Quality Partitioned Meshing of Multi-Material Objects
Chandrajit Bajaj\textsuperscript{a,}\textsuperscript{*}, Qin Zhang\textsuperscript{a}
\textsuperscript{a}Institute of Computational Engineering and Sciences, Computational Visualization Center, University of Texas, Austin, TX 78712

Abstract
We present a simple but effective algorithm for generating topologically and geometrically consistent quality triangular surface meshing of compactly packed multiple heterogeneous domains in $\mathbb{R}^3$. By compact packings we imply that adjacent homogeneous domains or materials share some 0, 1, and/or 2 dimensional boundary. Such packed multiple material (or multi-material) solids arise naturally from classification/partitioning/segmentation of homogeneous domains in $\mathbb{R}^3$ into different sub-regions. The multi-materials may also represent separate functionally classified sections or just be multiple component copies tightly fused together as perhaps by layered manufacturing processes. The input to our algorithm is a geometric representation of the entire multi-material solid, and a volumetric classification map identifying the individual materials. As output, each individual material region is represented by a triangulated 2-manifold boundary, with adjacent material regions having shared boundaries. Our algorithm has been implemented, and applied to different multi-material solids, and the results are additionally presented with quantitative analysis of detection and curation of non-manifold interfaces as well as spurious small components. These meshes are useful for combined boundary element analysis, however these simulation results are not presented.

1. Introduction
Heterogeneous domains are at times referred to as solids made of different constituent materials (aka multi-material objects). When the solid is made of continuously varying material compositions, it is classified as functionally gradient materials (FGM) (see e.g., [1,2]), and usually exhibits multiple and mixed mechanical, electrical, acoustic properties. A discontinuous change in material compositions however generates heterogeneous regions of distinct material types in the solid, and are often classed as a multi-material object (MMO). In the review paper [3], Kou and Tan list a few typical MMOs: wear resistant coatings, solid oxide fuel cells, dental implants, bone implants, and so on. Elucidating properties of such MMO’s through simulations motivates our problem of generating quality meshing of compactly packed multiple heterogeneous domains.

More formally: Given any geometric representation of the MMO, and additionally a 3-dimensional (3D) material classification map identifying the individual materials, we generate a quality triangular manifold surface mesh boundary representation for each individual material region. If two different material regions are adjacent, their...
corresponding meshes would share some 0-dim, 1-dim and/or 2-dim boundary. Each triangular mesh would be a 2-dimensional manifold, aka a 2-manifold. (see Definition 4.1). The 3D material classification map, as is the case when MMO’s are obtained from imaging data, while providing a partition of the entire MMO domain based on the classification may suffer from non-manifold vertices (0D) and edges (1D) [3–5]. Our additional contributions, thus include a scheme to detect and curate all potential non-manifold cases from the classification map, including removal of small isolated components. Finally, with the use of constrained geometric flow optimization we provide quality multi-material meshes.

The rest of the paper is organized as follows. In Section 2, we briefly review related prior work. In Section 3, we briefly formalize the sub-problems and give a sketch of the algorithm steps. Specific detail of each step are given in Section 4. Example results of our implementation are given in Section 5, including summary quantization of non-manifold detection, with mesh curation. These demonstrate the efficacy of our solution.

2. Related Prior Work

We focus our review here on only the geometric meshing aspects of heterogeneous domains and skip the material modeling that have also been reviewed in [3]. The prior multi-material meshing methods include voxel-based [6–8], finite element-based [5,9,10], mesh feature based [11,12], and layer-based [13,14] techniques. A systematic approach to multi-material solid modeling was proposed in [15] based on rm sets and rm classes. A 3D solid is subdivided into components made of unique materials and a non-manifold Boundary representation (B-rep) [16,17] is used to model such objects. Each component is homogeneous and has an assigned index of material. Chiu and Tan [18] developed a material tree structure to store different composition of an object. The material tree were then added to a data file to construct a modified STL file format suitable for rapid prototyping.

