma = \bigcup_{i,j=0}^{n} (1 - \delta_{i,j})\Gamma_{ij} , \tag{2}$$

where $\delta_{i,j}$ is the Kronecker delta defined to be 1 if $i = j$ and 0 otherwise. Therefore, the problem to be solved in this paper is to construct an adaptive and quality mesh approximating Γ of H, where each intermediate surface Γ_{ij} is represented by a two-manifold mesh patch and two linked patches should have consistent boundaries (i.e., the corresponding nodes are coincident). For instance, in Fig.1, an implicit heterogeneous object is originally given in Fig.1a – different color represents different materials; Fig.1b and 1c show the surface meshes extracted by our approach. It is easy to find that the meshes are compatible at boundaries.



(a) implicit representation **(b) extracted surface** **(c) wireframe display**

Fig. 1 Example I: a heterogeneous object consists of three materials

Major contribution: To convert an implicit represented heterogeneous volume into a set of two-manifold mesh surfaces that can be utilized in computational engineering applications, a novel algorithm will be presented in this paper to extract adaptive and high-quality surface meshes. Two-manifold is guaranteed on each patch of extracted surfaces. The consistency is preserved on the interface of adjacent material regions, which cannot be solved by directly applying isosurface extraction algorithms. Although the volumetric-element related approaches could also generate elements approximating intermediate surfaces if they are adaptive to them, but our approach directly generates surface meshes from implicit data – this greatly reduces the computer memory required. Our algorithm also guarantees the connectivity consistency on the boundaries of surface patches, which is important to many computational engineering applications. The resultant surface meshes are adaptive to curvature and provide good element shape. By choosing different cell size and different refinement accuracy, the mesh patches with different level-of-details can be easily determined.

The rest parts of the paper are organized as follows. In the next section, related researches in literature will be first reviewed. The outline of our algorithm is then presented in section 3 followed by the detail algorithms in succession. After that, experimental results are shown together with some applications of our approach. Finally, our paper ends with conclusions and discussions.

2. Review of related research

The algorithm presented here closely relates to isosurface extraction technology, but we are facing a problem which cannot be well solved by existed isosurface polygonization approaches in literature. The discretization techniques for implicit datasets fall into three categories: 1) marching cubes and its variants, 2) sharp feature preserved polygonization, and 3) isosurface approximation methods, which are reviewed below.

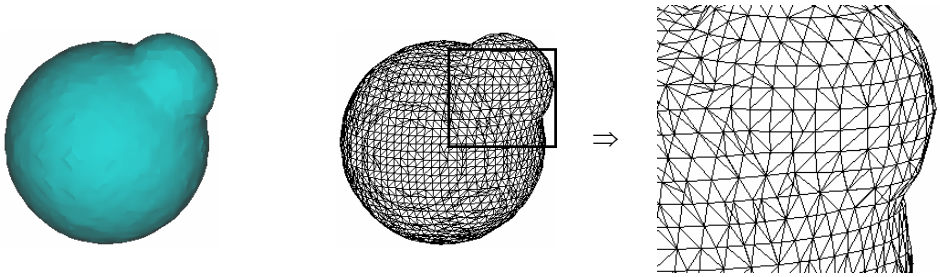


Fig. 2 Outline of the heterogeneous object in example I contoured using Marching Cubes [25]

Marching cubes and its variants

Marching cubes (MC) algorithm was first introduced by Lorensen and Cline [13] and has become the most commonly used method for isosurface extraction in scientific visualization. As first noted by Duerst [14], the original MC algorithm [13] may produce isosurfaces with holes due to topologically inconsistent decisions on the reconstruction of ambiguous faces, where the borders used by one incident cube do not match the borders of the other incident cube. Several approaches addressing this problem have been published (see [11, 15] for a review). As addressed by [11], disambiguation techniques reported so far have focused on two major concerns: topological consistency [16-21], i.e. producing closed surfaces by proper cube polygonization, and topological correctness [21-25], i.e. extracting a surface faithful to the geometry of the real surface.

A few works attempt to recover the original topology also inside the ambiguous cubes either by using critical point analysis [26, 27] or trilinear interpolation [28, 29]. All these techniques are data-dependent and therefore are noise-sensitive and cannot be applied to binary grids. In [11], the authors propose global strategies

for optimizing several topological and combinatorial measures of the isosurfaces including triangle count, genus, and number of shells. However, the decisions of the measurements to be given by users are not natural, i.e., novices may feel difficult to give good decision. Our mesh extraction algorithm follows the common requirement of a good isosurface extractor – *manifold guaranteed*, and the requested inputs from users are more intuitive – so easier to control. Also, in most MC algorithms, the element shape of a resulting mesh is not controlled. As shown in Fig.2, the mesh resulted from [25] has a lot of thin triangles and short edges, while our result in Fig.1 gives good element shapes. Although post-processing steps, such as triangle decimation techniques [30-32] as well as re-tiling algorithm [33] can be adopted to eliminate badly shaped triangles, the approximation accuracy can be better controlled if we directly address the aspect of generated triangles during the contouring process. In fact, the adaptive remeshing step of our algorithm borrows some ideas from the triangle decimation approaches. Another problem is about the common curves between the bounding surfaces of different material regions, which are not generated by all MC algorithms but are preserved in our approach (see Fig.1b).

A closely related work to our approach in literature is [12] for the application in biomedical engineering. The authors of [12] modified the MC for single material to M3C – a multi-material marching cubes algorithm (similar ideas have also been shown in [34, 35]). They solved the geometry ambiguous problem by assuming that continuity will be preserved when the same material IDs are shown on diagonals. However, as mentioned in [12], there are 8^8 (i.e., 16 777 216) possible cases when the eight cube vertices have eight different materials – thus, the programming task for topology correctness will become very tough. Our approach develops a compact cell-merging strategy to elegantly solve the problem of topology correctness. Besides, the post-smoothing step in [12] leads to the shrinkage of surface. The same problem will not occur in our approach since the remeshing is governed by the underlying distance-fields, which prevents the shrinking effect.

Sharp feature preserved polygonization

The accuracy of a marching cube algorithm is mainly governed by the resolution of underlying grids, so sharp features are destroyed. Some approaches addressed this problem. The techniques include generating adaptive grids based on octrees or using *adaptively sampled distance fields* (ADFs) [36]. The key challenge is to design criteria for generating adaptive subdivision. Two improved isosurface extraction algorithm, *extended marching cubes* [37] and *dual contouring* [38], have also been presented. Both algorithms use Hermite data and generate isosurfaces that contain sharp features. They work well when each cell contains no more than one sharp features or complex edges (i.e. edges with more than one intersection with a surface). The approach in [39] solved the problem with more than one sharp features by integrating the adaptive grid generation methods and the improved isosurface extraction algorithms.

The algorithms presented in [40, 41] improve the output of MC algorithms based on optimization techniques and smoothing operations. Their approach is based on mesh evolution towards a given implicit surface with simultaneous control of the mesh vertex position and mesh normals. The remeshing step in our method is akin to the dynamic mesh optimization approach [41]. The authors of [42] extended the idea in [39] to discretize functionally based heterogeneous objects. Their discretization result is with volumetric elements, and intermediate surfaces are adapted and generated during the advancing front procedure when creating volumetric elements. In other words, the surface meshes are not able to be extracted directly from heterogeneous volumes. Comparing to [42], our technique can directly extract intermediate surfaces, so less computer memory is needed;

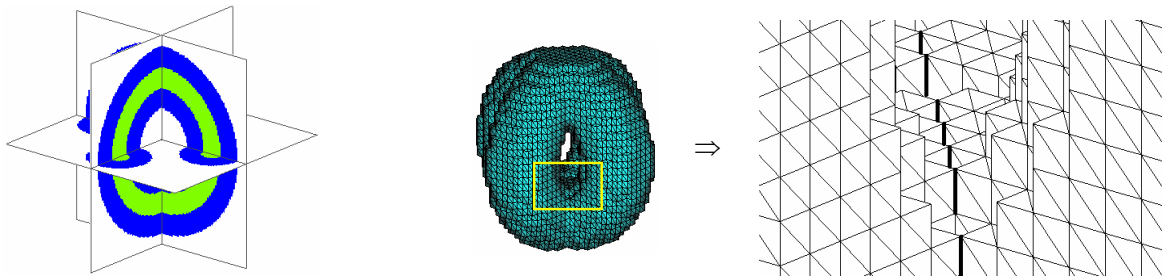
also, the tedious adaptively advancing front procedure is avoided, thus the implementation of our approach is easier and runs faster.

Isosurface approximation methods

One class of isosurface generation algorithm is based on the active model, where the constructed surface is deformed to approximate the underground isosurface embedded in a scalar-field. The methods presented in [41, 42] actually also belong to this category. Crossno and Angel [43] conducted particle systems to extract isosurfaces, where particles are programmed to attract towards a specific surface value while simultaneously repelling adjacent particles. The repulsive forces are based on the curvature of the surface at that location. Their approach presented the advantages include: vertex densities are based on surface features rather than on the sampling rate of the volume – so it is an adaptive approach; and a single scaling factor simplifies level-of-detail control. Our adaptive remeshing is also performed in the similar manner, but is speeded up by the distance-fields. A so-called *SurfaceNets* algorithm developed in [44] was an alternative to MC for building globally smooth but locally accurate triangle models from binary volume data. In [45], this algorithm is further enhanced in the *Kizamu* system to generate mesh models from distance values sampled on an adaptive grid. During our implement of their algorithm, we find that the *EdgeFace* table in [45] does not guarantee to generate a manifold mesh surface. For example, as shown in Fig.3, some voxels belonging to different toruses sharing a common edge, which leads to edges with four adjacent faces, so non-manifold topology occurs. The bolded lines in Fig.3b indicate these edges. This problem has been essentially solved in our approach.

Recently, Jin et al. [46] developed a subdivision based approach to interpolate the RBF represented implicit surfaces. During the subdivision, the newly introduced vertices are iteratively tracked to implicit surfaces. Their algorithm gives good result, however a good initial coarse mesh for subdivision is difficult to be generated automatically. The *Shrinkwrap* method presented in [47] starts with a triangulation of a sphere and next applies a series of deformations to this triangulation to transform it into the shape approximating the requested isosurface. The algorithm in [47] is adaptive in the sense that the lengths of the sides of the triangles in the mesh vary with the local curvature of the underlying surface. Unfortunately, only isosurfaces with genus number zero (i.e., topology similar to sphere) can be generated. Our method overcomes this limitation.

Marching triangles (MT) is another class of isosurface approximation approaches. It is firstly appeared in [48]. The MT algorithm employs the local 3D constraint to reconstruct a Delaunay triangulation of an arbitrary topology manifold surface. This method is further enhance in [49] by adapting the size of triangles to the curvature of surface and closing cracks at the end of mesh growing. However, the drawback inherent to all continuation methods still exists that it is difficult to determine seed triangles on the connex part of a surface.



(a) an implicitly represented heterogeneous object

(b) non-manifold edges produced (the bolded ones) by using [45] to construct the connectivity of isosurface

Fig. 3 Non-manifold happens when contouring two blended toruses by [45]

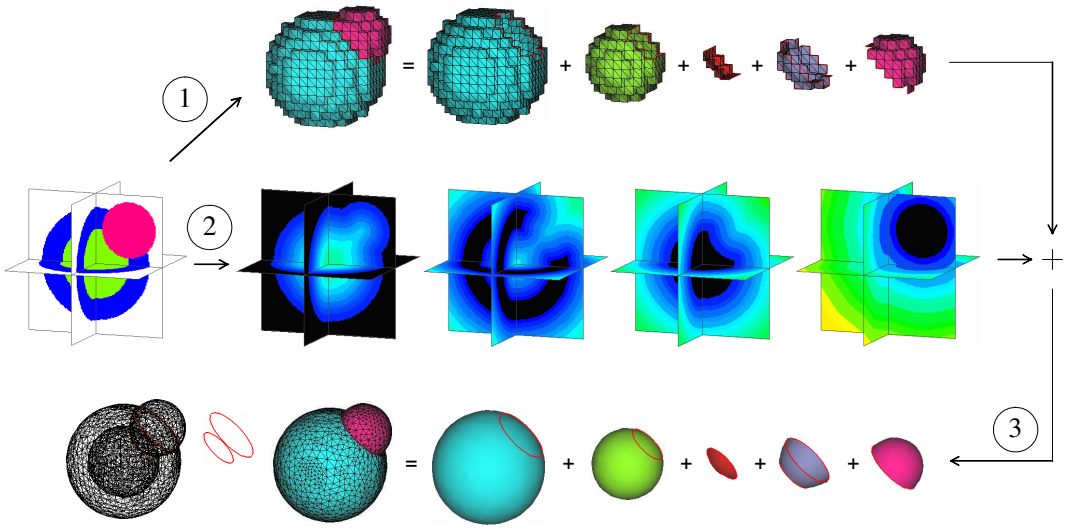


Fig. 4 Three steps of our algorithm

3. Outline of algorithm

For extracting mesh surfaces from implicitly represented heterogeneous volumes, our algorithm consists of three phases:

1. *Extraction and segmentation of two-manifold coarse patches*: The definition domain Ψ of a given heterogeneous object H is first sampled by uniformly subdivided cells with user specified cell size. A cell is identified to be filled with material class i if $F(p)=i$ is agreed at most points inside the cell (the *cell sampling* step). Then, cells belonging to the same material class are merged by joining the shared cell nodes (the *cell merging* step), and a polygon soup enclosing the different material regions is extracted on the interface of neighboring cells filled with different materials (the *polygon soup* extraction step). After that, the polygon soup is segmented into a collection of two-manifold polygonal mesh patches $\mathbf{M}=\bigcup_{i=0,j>i}^{n}\mathbf{M}_{i,j}$ where a patch $\mathbf{M}_{i,j}$ is an intermediate surface separating material region i and j (the *patch segmentation* step). Every segmented patch is a two-manifold mesh surface with linking information stored on boundary vertices. For the object of example I originally given in Fig.1, the extracted and segmented patches after this phase are shown at the top row of Fig.4. An illustration for the algorithm in this phase is given in Fig.5.
2. *Construction of signed distance-fields for different material stuff*: Signed distance-fields are constructed from each homogeneous material region (i.e., a signed distance-field $D_i(p)$ will be generated for the region Ω_i, where for any p, $D_i(p)$ denotes the signed Euclidean distance from p to $\partial\Omega_i$). These distance-fields will act as potential fields in the third phase to govern the movement of mesh vertices. For the intermediate surface Γ_{ij} between Ω_i and Ω_j, both the attraction from D_i and the one from D_j will be applied on it; however, for the most outside surface, which is with one side has no material stuff, only the attraction from one distance-field is applied on it. This easily leads to instability. Therefore, we consider the region with no material as Ω_0, a signed distance-field $D_0(p)$ is also constructed for it. As illustrated in the middle row of Fig.4, for the heterogeneous object H with three classes of materials, four signed distance-fields are computed, where the black portion represents points with negative Euclidean distances.

3. *Adaptive remeshing process*: With the help of signed distance-fields, the collection of mesh patches $\mathbf{M}$ generated in the first phase are refined and optimized to generate an adaptive and high-quality approximation of Γ. The following operations are iteratively applied on $\mathbf{M}$.

   (a) *Vertex Repositioning*: Using the distance-fields D_i and D_j to generate attractions on the vertices of7 mesh $\mathbf{M}_{i,j}$, the vertices are driven towards the isosurface of D_i and D_j (i.e., Γ_{ij}). During the repositioning, the vertices distribution inside a mesh is also relaxed so that every vertex is close to the average position of it one-ring neighborhoods.

   (b) *Adaptive Refinement*: The distance from the middle point of every edge in $\mathbf{M}$ to Γ_{ij} is detected, if it is greater than a tolerance, a new vertex will be introduced to split the edge and its related triangles.

   (c) *Element-Shape Optimization*: In this operation, extreme short edges in $\mathbf{M}$ will be eliminated by *edge collapse*, and the edges yield sharp triangles will be replaced by their dual edges using the *edge swap* operator. The edge operations leading to invalid topological structure will be avoided, and the normal flips are also prevented.

   (d) *Normal Preservation*: This is an optional operation. To preserve sharp features, the normal preservation process is first applied on every inner vertex of $\mathbf{M}$ to let the normal of polygonal faces following the normal given by Γ. Then, regions with sharp features are detected through the principal curvatures and the mean curvature on mesh vertices. In the following, the positions of vertices on sharp features are further adjusted by preserving their related polygonal normals while non-sharp-feature vertices are repositioned smoothly.

   (e) *Final Contouring*: Finally, we go through every edges of $\mathbf{M}$ to ensure the element shape of triangles and the preservation of sharp features, where the *edge swap* operator will be conducted.

   After iteratively applying the above operations on the coarse mesh patches constructed in the first phase, the final mesh patches approximating Γ are determined. For instance, the results of example I are listed at the bottom row in Fig. 4.

The implementation details of every algorithm phase will be described successively in the following sections.

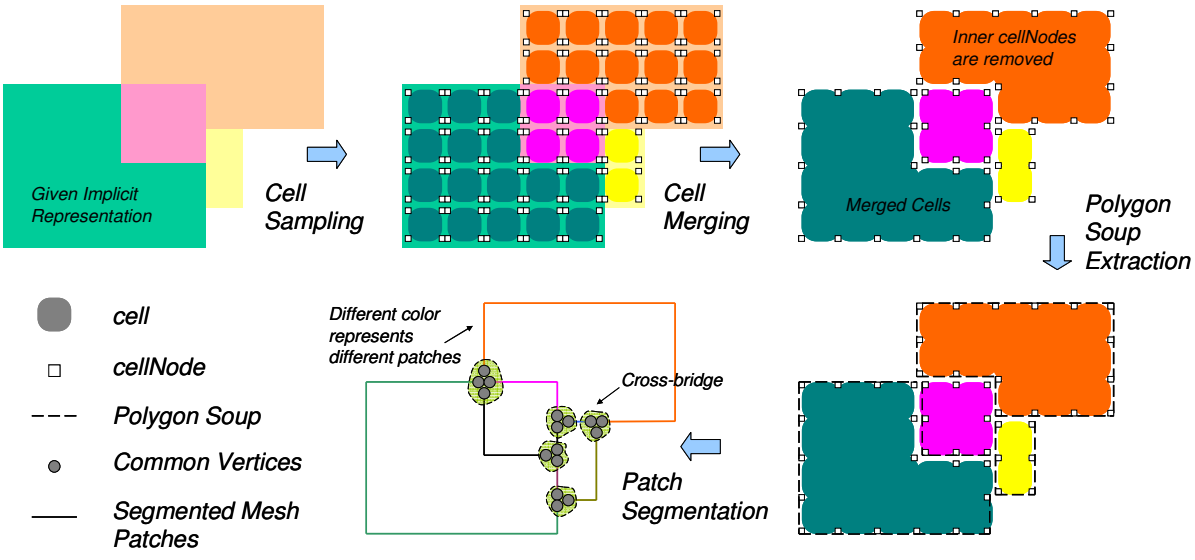


Fig. 5 2D illustration of the extraction and segmentation of two-manifold coarse patches

Table 1 Entities: *cell*, *cellNode*, and *cellFace*

```
Class cell {
    cellNode* nodes[8]; // the pointers of eight nodes in a cell
    cellFace* faces[6]; // the pointers of six faces in a cell
    bool bMerged[6];    // the flags to identify whether the cell in the ith direction has been merged with this one
    int material;       // the index of material class with maximal volume in this cell
}
Class cellNode {
    float pos[3];       // the position of this cellNode
    cell** cellSet;     // the set of cells containing this cellNode
    cellFace** faceSet; // the set of cellFaces containing this cellNode
}
Class cellFace {
    cellNode* nodes[4]; // the pointers of four nodes in a face in anti-clockwise direction
    int posMaterial, negMaterial; // index of material class at the positive/negative direction region of this face
}
```

4. Extraction and segmentation of two-manifold coarse patches

In the first phase of our algorithm, we sample the definition domain Ψ of a given heterogeneous object *H* into cells with uniform size, merge the cells to generate a polygonal soup, and segment the soup into linked two-manifold mesh patches. The two-manifold topology is preserved on every step.

4.1. Cell Sampling

The definition domain $\Psi \subset \Re^3$ bounded by $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$ is first divided into $M \times N \times L$ sub-regions with a user specified width h_c, where a sub-region $\psi_{i,j,k}$ is defined as a cubic space with the interval $[x_{\min} + ih_c, x_{\min} + (i+1)h_c] \times [y_{\min} + jh_c, y_{\min} + (j+1)h_c] \times [z_{\min} + kh_c, z_{\min} + (k+1)h_c]$. The volumes of regions occupied by different materials are computed in each $\psi_{i,j,k}$, including the region of material class *zero* – no material region. For a material with the type index ξ which covers the maximal volume, if $\xi \neq 0$, a cell $C_{i,j,k}$ is created on this sub-region, and the *material type* in $C_{i,j,k}$ is then assigned to ξ. The data structures of a *cell* entity and its related *cellNode* entity are listed in Table 1, where every cell is a box containing eight nodes, six faces, and the flags about whether the cell has merged it corresponding nodes with the adjacent cells in six Cartesian directions ($\pm x, \pm y, \pm z$). When constructing a *cell C*, eight *cellNode*s are also created at the locations of $(c_c.z \pm \frac{h_c}{2}, c_c.y \pm \frac{h_c}{2}, c_c.z \pm \frac{h_c}{2})$ and filled in the related slots in *C*, where c_c is the center of cell. Every *cellNode* has this cell *C* stored in its *cellSet*.

Analysis for Two-manifold Preservation: For the six *cellFace*s in a cell, we just leave them *null* now. They will be constructed and filled later. If all *cellFaces* are filled at this moment, the model constructed is an object with a lot of cubes each bounding a cell filled with one homogeneous material. In fact, the object is represented by many two-manifold cubes with the uniform size at this moment.