For the single material object, isocontouring is usually used to extract the mesh representation for efficient visualization and computation. The most common isocontouring scheme is marching cube method proposed in [19]. Later, this single-material contouring scheme was expanded to multi-materials case by Wu and Sullivan in [20], named as the M3C algorithm. After a smoothing step, the extracted multi-material meshes look nice but volume shrinkage issues inherent in their method cannot be usually ignored. Also gaps exist between the different materials. In [21], Wang extracted surface meshes from multi-material volumes using three steps: a combined coarse patche extraction, signed distance fields construction and an adaptive remeshing process. It is reported in the paper that the extracted meshes are 2-manifold with high quality. Wang and Feng [22] implement the construction a boundary surfaces of homogeneous objects (BSHO) method to extract multi-material meshes on GPU’s. Shammal et al. [23] combined region growing and graph-cut methods to classify the volumetric model into its component material domains and adopted a generalized marching cube method to generate triangulated mesh surfaces. In [4], an efficient pipeline is proposed to generate multi-domain quality meshes from volumetric data. The fundamental difference between the current paper and what was reported in [4] is that there are gaps between the independent and multiple material domain meshes generated by their method. The multi-material meshes of this paper, if they are adjacent to each other, always share some common boundary. Furthermore, the meshes we generate are also guaranteed to be 2-manifolds, and don’t possess spurious and miniscule material components.

3. Notation, Problem Statement and Algorithm Sketch

Notes. A grid $G$ is composed of grid point as $G = \{G_{ijk} = (x_i, y_j, z_k); 0 \leq i < G_x, 0 \leq j < G_y, 0 \leq k < G_z\}$, where $(G_x, G_y, G_z)$ is the grid dimension. Without loss of generality, we assume $G_x = G_y = G_z$ and $d_i = x_i - x_{i-1}$ is identical to $d_j = y_j - y_{j-1}$ and $d_k = z_k - z_{k-1}$ for all $(i, j, k) \in (0, G_x) \times (0, G_y) \times (0, G_z)$. The multi-material object is represented by a volumetric representation or a classification map $\mathcal{V}$ as

$$f : G \subset \mathbb{R}^3 \rightarrow \mathbb{N}$$

$$(x_i, y_j, z_k) \rightarrow f(x_i, y_j, z_k) = f_{ijk}, \quad i, j, k = 0, \ldots, G_x - 1,$$

where $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of natural numbers with 0 representing the background. If the object is made of $N$ materials, then $0 \leq f_{ijk} \leq N, \forall (i, j, k) \in [0, G_x - 1]^3$ and $f_{ijk}$ is the material ID of grid point $G_{ijk}$. For grid point $G_{ijk}$,
Fig. 1. Multi-material mesh of the Brodmann classified regions of the human cortex. (a) region based colored rendering of the Input voxel resolution 3D classification map (b) Shaded rendering of the final multiple material mesh. (c) Multiple material mesh representation with highlighted boundary vertices and curvilinear boundary edges. (d) Boundary graph with vertex(0-dim), curvilinear edge (1-dim) (e) Shaded rendering with wireframe of the final multiple material mesh. (f) Close in zoomed view of the final multiple material mesh as shown in (e).

we denote by \( V_{ijk} \) the dual voxel or voxel, that is \( V_{ijk} = [x_i - \frac{1}{2}d_x, x_i + \frac{1}{2}d_x] \times [y_j - \frac{1}{2}d_y, y_j + \frac{1}{2}d_y] \times [z_k - \frac{1}{2}d_z, z_k + \frac{1}{2}d_z] \). One voxel consists of six faces \( f \), twelve edges \( e \), eight vertices \( v \), and the space enclosed by the six faces. Obviously, the center of the voxel \( V_{ijk} \) is the grid point \( G_{ijk} \).

If \( f_{ijk} = f_{(i+1)jk} \), we say that voxels \( V_{ijk} \) and \( V_{(i+1)jk} \) share a face \( f \). If \( f_{ijk} = f_{(i+1)(j+1)k} \) and \( V_{(i+1)(j+1)k} \) share an edge \( e \). If \( f_{ijk} = f_{(i+1)(j+1)(k+1)} \), we say that voxels \( V_{ijk} \) and \( V_{(i+1)(j+1)(k+1)} \) share a vertex \( v \). Similar definitions apply to the other allowable cases. Thus, one voxel \( V_{ijk} \) has six faces, twelve edges and eight vertices that could be shared with other voxels.

For the classification map \( V \), we can generate a partition graph \( P^G(V) \), which is the boundary representation of the multi-material object. A partition graph \( P^G \) is composed of partition face set \( P^f \), partition edge set \( P^e \), and partition vertex set \( P^v \), that is, \( P^G(V) = P^f(V) \cup P^e(V) \cup P^v(V) \). The partition face set \( P^f(V) \) is defined as the union of faces with each face shared by two voxels having different material IDs, that is,

\[
P^f(V) = \bigcup_{i,j,k,l,m,n=0,\ldots,G_x-1} \{ f \in V_{ijk} \text{ and } f_{ijk} \neq f_{lmn}, |l - i| + |m - j| + |n - k| = 1 \}.
\]