4.2. Cell Merging

Each *cell* created during cell sampling holds eight *cellNode*s. As a necessary step of constructing two-manifold polygonal surface patches, the nodes at adjacent cells with the same material type are merged. This is called *cell merging*.

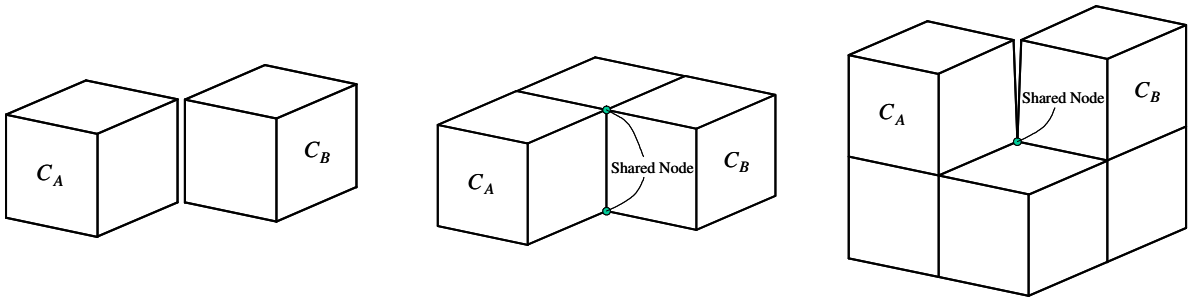
Before describing the process of cell merging, we need to introduce a common operator utilized in cell merging: *node-merge*. The operator *node-merge* is applied on two coincident nodes: v_1 and v_2 ($v_1 \neq v_2$). The three steps of *node-merge* are:

Step 1: Replacing v_2 in all cells containing it by v_1 (the cells containing v_2 can be easily accessed by the *cellSet* of v_2);

Step 2: Add all *cell*s in the *cellSet* of v_2 into the cellSet of v_1;

Step 3: Making the *cellSet* of v_2 empty.

The merging operation is then applied on every cell $C_{i,j,k}$: for every Cartesian direction of $C_{i,j,k}$ (i.e., $i \pm 1, j \pm 1, k \pm 1$), we detect whether there is an adjacent cell C_a of $C_{i,j,k}$ having the same material type. If there has such an adjacent cell and no previous merge (detected by the *bMerged* flag) is applied between them, we merge their four coincident nodes by *node-merge*. After checking and merging all *cell*s, all the *cellNode*s with their *cellSet* empty are removed finally.



(a) no node shared **(b) two nodes shared** **(c) one node shared**

Fig. 6 Case study of results after *cell merging*

Analysis for Two-manifold Preservation: By the operations conducted above, all inner nodes of a homogeneous region are removed. Now, if the cellFaces are constructed on boundary nodes, the resultant polygonal surfaces are two-manifold. Each encloses a homogenous region – they are two-manifold. However, on the interface between two cells with different materials, duplicate faces will be given. By applying the following polygon soup extraction steps, we can construct a polygon soup which shows single-layer intermediate surfaces. With the help of cell merging, non-manifold is definitely prevented on the boundary of any Ω_i. For instance, three different cases of two cells C_A and C_B having only one edge coincident are illustrated in Fig.6, where C_A and C_B contain the same material stuff. Correct results of the three cases are automatically generated through cell-merging according to different configurations around the coincident edge.

4.3. Polygon Soup Extraction

After cell merging, the polygon soup S approximating Γ will be extracted. The polygon soup here is a collection of quadrilateral polygons - *cellFace*s. The data structure of a *cellFace* entity has already been given in Table 1. The *cellFace*s in polygon soup are constructed by the following rules:

Rule 1: Every cell $C_{i,j,k}$ is detected in six Cartesian directions to see whether there is an neighboring cell C_a.

Rule 2: If there is no C_a neighboring to $C_{i,j,k}$, a *cellFace* $cf_{\alpha,\beta}$ is created by the four nodes of $C_{i,j,k}$ in the corresponding Cartesian direction. The order of nodes in $cf_{\alpha,\beta}$ is to let its normal pointing outwards

$C_{i,j,k}$, where the subscripts α and β represent the material type in positive/negative directions (i.e., so $\alpha = 0$ and $\beta = \xi(C_{i,j,k})$ with $\xi(C_{i,j,k})$ giving the material index of $C_{i,j,k}$). $cf_{\alpha,\beta}$ is then filled into the related slot of $C_{i,j,k}$.

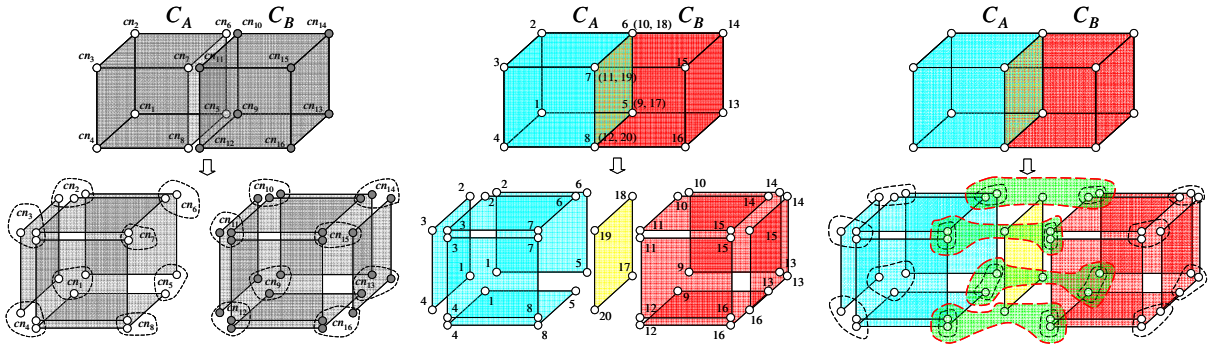
Rule 3: If there is a neighboring cell C_a to $C_{i,j,k}$, the material indexes in C_a and $C_{i,j,k}$ are detected – adjacent cells with the same material need no *cellFace* in between; also, if a *cellFace* $cf_{\alpha,\beta}$ between C_a and $C_{i,j,k}$ has been generated, no *cellFace* is needed any more.

Rule 4: If a *cellFace* $cf_{\alpha,\beta}$ needs to be generated to separate C_a and $C_{i,j,k}$ (i.e., $\xi(C_a) \neq \xi(C_{i,j,k})$), the $cf_{\alpha,\beta}$ is created by connecting four nodes in the cell with greater $\xi(\cdots)$, and the normal of $cf_{\alpha,\beta}$ points to the cell with smaller $\xi(\cdots)$. The values of α and β are assigned by the related $\xi(\cdots)$ s. Also, the newly created $cf_{\alpha,\beta}$ is filled in its related slots of both C_a and $C_{i,j,k}$.

Rule 5: After creating any *cellFace* $cf_{\alpha,\beta}$, the *faceSet* of each *cellNode* in $cf_{\alpha,\beta}$ will be updated appropriately, where the *faceSet* of a *cellNode* contains all the $cf_{\alpha,\beta}$ s adjacent to this node.

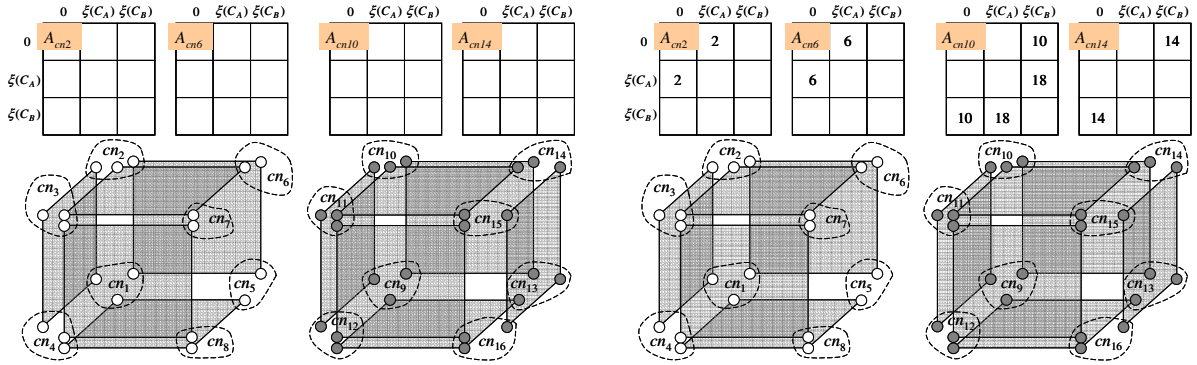
An example of polygonal soup generated by above rules on two cells enclosing different materials with $\xi(C_B) > \xi(C_A)$ is shown in Fig. 7a. Following Rule 1, 3 and 4, the *cellFace* between C_A and C_B is created by four nodes (i.e., *cellNode*s with no.9-12) in C_B and with the order 9-12-11-10 so that the normal of the face pointing to C_A - the cell with smaller material index.

Analysis for Two-manifold Preservation: For *cellFace*s on the adjacent *cell*s with different materials, the duplication is removed after this step of algorithm. Considering about a region of material type ξ, it is bounded by all cellFaces, $cf_{\alpha,\beta}$, with either $\alpha = \xi$ or $\beta = \xi$. If flipping all *cellFaces* $cf_{\xi,\beta}$ into $cf_{\beta,\xi}$, we obtain a two-manifold polygonal surface approximating Ω_ξ.



(a) faces and nodes in a polygon soup before segmentation **(b) polygonal patches after segmentation** **(c) segmented patches with cross-bridges**

Fig. 7 An example of the two-manifold patch segmentation on a polygon soup with two cells C_A and C_B ($\xi(C_B) > \xi(C_A)$) – different colors represent different mesh patches after segmentation where blue denotes the patch $M_{A,0}$, yellow is the patch $M_{B,A}$, and red is for the patch $M_{B,0}$. The numbers denote different vertices.



(a) alias-matrices are constructed on every cellNode – indices represent the related regions

(b) after the *vertices construction* step, the vertices are created and filled in the alias-matrices

Fig. 8 An illustration for explaining the functionality of alias-matrices (four selected matrices for the cellNode cn_2, cn_6, cn_{10} and cn_{14} are shown in the figure).

4.4. Two-manifold Patch Segmentation

A polygon soup is going to be further segmented into a collection of two-manifold mesh surface patches, where each mesh surface patch $\mathrm{M}_{i,j}$ is defined as a pair (K,V), where K is a simplicial complex specifying the connectivity of vertices, edges, and faces (i.e., the topological graph of $\mathrm{M}_{i,j}$), and $V=\{v_1,\cdots,v_m\}$ is the set of vertices defining the shape of a polyhedral patch in $\Re^3$. From K, it is straightforward for our algorithm to fetch the adjacent nodes, edges, and faces of a triangular node in constant time. The above definition follows the notation in [31].

Before presenting the segmentation algorithm, let us first introduce the concept of *common vertex*. Suppose a common point *p* shared by *m* intermediate surfaces of *H*, since each $\Gamma_{ij}=\Gamma_{ji}$ is requested to be represented by a manifold mesh $\mathrm{M}_{i,j}$ (*i*<*j*), totally *m* vertices need to be constructed at *p* – everyone belongs to different mesh patches. To prevent the duplication, no $\mathrm{M}_{j,i}$ (*i*<*j*) will be constructed. The *m* vertices are kept coincident – they are called *common vertices*. For one of the *m* common vertices *v*, the set of other (*m*-1) common vertices are denoted by $\Theta(v)$. In the finally constructed $\mathrm{M}=\bigcup_{i=0,j>i}^{n}\mathrm{M}_{i,j}$, every common vertex *v* stores the links to the other (*m*-1) common vertices, $\Theta(v)$, sharing the same position. The links are named as *cross-bridge*s.

The position of every node in the polygon soup S could possibly be with a set of common vertices. Hence, in our segmentation algorithm, a $(n+1)\times(n+1)$ alias-matrix $A_{cn}=\{a_{i,j}\}$ is kept at every cellNode $cn\in S$, where its entry $a_{i,j}$ is an alias of a vertex in $\mathrm{M}_{i,j}$ located coincident to cn. When $a_{i,j}=\phi$, it means that there is no related vertex in $\mathrm{M}_{i,j}$. After creating the alias-matrices on every cellNode, we go through all cellFaces in S *twice* to create vertices and update the alias-matrix of every node in the first round, and then create faces in the second round.

1) **Vertices Construction:** For any cellFace $cf_{\alpha,\beta}$ in S contains the cellNode cn, if $a_{\alpha,\beta}=\phi$ ($\forall a_{\alpha,\beta}\in A_{cn}$), we create a new vertex v_{cn} in $\mathrm{M}_{\alpha,\beta}$ and let $a_{\alpha,\beta}=v_{cn}$.

2) **Faces Construction:** After creating all necessary vertices, we construct faces by every cellFace $cf_{\alpha,\beta} \in S$. The generated faces are stored in mesh $\mathrm{M}_{\alpha,\beta}$. For a face created according to $cf_{\alpha,\beta}$, the face is composed of the vertices indicated by $a_{\alpha,\beta} \in A_{cn}$ and with the same orientation as $cf_{\alpha,\beta}$, where cn is a *cellNode* in $cf_{\alpha,\beta}$. In other words, the newly created faces are connected to the vertices stored in the alias-matrix of a *cellNode* but not the *cellNode* itself.

The faces in M could be either triangular or quadrilateral. However, our latter remeshing operations requires triangular element, so two triangles are constructed corresponding to every *cellFace*. Following these, we can easily create every mesh patch $\mathrm{M}_{i,j}$ corresponding to the intermediate surface $\Gamma_{ij} = \Gamma_{ji}$ of *H*.

For the example polygon soup S with two cells of different material shown in Fig.7a, the obtained mesh patches is as shown in Fig.7b – different colors represent different mesh patches (i.e., three patches are generated). When segmenting the mesh patches from a polygon soup, the alias-matrices play a significant role. As shown in the example of Fig.7 and 8, with the help of the alias-matrices, we can easily separate the vertices no.17-20 from the vertices no.9-12 (see Fig.7b) – for example the cellNode cn_{10} in Fig.7a, two vertices are created (vertex 10 and 18 in Fig.7b) and filled into its alias-matrix as $a_{\xi(B),\xi(A)}$ (vertex no.18 for the mesh patch $\mathrm{M}_{\xi(B),\xi(A)}$) and $a_{\xi(B),0}$ (vertex no.10 for the patch $\mathrm{M}_{\xi(B),0}$); for the cellNode cn_6, only one vertex (vertex no.6 for the patch $\mathrm{M}_{\xi(A),0}$) is created and filled into its alias-matrix as $a_{\xi(A),0}$. By the alias-matrices, we can fill the correct vertices into the face in a constant time (just using the index to search a vertex) – it is very efficient. After processing all nodes and faces in the polygon soup *S*, a collection of mesh patches has been created where each patch $\mathrm{M}_{i,j}$ (*i*<*j*) is a two-manifold open surface.

Finally, the linking information of common vertices is constructed on the boundary vertices of each mesh patch. Our approach is based on the positions of boundary vertices. For the boundary vertices that are coincident, the cross-bridges are constructed on them. The coincident can be easily and efficiently detected by a uniform subdivision of the definition domain $\Psi \subset \Re^3$. However, this time every grid box is with mesh vertices at its center instead of corners. In detail, a sub-region $\varpi_{i,j,k}$ is a cubic space with the interval: $[x_{\min} + (i-\frac{1}{2})h_c, x_{\min} + (i+\frac{1}{2})h_c] \times [y_{\min} + (j-\frac{1}{2})h_c, y_{\min} + (j+\frac{1}{2})h_c] \times [z_{\min} + (k-\frac{1}{2})h_c, z_{\min} + (k+\frac{1}{2})h_c]$. If some boundary vertices fall in the same sub-region $\varpi_{i,j,k}$, the cross-bridges are constructed among them. These vertices should be maintained coincident in the later refinement steps to achieve the compatible boundaries.

The finally segmented $\mathrm{M} = \bigcup_{i=0,j>i}^{n} \mathrm{M}_{i,j}$ is a collection of two-manifold mesh patches, which gives an approximation of Γ in coarse level. For the example in Fig.7, the result is as shown in Fig.7c, where the nodes inside a black dash circle represent a unique vertex and the vertices surrounded by a red dash circle are common vertices linked by cross-bridges. Our later processes will refine M to make its approximation to Γ become more accurate.

Analysis for Two-manifold Preservation: For the homogeneous region Ω_ξ, it is approximated by the collection of all patches $\mathrm{M}_{i,\xi}$ and $\mathrm{M}_{\xi,j}$, where the normal of each triangle on $\mathrm{M}_{i,\xi}$ facing outwards but the normal vectors on $\mathrm{M}_{\xi,j}$ facing inwards. Since the meshes are created according to the two-manifold preserved polygon soups, they are two-manifold. In summary, every steps of above algorithm preserve two-manifold topology on the constructed model, so the final result guarantees two-manifold on each mesh patch. Fig. 5 has already given an illustration of our algorithm on 2D, where a collection of two-manifold patches are extracted from the given implicit representation. Our results follow the regularization theory (ref. [50]) for computing a two-manifold model from non-manifold objects.

5. Signed distance-fields for different material stuff

In this section, discrete signed distance-fields are constructed, which will govern the movement of mesh vertices in the following remeshing phase. For a give region Ω_α with material class α in *H*, a signed distance-field defined on Ω_α is a function $D_\alpha(p)$ assigning to every point $p=(x,y,z)\in\Re^3$ its distance $D_\alpha(p)=dist(p,\partial\Omega_\alpha)$, where a positive sign for points $p\notin\Omega_\alpha$, a negative sign for points $p\in(\Omega_\alpha-\partial\Omega_\alpha)$, and $D_\alpha(p)=0$ if $p\in\partial\Omega_\alpha$. A convenient way to store the distance field $D_\alpha(p)$ for Ω_α in an efficient data structure is to sample $D_\alpha(p)$ on uniform spatial grids with nodes $d_{i,j,k}=(i\Delta h, j\Delta h, k\Delta h)$. For a point $p=(x,y,z)$ with $x\in[i\Delta h,(i+1)\Delta h)$, $y\in[j\Delta h,(j+1)\Delta h)$ and $z\in[k\Delta h,(k+1)\Delta h)$, its Euclidean distance to $\partial\Omega_\alpha$ can be interpolated on the grid cell $[i\Delta h,(i+1)\Delta h)\times[j\Delta h,(j+1)\Delta h)\times[k\Delta h,(k+1)\Delta h)$ by a tri-linear function such that we obtain a piecewise tri-linear approximation $\overline{D_\alpha}(p)$ for the original distance-field $D_\alpha(p)$. At the meanwhile, a corresponding isosurface $\overline{\partial\Omega_\alpha}$ defined by $\overline{D_\alpha}(p)=0$ gives an approximation to $\partial\Omega_\alpha$. The unit normal vector of $\partial\Omega_\alpha$ at any point $p\in\partial\Omega_\alpha$ can be calculated by the negative gradient $\nabla\overline{D_\alpha}(p)$ of $\overline{D_\alpha}(p)$ at *p* as

$$\overline{n_\alpha}(p)=-\nabla\overline{D_\alpha}(p)\Big/\left\|\nabla\overline{D_\alpha}(p)\right\|, \tag{3}$$

where the gradient vector can be simulated by the numerical central differences

$$\nabla\overline{D_\alpha}(p)=\frac{1}{2h}\begin{pmatrix}\overline{D_\alpha}(x+h,y,z)-\overline{D_\alpha}(x-h,y,z)\\ \overline{D_\alpha}(x,y+h,z)-\overline{D_\alpha}(x,y-h,z)\\ \overline{D_\alpha}(x,y,z+h)-\overline{D_\alpha}(x,y,z-h)\end{pmatrix}^T. \tag{4}$$

The smaller grid size Δh chosen, the more accurate approximation of $D_\alpha(p)$ is given by $\overline{D_\alpha}(p)$, but more computer memory is needed for storing $\overline{D_\alpha}(p)$. An alternative solution is the adaptively sampled distance-field as shown in [36]. In our current implementation, we uniformly sample the definition domain Ψ of *H* to detect whether a sampled grid node $d_{i,j,k}$ is in Ω_α. For a grid node $d_{i,j,k}\in\Omega_\alpha$ with one of its neighborhood $d_{i\pm1,j\pm1,k\pm1}\notin\Omega_\alpha$, we assign the distance from $d_{i,j,k}$ to $\partial\Omega_\alpha$ as *zero*. Then, after setting the distance values of other sample points to ∞, the vector distance transforms (VDTs) presented in [51] is applied to propagate the distance values. The sign of distance at every sample point (x_i,y_j,z_k) can be detected by whether $d_{i,j,k}\in\Omega_\alpha$.

If $d_{i,j,k} \in \Omega_\alpha$, the sign is negative; otherwise, a positive sign is given. The reason why we did not choose the fast marching method [52] is that, in the routine of fast marching, it needs repeatedly query the node with the smallest distance in its current marching set (stored by a minimal heap). The time complexity of the query is O(log*T*) if there are *T* nodes in the heap (in the worst case, *T* could be a very large number), which makes the fast marching slower than the VDTs in [51] where the processing time of every node is a constant.

6. Adaptive remeshing

By the signed distance-fields generated above, in this section, the mesh patches in collection M are refined and optimized to give an adaptive and quality approximation of Γ. The operations adopted here include: vertex repositioning, adaptive refinement, element shape optimization, normal preservation, and final contouring. They will be iteratively applied on M in the remeshing algorithm, where the refinement and optimization of M are governed by the signed distance-fields.