Similarly, the partition edge set \( P^e(V) \) is defined as the union of edges with each edge shared by two voxels having different material IDs, that is,

\[
P^e(V) = \bigcup_{0 \leq |l - i|, |m - j|, |n - k| \leq 1, i,j,k,l,m,n=0,\ldots,G_x-1} \{ e \in V_{ijk} \text{ and } f_{ijk} \neq f_{lmn}, |l - i| + |m - j| + |n - k| = 2 \}.
\]

The partition vertex set \( P^v(V) \) is defined as the union of vertices with each vertex shared by two voxels having different material IDs, that is,

\[
P^v(V) = \bigcup_{i,j,k,l,m,n=0,\ldots,G_x-1} \{ v \in V_{ijk} \text{ and } f_{ijk} \neq f_{lmn}, |l - i| \neq |m - j| \neq |n - k| = 1 \}.
\]
Then a partition mesh \( P_t \) for a particular material ID \( t \) is part of the partition graph \( P^G(V) \), and is defined by
\[
P_t(V) = \{ f \mid f \in V_{ijk} \cap P^d(V), f_{ijk} = t, i, j, k, = 0, \ldots, G_x - 1 \}.
\]

See for example Fig ??.

**Problem statement.** Our goal is to generate quality surface meshes \( M_t(i = 1, \ldots, N) \), representing the \( N \) materials and satisfying the following two criteria:

1. Each mesh \( M_t(i = 1, \ldots, N) \) is a 2-manifold.
2. (a) If \( f_{ijk} \neq f_{(i+1)jk} \), then \( H^2(V_{ijk} \cap V_{(i+1)jk}) > 0 \) and \( H^2(M_{f_{ijk}} \cap M_{f_{(i+1)jk}}) > 0 \).
   (b) If \( f_{ijk} \neq f_{(i+1)(j+1)k} \), then \( H^1(V_{ijk} \cap V_{(i+1)jk}) > 0 \) and \( H^1(M_{f_{ijk}} \cap M_{f_{(i+1)jk}}) > 0 \).
   (c) If \( f_{ijk} \neq f_{(i+1)(j+1)(k+1)} \), then \( H^0(V_{ijk} \cap V_{(i+1)jk}) > 0 \) and \( H^0(M_{f_{ijk}} \cap M_{f_{(i+1)jk}}) > 0 \).

Here \( H^d(A) \) is the \( d \)-dimensional Hausdorff measure \([24]\) of set \( A \), which is intuitively the generalization of number of points in a finite set (\( d = 0 \)), the length of a curve (\( d = 1 \)), the area of a surface (\( d = 2 \)), the volume of a solid object (\( d = 3 \)), etc. We can describe Criterion 2 succinctly as the collection meshes which are generated are required to keep the partition topology as specified by the input voxel resolution classification map. That means, if two differently classified voxels share a face, the two different material meshes, and individually containing their respective voxels, share a 2-dim face. If the original two voxels share an edge, the two material meshes also share a 1-dim edge. If the original two voxels share a vertex, the two generated meshes also share a 0-dim vertex. Furthermore, we can simplify this to say that if two grid points are adjacent in the classification map having different material IDs, the final meshes are gaps-free, and share a common boundary (a face, an edge, or a vertex). Given the grid-based representation of the object, we know that one face can be shared at most by two different material meshes, one edge can be shared at most by four different material meshes and one vertex can be shared at most by eight different material meshes.

**Sketch of multi-material meshing algorithm.** The input 3D multi-material classification map is often obtained by multi-domain segmentation or classifiers using normalized graph cut operating on a voxelized 3D reconstructed image of the MMO \([25,26]\). Software such as AsymSeg and SymSeg \([27,28]\) implemented and publicly available from VolRover \([29]\) and Segger \([30]\) can be used to generate such material classification maps. Generally, if a face is shared by two voxels with the same material ID \( i \), then the face will be removed and the two voxels are clustered into the same \( i \) material region. Otherwise, the face will be part of the partition mesh \( P_t \) and partitioned into two different material regions. Unfortunately, the partition meshes \( \{P^d_{\text{face}}\} \) generated by this simple way usually suffer from (a) non-manifold vertices (0D) and non-manifold edges (1D) as well as (b) isolated and extremely tiny independent material regions. Our MMO meshing algorithm includes as a first step a detection and curation of such classification artifacts, for if ignored, the resulting meshes do not conform to meshes required by standard boundary element simulations. Our algorithm sketch is as follows. (See also Fig ??).