Vertex Repositioning: The purposes of this operation are to 1) move all vertices of M towards Γ and 2) relax the distribution of vertices on M. For a vertex $v \in \mathrm{M}_{i,j}$, to let $\mathrm{M}_{i,j}$ accurately approximate Γ_{ij}, v should lie on Γ_{ij}. However, the surface Γ_{ij} is not explicitly given, so we conduct two isosurfaces - $\overline{D_i}(p) = 0$ and $\overline{D_j}(p) = 0$ to simulate Γ_{ij}. The vertex v is attracted to move towards these two isosurfaces. At the same moment, in order to improve the mesh regularity when moving the vertices, every vertex is expected to be close to the average position of its one-ring neighbors. In summary, the following functional is defined to govern the vertex repositioning

$$\arg\min_{v} \left\{ \omega\left(\overline{D_i}(v)\right)^2 + \omega\left(\overline{D_j}(v)\right)^2 + \lambda \left\| \frac{1}{n}\sum_{k=1}^{n} q_k - v \right\|^2 \right\}, \tag{5}$$

where in the third term the q_k s are the one-ring neighbors of *v*. For the weights of functional terms, we choose $\omega = 0.125$ and $\lambda = 0.25$ in our implementation to balance the weights of attraction and relaxation. To make boundary curves smooth and maintain the position continuity between linked patches, a special functional is needed for relocating common vertices. Suppose that the vertex v_c is a common vertex lying on a common boundary of several mesh patches - M_P s, the collection L includes the indexes of all material regions at the both sides of every M_P. The functional on v_c is then defined by

$$\arg\min_{v_c} \left\{ \lambda \left\| \frac{1}{m}\sum_{k=1}^{m} \hat{q}_k - v_c \right\|^2 + \omega \sum_{i \in L} \left(\overline{D_i}(v_c)\right)^2 \right\}. \tag{6}$$

The $\hat{q}_k$ s are boundary vertices in M adjacent to all common vertices shared with v_c (including v_c). The weights in this functional are again set as $\omega = 0.125$ and $\lambda = 0.25$. By minimizing the functionals given in Eq. (5) and (6) iteratively, the vertices in M are moved closer and closer to Γ while being relaxed.

Adaptive Refinement: To make the approximation of Γ adaptive to its shape, we split some triangular edges to introduce new vertices. This is very important for the surfaces with large curvatures. Considering about any

triangular edge $e \in \mathrm{M}_{i,j} \subset \mathrm{M}$ with p_e as its middle point, if $\frac{\left\|\overline{D_i}(p_e)\right\| + \left\|\overline{D_j}(p_e)\right\|}{2} > \varepsilon_r$, we insert a new vertex at p_e. In our implementation, we choose $\varepsilon_r = 0.75\Delta h$ where Δh is the sampling rate of underlying distance-fields. After checking all triangular edges on M, several new vertices are created. If the two vertices v_s and v_e of a divided edge are both common vertices, to eliminate T-junctions on the boundary of surface patches, all other edges with vertices having cross-bridges linking to both v_s and v_e must also be split by inserting new common vertices in the middle; also, the cross-bridges need to be constructed on the newly created common vertices.

Element-Shape Optimization: Two types of element shape optimizations are conducted during the remeshing of M to amend the shape of its elements. They are *edge collapse* and *edge swap* in [31]. These two operators work together to iteratively remove triangles with extreme small and large angles. Note that the operation leads to topology degeneration cases shown in [53] should be prevented. An illustration for how the degeneration happens is given in the Appendix of this paper. Again, to avoid T-junctions the operations on boundary edges are not allowed either.

Normal Preservation: This is an optional operation. The function of this operation is to preserve sharp features on M through minimizing the difference of each face's normal and the normal given by Γ at its center. The preservation is achieved through two phases of vertex position optimization. By moving mesh vertices iteratively, we can make the normal of every mesh triangle $f \in \mathrm{M}_{i,j}$ closer to the reference normal given by $\overline{D_i}(p_c)$ and $\overline{D_j}(p_c)$ at the centroid p_c of f (details refer to [41]). The first phase of normal preservation is to update every vertices in M except the boundary ones for several iteration steps to optimize normals on their adjacent triangles. The sharp-feature vertices are then detected through the curvature tensors (ref. [54]). The discrete mean curvature at every interior vertex is computed (eq.(8) in [54]) – if the mean curvature is greater than a user specified threshold, it is identified as a sharp-feature vertex. Note that, the threshold can be determined in a trial-and-error manner or be specified interactively: the user can specify a few vertices he thought to be on a sharp edge and some others not on a sharp edge, these vertices can then be used to compute the threshold for classification. After classifying vertices, we start the second phase of normal preservation. For the non-sharp-feature vertices, we reposition them by minimizing the functional given in Eq.(5) or Eq.(6); while for the sharp-feature vertices, we go on updating their positions to let its adjacent triangles satisfy the reference normal vectors.

Final Contouring: Finally, after using above operations to change the geometry and connectivity of M, we go through every inner edge twice to see whether *edge swap* is necessary. In the first run, the distances from the middle point of every edge and its dual edge to Γ are compared. The one with smaller distance is expected. The second pass detects normals on the adjacent faces of every edge and its dual edge. If the normals of faces adjacent to the dual edge gives better approximation of Γ (simulated by distance fields), edge swap is applied on the edge to enhance the approximation of Γ by M - this greatly improves the sharp features on resultant meshes. Again, the *edge swap* leading to topology degenerations must be prevented.

By iteratively applying above operations to the two-manifold coarse patches, we can finally construct the adaptive and quality M which gives a good approximation of Γ. The overall remeshing algorithm is written in pseudo-code in Table 2. For the loop of **Procedure** *Remeshing*, the terminal condition of iteration is that: $\frac{1}{2}(\|\overline{D_i}(p_e)\| + \|\overline{D_j}(p_e)\|) \le \varepsilon_r$ is satisfied on every $e \in \mathrm{M}_{i,j} \subset \mathrm{M}$ with p_e as its middle point, where $\varepsilon_r = 0.75\Delta h$ – the same as what we used in *Adaptive Refinement*. Since all operations utilized in *remeshing* preserves 1) two-manifold on a mesh patch and 2) continuity cross linked patches, the resultant M of remeshing is guaranteed to be topologically valid. Every homogeneous region in the given heterogeneous object Γ is explicitly represented by the assembled two-manifold mesh patches; the elements on meshes are with good shape, and are adaptive to the curvature of Γ.

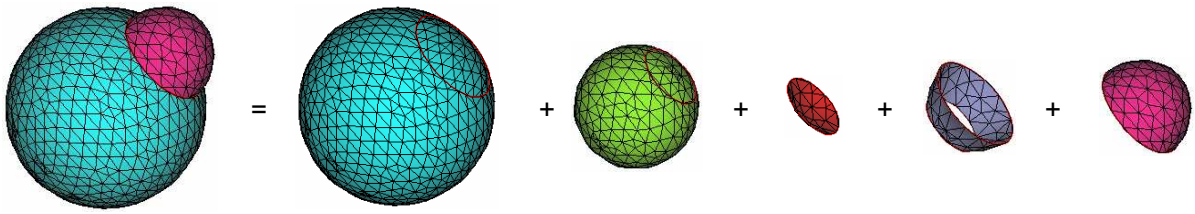
Table 2 Procedure *Remeshing*

```
Procedure Remeshing( M )
{
    Vertex Repositioning;
    Do {
        Adaptive Refinement;
        Vertex Repositioning;
        Element-Shape Optimization;
    }While(the terminal condition of refinement is NOT satisfied);
    Vertex Repositioning;
    Normal Preservation;        // optional
    Final Contouring;
}
```

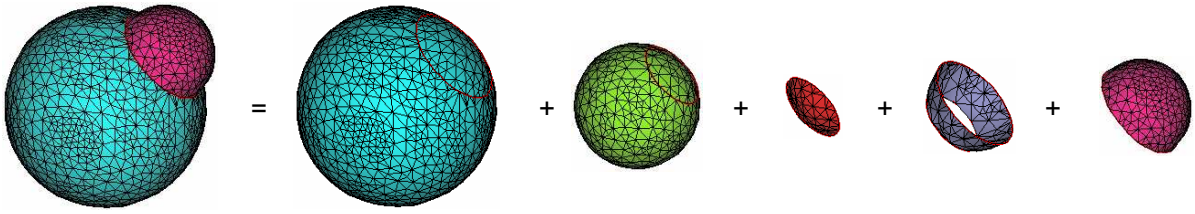
7. Experimental Results

In this section, several experimental results will be shown. Our first example is a heterogeneous volume consists of three materials which has been previously shown in Fig.1a. The coarse meshes and signed distance-fields extracted from the implicitly represented *H* have already been listed at the top two rows of Fig.4. The progressive results during **Procedure** *Remeshing* are shown in Fig.9. It is easy to find that the remeshing procedure significantly improves the quality of M approximating Γ. N_v, N_e, and N_f denote the number of vertices, edges, and faces in M respectively.

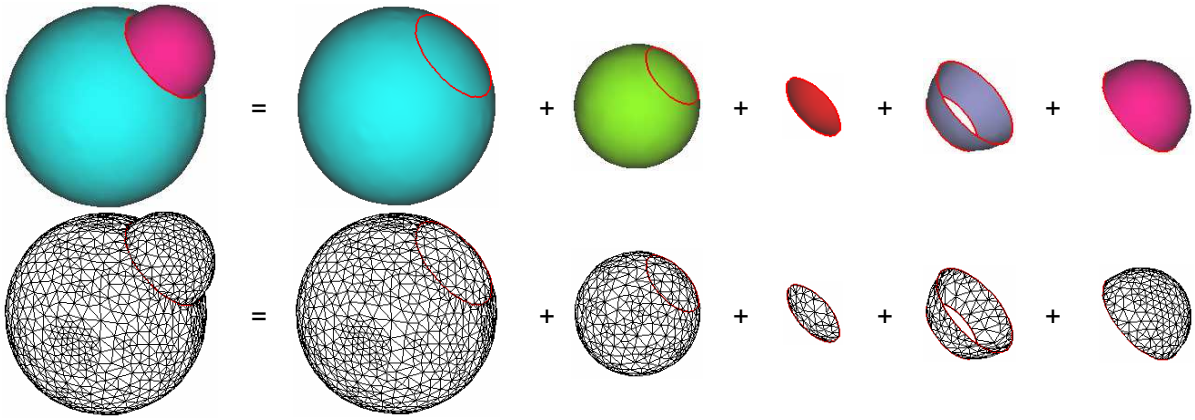
Example II demonstrates the performance of sharp feature preservation in our algorithm. The implicitly represented heterogeneous object *H* of two materials is given as shown in Fig.10a with 0.5mm sampling accuracy. The coarse two-manifold meshes in M are extracted and segmented by choosing $h_c = 3$ mm (see Fig.10b). If no normal preservation is applied on M in **Procedure** *Remeshing*, the resultant object will degenerate to smooth regions at the place where sharp edges are expected (see Fig.10c). In the first phase of normal preservation, every inner vertices on M are moved to preserve normals on its neighboring faces; however, unexpected sharpening is generated on some regions expected to be smooth (e.g., the surface of sphere becomes unsmooth in Fig.10d). To further enhance M, the vertices on sharp features are detected (i.e., the vertices with small cubes in Fig.10e); then the second phase of normal preservation and the final contouring are applied on M to give the final result as shown in Fig.10f. To illustrate sharp features clearly, we adopt the flat shading to display the results in Fig.10.



(a) after *vertex repositioning*: $N_v = 1234$, $N_e = 3534$, **and** $N_f = 2304$

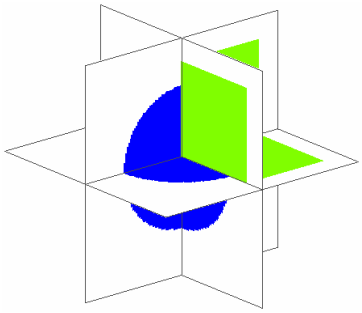


(b) after partially *refinement* and *element-shape optimization*: $N_v = 2132$, $N_e = 6102$, **and** $N_f = 3974$

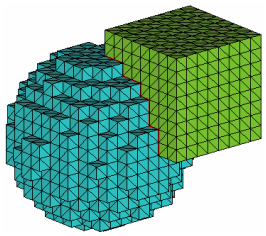


(c) the final result: $N_v = 2209$, $N_e = 6303$, **and** $N_f = 4098$

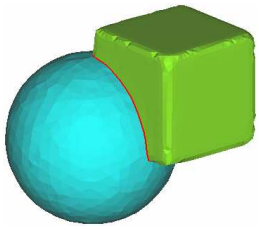
Fig. 9 Progressive results of Example I during remeshing



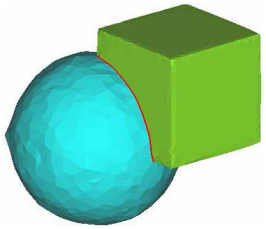
(a) the given implicit heterogeneous object



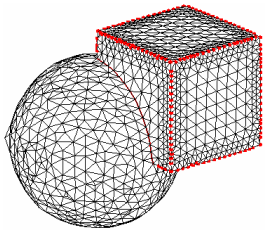
(b) the coarse meshes before remeshing



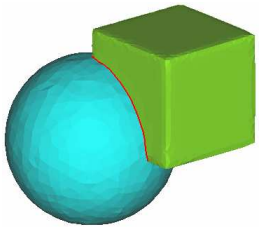
(c) the result without normal preservation



(d) the noisy mesh after Phase I of normal preservation



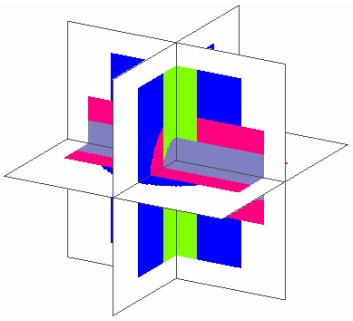
(e) the vertices in sharp feature region have been detected



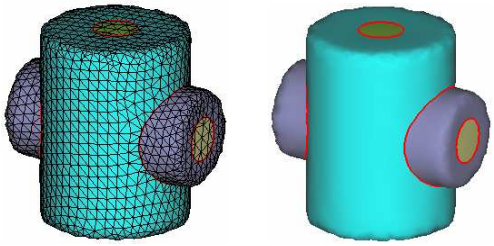
(f) the final result of our remeshing algorithm

Fig. 10 Example II: demonstrate the performance of sharp feature preservation

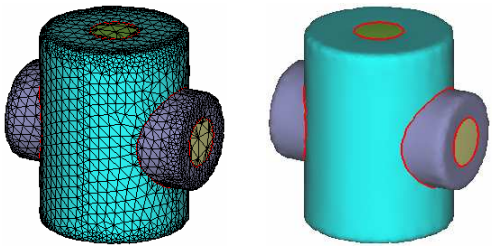
In our example III, a heterogeneous object with four materials is given (see Fig.11a). Our surface extraction algorithm provides two parameters to control the resolution of resulting meshes: one is the size of cells h_c that determines the initial mesh resolution before remeshing, and the other is the sampling accuracy of underlying distance-fields Δh which effect on the *Adaptive Refinement*, *Vertex Repositioning*, and *Normal Preservation* in **Procedure** *Remeshing*. Comparing the results given in Fig.11b and 11c, since they adopt the same h_c, the elements at flat and curved regions are with similar size; however, with smaller Δh, more elements have been created to adapt high curvature regions on Γ of H. When keeping the same Δh, but using smaller h_c (e.g., the one in Fig.11d), the elements at low curvature regions become smaller either – so more elements are generated. It is not difficult to find that every mesh patch in the final result gives very smooth boundary curves, which are illustrated by the red color edges.



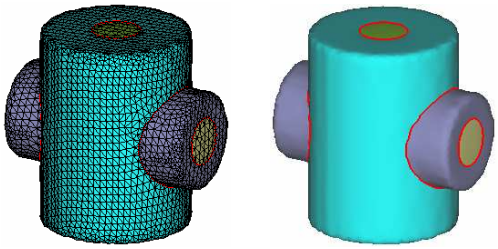
(a) the given implicit heterogeneous object



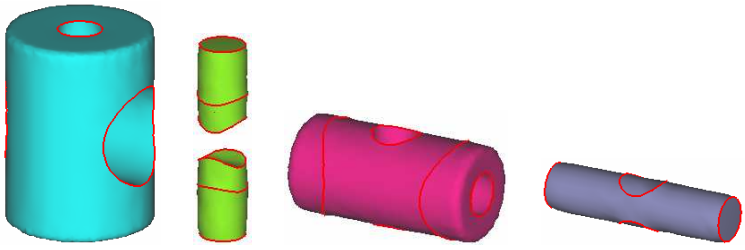
(b) the result with $h_c = 3$ mm and $\Delta h = 1$ mm ($N_v = 2854$, $N_e = 7863$, and $N_f = 5009$)



(c) the result with $h_c = 3$ mm and $\Delta h = 0.5$ mm ($N_v = 5143$, $N_e = 13878$, and $N_f = 8735$)

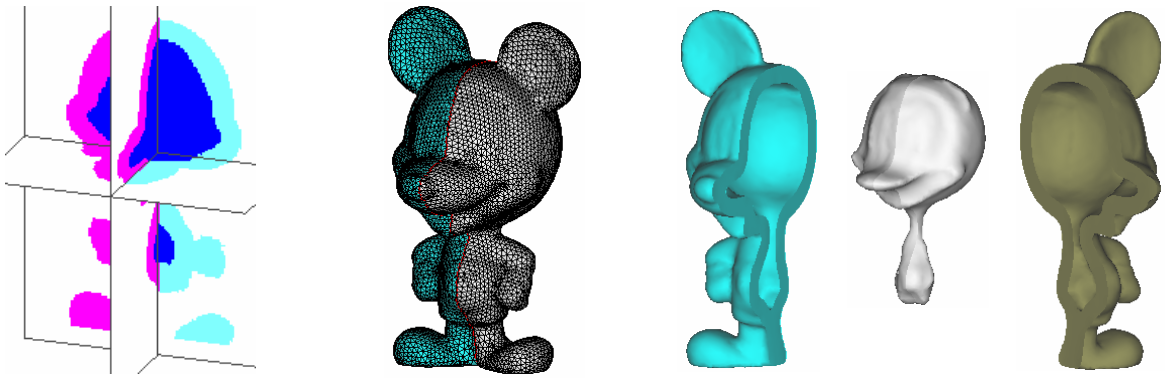


(d) the result with $h_c = 2$ mm and $\Delta h = 0.5$ mm ($N_v = 6382$, $N_e = 17621$, and $N_f = 11239$)



(e) the parts with different classes of materials

Fig. 11 Example III: the approximation of Γ in different resolutions



(a) the given implicit representation **(b) the extracted surface** **(c) the parts with different classes of materials**

Fig. 12 Example IV: bounding surfaces of different material regions for a freeform model

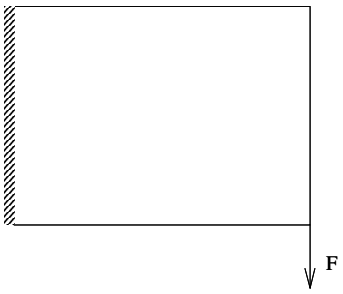
The mesh patches M extracted in our algorithm, which approximate the interfaces Γ of a given implicit heterogeneous object *H*, can also be employed to generate two-manifold mesh surfaces for freeform fabrication of heterogeneous structures (e.g., the multi-volume B-rep models requested by [55] can be easily generated). Suppose the part of material class X in M needs to be fabricated, its related STL file is produced by the following two steps: 1) for a two-manifold mesh patch $\mathrm{M}_{i,j} \in \mathrm{M}$, if $j = \mathrm{X}$, every triangle in $\mathrm{M}_{i,j}$ is written into the STL file, and 2) for the patch $\mathrm{M}_{i,j} \in \mathrm{M}$ with $i = \mathrm{X}$, its triangles are also written into the STL file, but in the opposite direction – i.e., when writing, the order of nodes must be reversed. Fig.11e shows the parts in STL files with different materials on Example III, and Fig.12 demonstrates this on a freeform model.

Our fifth example comes from the application of structure optimization. For a cantilever beam loaded vertically at the bottom of its free end as shown in Fig.13a, to minimize the mean compliance by distributing six given materials, the heterogeneous structure can be optimized by using the "color" level set method [2]. When the resultant structure is implicitly represented as Fig.13b, our approach presented in this paper can extract mesh representation of the heterogeneous structure. Both the bounding surfaces of structure and the intermediate surfaces inside will be constructed. The resultant surfaces are given in Fig.13c, from which it is not difficult to find that our method works well on narrow geometries (as long as the thickness of the geometry is greater than Δh and h_c). As shown in the zoom-view – Fig.13d, a sharp intersection angle between two boundaries can also be reconstructed successfully.

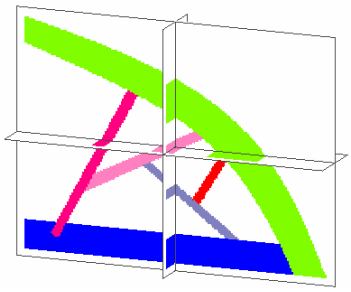
The method presented in this paper also works for the applications of biomedical engineering (e.g., Fig.14). After segmenting the CT volume images of a human brain, we obtain an implicitly represented volume data (see Fig.14a, where different colors represent different tissues). By the approach presented in this paper, the mesh surfaces approximate the interfaces between the tissues are successfully extracted (see Fig.14b). Meshes on the reconstructed surface are adaptive to curvatures (see the meshes in Fig.14c). By the resultant model from our approach (e.g., in Fig.14d), it is easy to generate the B-rep models for different tissues to be applied in the further simulation and processing.

By the examples shown in the Fig.12 and 14, our approach has been proved its ability to process the meshes with vertices at the range of tens thousand (in detail, example IV is with 12,200 vertices and example VI is with 29,932 vertices). When processing these two examples, only about 200 to 300 MB RAM is used. Also, the

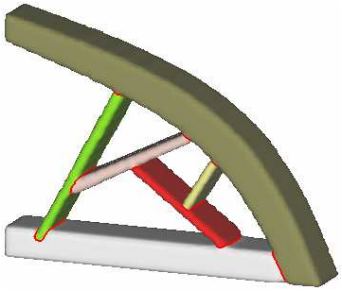
memory usage in our approach is linearly proportional to the number of mesh vertices. Therefore, it is not difficult to conclude that the proposed approach can process the meshes with even >100K vertices.



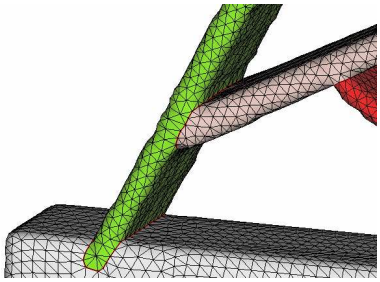
(a) structure to be optimized



(b) optimized structure with five materials in implicit representation

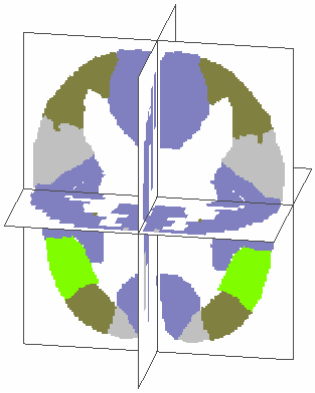


(c) mesh surface extracted

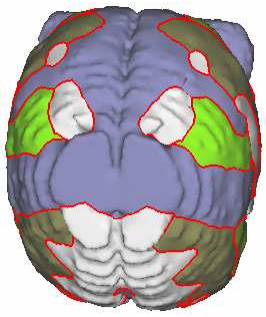


(d) zoom-view

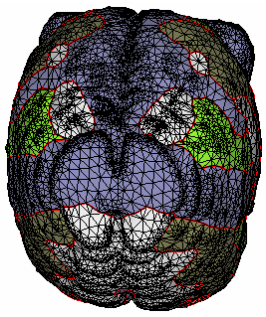
Fig. 13 Example V: optimized heterogeneous structure



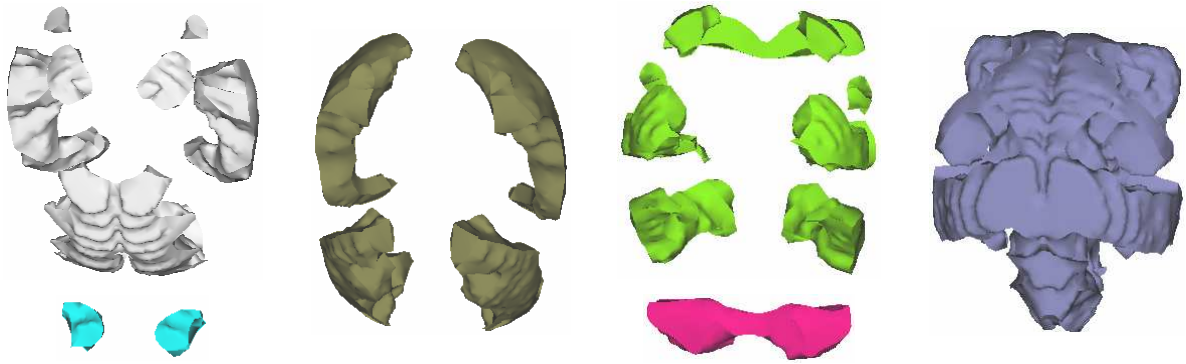
(a) the segmented CT medical image



(b) the extracted surface patches



(c) mesh view



(d) the parts with different tissues

Fig. 14 Example VI: the application of our approach in biomedical engineering to extract mesh surfaces from a segmented CT image data

8. Conclusion and Discussion

In this paper, a novel direct method is presented to extract adaptive and quality surface meshes from implicitly represented heterogeneous volumes made of different materials. Our approach can directly convert implicitly represented heterogeneous objects into a surface model separating homogeneous regions with different material stuffs, where every homogeneous region in a heterogeneous structure is bounded by a set of two-manifold surface meshes. The element size on resultant meshes is adaptive to the curvature of surfaces, and the shape quality of each element is well controlled. The intermediate surfaces between two constituent materials in a heterogeneous structure are directly extracted by our algorithm. Therefore, it is more convenient to adopt the surface meshes generated in our approach for BEM computing or as a starting model to build volumetric meshes for FEM computing. Also, it is easy for our approach to generate models for manufacturing, where two-manifold is always required. In summary, our algorithm shows the following advantages:

- Two-manifold is preserved on each patch of extracted surface meshes;
- The consistency is preserved on the intermediate surfaces of adjacent material regions, which cannot be solved by applying existed isosurface extraction algorithms on each homogeneous material region;
- Comparing to volumetric-element related approaches, our method conducts less computer memory;
- The consistency is also preserved on the shared boundaries of mesh patches by maintaining the linking relationship on boundary common vertices;
- The resultant mesh surfaces are adaptive to curvature and provide good element shape;
- By choosing different cell size and different distance-field sampling size, the mesh patches in different level-of-details can be easily generated.

In summary, this paper presents a novel approach for solving the problem of directly extracting intermediate surfaces from an implicit heterogeneous volume. The algorithm is stable and fast. The results of all examples listed in this paper can be computed in from several seconds to tens seconds on a PC with standard configuration (PIII 1.0GHz CPU + 512MB RAM). Our program is written in ANSI C++ and the GLUT library.

Limitations: The approach presented in this paper shows some limitations in the aspects of normal preservation, models with thin-wall structures, types of heterogeneous volume, and accuracy of distance-field approximation.

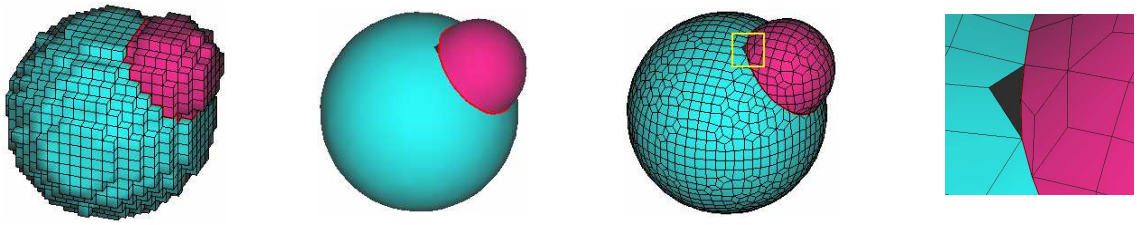
- The normal preservation method presented in this paper actually depends on the approximation of underlying distance-fields. However, as mentioned in [37], the distance-field does not converge to the normal field while increasing the sampling rate. For this reason, the influence on our approach is that the reconstructed surface sharpness may not give sharp enough edges (e.g., on the sharp edges expected to be with $\theta = \frac{\pi}{2}$ dihedral angles, maybe only the edges with $\theta < \frac{\pi}{2}$ dihedral angles are constructed).
- The reconstruction of thin-wall geometry depends on the user specified parameter: Δh. If the thickness of a thin-wall structure is less than Δh, our algorithm will become instable. For another user specified parameter h_c, $h_c > \Delta h$ should be kept; otherwise, the cell-merging algorithm will fail.
- The presented algorithm only works for the heterogeneous volumes which have clear interfaces between different material regions. For those heterogeneous volumes described by continuous

functions, where the stuff on a single point is a blending of several material types, the strategy to separate different regions is still under investigation.

- The mesh quality around boundaries of mesh patches is sometime not high enough – this is mainly because that some of the mesh optimization operators are prevented so that to remain the topological structure. This problem can be solved if adaptive remeshing technique is applied to post-process the output of our approach.
- The signed distance-field adopted in our current implementation is tri-linear, which shows relative low approximation accuracy.

The mesh surfaces constructed above are with triangular elements. However, quadrilateral meshes are widely utilized in FEM for its better numerical quality. The triangular mesh patches generated in our approach of course could take the role of underlying meshes to generate quadrilateral meshes using the Q-Morph approach [56], which is an indirect approach. As an alternative, our algorithm presented above can also be conducted as a direct quadrilateral meshing approach after giving some modification. First of all, when extracting and segmenting coarse mesh patches from the polygonal soup, we keep the faces in the mesh M being quadrilateral. Fig.15a gives an example of the coarse two-manifold quadrilateral meshes segmented from the polygonal soup. Then, in the **Procedure** *Remeshing*, only *Vertex Repositioning* and *Normal Preservation* can be applied on quadrilateral meshes. After iteratively applying these two operations to the coarse two-manifold quadrilateral meshes, the final shape of quadrilateral meshes approximating the interfaces of H is determined. For instance, Fig. 15b and 15c give the quadrilateral mesh representation of heterogeneous object H originally given in Example I (Fig.1a). The overall geometry of M is good; but when looking at the shape of each element, some quadrilateral elements are degenerated (with obtuse inner angle). The degraded elements usually happen near the boundaries of mesh patches (e.g., the black quad given in Fig.15d). In order to solve this problem, further research shall be conducted to develop a quadrilateral version of *Element-Shape Optimization* in **Procedure** *Remeshing*. Our preliminary review finds that the polygonal surface mesh optimization approach in [57] is helpful; since it is not the major concern in this paper, we just leave it for the future investigation.

According to the approximation error shown on the signed distance-field, one possible future work is to adopt non-linear basis functions defined on grid node to give a non-linear approximation of the distance-fields (e.g., the B-spline basis employed in [58]) – so that the approximation error could be decreased without increasing the grid nodes. Other possible future works based on this research include the investigations about how to further enhance the performance of normal preservation and how to extend current in-core algorithm into an algorithm to extract the mesh surfaces in the out-of-core manner, which is very useful when handling the implicit volume sampled on billions of grid nodes.



(a) two-manifold coarse mesh patches **(b) surface after** *Vertex Repositioning* **(c) mesh view** **(d) degenerated quadrilateral element**

Fig. 15 Quadrilateral mesh generated by our approach

9. Reference

[1] Biswas A, Shapiro V, and Tsukanov I. Heterogeneous material modeling with distance fields. *Computer-Aided Geometric Design*, 2004; 21:215-242.

[2] Wang MY, Wang X. A level-set based variational method for design and optimization of heterogeneous objects. *Computer-Aided Design*, 2004; 37:321-337.

[3] Schroeder C, Regli WC, Shokoufandeh A, and Sun W. Computer-aided design of porous artifacts. *Computer-Aided Design*, 2004; 37:339-353.

[4] Adzhiev V, Kartasheva E, Kunii T, Pasko A, and Schmitt B. Cellular-functional modeling of heterogeneous objects. *Proceedings of the seventh ACM symposium on Solid modeling and applications*, Saarbrücken, Germany; 2002, pp.192-203.

[5] Pasko A., Adzhiev V, Schmitt B, and Schlick C. Constructive hypervolume modeling. *Graphical Models*, 2001, 63:413-442.

[6] Kim NH, Choi KK, Chen J-S, and Botkin ME. Meshfree analysis and design sensitivity analysis for shell structures. *International Journal for Numerical Methods in Engineering*, 2002; 53:2087-2116.

[7] Shapiro V, Tsukanov I. Meshfree simulation of deforming domains. *Computer-Aided Design*, 1999; 31: 459-471.

[8] Peraire J, Peiró J, and Morgan K. Advancing front grid generation. *Handbook of grid generation* (edited by Thompson JF, Soni BK, Weatherill NP); CRC Press; 1999, 17-1.

[9] Si H, *TetGen: A Quality Tetrahedral Mesh Generator and Three-Dimensional Delaunay Triangulator*; http://tetgen.berlios.de/.

[10] Cappello F, Mancuso A. A genetic algorithm for combined topology and shape optimisations. *Computer-Aided Design*, 2003; 35:761-769.

[11] Andujar C, Brunet P, Chica A, Navazo I, Rossignac J, and Vinacua A. Optimizing the topological and combinatorial complexity of isosurfaces. *Computer-Aided Design*, 2005; 37:847-857.

[12] Wu Z, Sullivan Jr JM. Multiple material marching cubes algorithm. *International Journal for Numerical Methods in Engineering*, 2003; 58:189-207.

[13] Lorensen W, Cline H. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics*, 1987; 21:163-169.

[14] Duerst MJ. Letters: Additional reference to marching cubes. *Computer Graphics*, 1988; 22:72-73.

[15] Ning P, Bloomenthal J. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 1993; 13:33-41.

[16] Lachaud JO. Topologically defined iso-surfaces. *In Proc. 6th Discrete Geometry for Computer Imagery (DGCI'96)*, Lyon, France, 1996, pp.245-256. Springer-Verlag, Berlin.

[17] Montani C, Scateni R, Scopigno R. A modified look-up table for implicit disambiguation of marching cubes. *The Visual Computer*, 1994; 10:353-355.

[18] Bloomenthal J. An implicit surface polygonizer. In Paul S. Heckbert, editor, *Graphics Gems IV*, 1994, pp.324-349. Academic Press.

[19] Zahlten C. Piecewise linear approximation of isovalued surfaces. In F. H. Post and A. J. S. Hin, editors, *Advances in Scientific Visualization*, 1992, pp.105-118. Springer-Verlag.

[20] Nielson GM, Foley TA, Hamann B, and Lane D. Visualizing and modeling scattered multivariate data. *IEEE Computer Graphics and Applications*, 1991; 11:47-55.

[21] Wallin A. Constructing isosurfaces from CT data. *IEEE Computer Graphics and Applications*, 1991; 11:28-33.

[22] Nielson GM, Hamann B. The asymptotic decider: Resolving the ambiguity in marching cubes. *In Proc. of IEEE Visualization 1991*, 1991, pp.83-91.

[23] Wilhelms J, Van Gelder A. Topological considerations in isosurface generation. *Computer Graphics*, 1990; 24:79-86.

[24] Wyvill G, McPheeters C, and Wyvill B. Data structures for soft objects. *The Visual Computer*, 1986; 2:227-234.

[25] Lewiner T, Lopes H, Vieira AW, and Tavares G. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of Graphics Tools*, 2003; 8:1-15.

[26] Weber GH, Scheuermann G, Hagen H, Hamann B. Exploring scalar fields using critical isovalues. *In Proc. of IEEE Visualization 2002*, 2002, pp. 171-178.

[27] Stander BT, Hart JC. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. *In Proceedings of SIGGRAPH 97*, pp.279-286, 1997.

[28] Nielson G. On marching cubes. *IEEE Transactions on Visualization and Computer Graphics*, 2003; 9:283-297.

[29] Cignoni P, Ganovelli F, Montani C, and Scopigno R. Reconstruction of topologically correct and adaptive trilinear isosurfaces. *Computers and Graphics*, 2000; 24:399-418.

[30] Schroeder W, Zarge J, and Lorensen W. Decimation of triangle meshes. *Computer Graphics*, 1992; 26:65-70.

[31] Hoppe H, DeRose T, Duchamp T, McDonald J, and Stuetzle W. Mesh optimization. *In Proc. of SIGGRAPH 1993*, 1993, pp.19-26.

[32] Kalvin A, Taylor R. Superfaces: polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Applications*, 1996; 16:64-77.

[33] Turk G. Re-tiling polygonal surfaces. *Computer Graphics*, 1992; 26:55-64.

[34] Reitinger B, Bornik A, and Beichel R. Constructing Smooth Non-Manifold Meshes of Multi-Labeled Volumetric Datasets. *In Proc. of WSCG'2005*, 2005, pp.227-234,

[35] Fujimori T, Suzuki H. Surface extraction from multi-material CT. *In Proc. of Ninth International Conference on Computer Aided Design and Computer Graphics*, 2005.

[36] Frisken S, Perry R, Rockwood A, and Jones R. Adaptively sampled distance field: a general representation of shapes for computer graphics. *In Proc. of ACM SIGGRAPH 2000*, 2000, pp.249-254.

[37] Kobbelt L, Botsch M, Schwanecke U, and Seidel HP. Feature-sensitive surface extraction from volume data. *In Proc. of ACM SIGGRAPH 2001*, 2001, pp.57-66.

[38] Ju T, Losasso F, Schaefer S, and Warren J. Dual contouring of hermite data. *In Proc. of ACM SIGGRAPH 2002*, 2002, pp.339-346.

[39] Varadhan G, Krishnan S, Kim YJ, and Manocha D. Feature-sensitive subdivision and isosurface reconstruction. *In Proc. of IEEE Visualization 2003*, 2003, pp.99-106.

[40] Ohtake Y, Belyaev A. Mesh optimization for polygonized isosurfaces. *Computer Graphics Forum*, 2001; 20:368-376.

[41] Ohtake Y, Belyaev A, and Pasko A. Dynamic mesh optimization for polygonized implicit surfaces with sharp features. *The Visual Computer*, 2003; 19:115-126.

[42] Kartasheva E, Adzhiev V, Pasko A, Fryazinov O, and Gasilov V. Surface and volume discretization of functionally based heterogeneous objects, *Journal of Computing and Information Science and Engineering*, *ASME Transactions*, 2003; 3:285-294.

[43] Crossno P, Angel E. Isosurface extraction using particle systems. *In Proc. of IEEE Visualization 97*, 1997, pp.495-498.

[44] Gibson S. Using distance maps for smooth surface representation in sampled volumes. *In Proc. of 1998 IEEE Volume Visualization Symposium*, 1998, pp.23-30.

[45] Perry RN, Frisken SF. Kizamu: a system for sculpting digital characters. *In Proc. of ACM SIGGRAPH 2001*, 2001, pp.47-56.

[46] Jin X, Sun H, and Peng Q. Subdivision Interpolating Implicit Surfaces. *Computers & Graphics*, 2003; 27:763-772.

[47] van Overveld K, Wyvill B. Shrinkwrap: an efficient adaptive algorithm for triangulating an iso-surface. *The Visual Computer*, 2004; 20:362-379.

[48] Hilton A, Stoddart AJ, Illingworth J, and Windeatt T. Marching triangles: range image fusion for complex object modelling. *Proceedings of 3rd IEEE International Conference on Image Processing*, 1996, vol.2, pp.381-384.

[49] Akkouche S, Galin E. Adaptive implicit surface polygonization using marching triangles. *Computer Graphic Forum*, 2001; 20:67-80.

[50] Mortenson ME. *Geometric modeling*, 1985, New York: Wiley.

[51] Jones MW, Satherley RA. Shape representation using space filled sub-voxel distance fields. *Proceedings International Conference on Shape Modeling and Applications 2001*, 2001, pp.316-325.

[52] Mauch S, Breen D. A fast marching method of computing closest points. (http://www.cco.caltech.edu/~sean/closestpoint/closept.html)

[53] Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W. Mesh optimization. *Extended_TR_UW_CSE_1993-01-01* (http://www.research.microsoft.com/~hhoppe).

[54] Meyer M, Desbrun M, Schröder P, and Barr AH. Discrete differential-geometry operators for triangulated 2-manifolds. *VisMath '02*, Berlin (Germany), 2002.

[55] Sun W, Jiang T., and Lin F. A processing algorithm for freeform fabrication of heterogeneous structures. *Rapid Prototyping Journal*, 2004; 10:316-326.

[56] Owen SJ, Staten ML, Canann SA, and Saigal S. Q-morph: an indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering*, 1999; 44:1371-1340.

[57] Garimella RV, Shashkov MJ. Polygonal surface mesh optimization. *Engineering with Computers*, 2004; 20:265-272.

[58] Freytag M, Shapiro V, and Tsukanov I. Field modeling with sampled distances. *Computer-Aided Design*, 2005; in press.

Appendix Topology Degeneration in Element-Shape Optimization

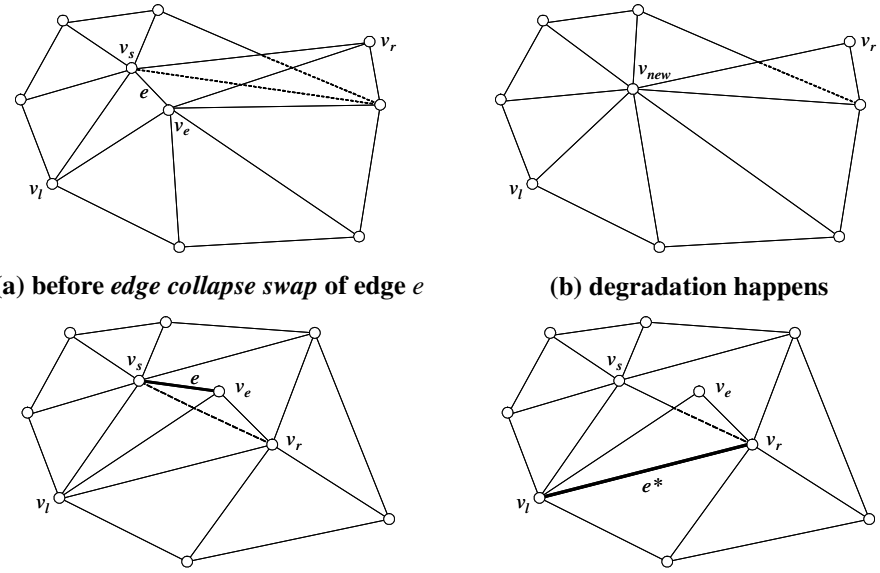
The figures below show the cases which will lead to invalid topology. In detail, for *edge collapse*, if

$$star(v_s) \bigcap star(v_r) = \{v_l, v_r, v_i, \cdots\} \supset \{v_l, v_r\}$$

but not

$$star(v_s) \bigcap star(v_r) = \{v_l, v_r\},$$

the degeneration happens after collapsing $e = \langle v_s, v_r \rangle \in \mathrm{M}$; for *edge swap*, if there exist an edge $\langle v_l, v_r \rangle \in \mathrm{M}$, invalid topology will occur after swapping *e*.



(a) before *edge collapse swap* of edge *e* **(b) degradation happens**

(c) before *edge swap* of edge *e* **(d) degradation happens**

Fig. A Mesh degradation cases should be prevented in *edge collapse* and *edge swap*