1. The partition mesh \( P_t(t = 1, \ldots, N) \) may include many small single material voxel clusters. We call this the tiny component sub-problem, and detect and curate the \( P_t(t = 1, \ldots, N) \) to remove these.
2. The partition graph which is the union of partition meshes \( P_t(t = 1, \ldots, N) \) may not be 2-manifold at the interfaces between different material. We call this problem the non-manifold sub-problem, and detect and curate the \( P_t(t = 1, \ldots, N) \) to remove non-manifold vertices and edges.
3. Generate and curate a partitioned graph \( P^G \) from the given voxel classification map. For a classified material ID \( t(t = 1, \ldots, N) \), each partition mesh \( P_t \) of \( P^G \) is the boundary of the union of voxels whose centers have material ID \( t \). The topology of the partition graph \( P^G \) captures the topological adjacency or arrangement of multiple material region, and induces a similar partitioning of the surface domain geometry.
4. Our final step is to dimensionally refine the geometry of our partition graph \( P^G \) by first smoothing all curvilinear 1D edges, and then smoothing and improving the mesh quality of the interior 2D faces, the surface patch regions of individual materials. We use a geometric flow smoothing scheme, which preserves the integral length of the edges and surface area of the faces, respectively, while maintaining the partition graph \( P^G \) topology and geometry (i.e. no intersections).
Each of these steps are detailed in sub-sections 4.1, 4.2, 4.3 and 4.4, respectively. When we process our input solid domain, we perform steps 1 and 2 repeatedly and in that order, i.e we first fix the tiny component problem, and then fix the non-manifold problem. If the non-manifold fix step causes tiny component problems, we repeat the steps as necessary. While we don’t have a formal proof of why this process converges, we succeeded in removing all the artifacts in all realistic data examples we tried. See for example our reported statistics in Table 1, where all non-manifold vertices and edges were eliminated, starting from an initial number of 2867.

4. Algorithm and Implementation Details with Results

4.1. Fixing the non-manifold problem

For multi-material surface meshing, Wang and Feng [22] devised a complicated vertex classification and merging lookup table to preserve the 2-manifold feature, but the resulting mesh quality was not very high since 2-manifold feature is not easy to be preserved when meshes are smoothed. Wang [21] pays attributes to the regularization theory [17] for generating 2-manifold model from non-manifold objects. In [31], several approaches, such as blending and chamfering, sweeping, joining and splitting along edges, planar sectioning, are provided to handle non-manifold modeling. However, these approaches are not suitable for our multi-material cases since these approaches are suitable for only one material [32].

Definition 4.1. A surface mesh is a 2-manifold if the local neighborhood of every point on the mesh is topologically equivalent or homeomorphic to a disk.

Intuitively, it means the local neighborhood of every point may be continuously transformed into a disk by stretching and bending, but without cutting, tearing or gluing. For example, in Fig. 2, vertex $P$ is a manifold vertex but vertex $Q$ is a non-manifold vertex. Edge $L$ is a non-manifold edge. For a mesh with non-manifold vertices or edges, it is not trivial to fix them. There are several approaches to fixing the non-manifold problem, such as cutting and stitching [33] and simulated annealing [32]. In this paper, we do not fix the non-manifold problem on the surface mesh, but instead directly fix the 3D voxel classification map and then use a single dimensionality traversal of the graph to preserve the non-manifold feature.

We handle the non-manifold vertex and non-manifold edge problems separately. For non-manifold vertex case, we detect all the possible cases as shown in Figure 3. Assume the left-bottom grid point is $ijk$. Generally, for each grid point $ijk$, there are 8 neighbors of $lmn$. Here we show the case $l = i + 1, m = j + 1, n = k + 1$. Basically we consider two cases: the voxel $V_{ijk}$ is background (see figure 3 (c) and (d)) or not background (see figures 3 (a) and (b)). In this figure, the blue, green and orange points are grid points with non-zero material IDs. The red vertex is a non-manifold vertex on the partition mesh.
In Algorithm 1, we give the detection and curation method details, where the information after « are comments. The \( \text{Tag}(i) \) function is defined to remember whether the voxel \( i \) has been processed or not and it will be also used for potential edge non-manifold detection in Algorithm 2. In Figure 4, we show the detection of Case 1 non-manifold vertex and the curation method. The numbers 1,...,6 representing the six voxels with 1, 2, 3 voxels on the bottom layer and 4, 5, 6 voxels on the top layer are shown in Figure (b). After detection and curation, Figure (c) shows the result and Figure (d) shows the neighborhood status. In slightly more detail, we first find that voxel \( i \) is not a background voxel (line 2 in Algorithm 1) and \( f_{i,j,k} = f_{l,m,n} \) (line 3 in Algorithm 1). Then we check the six neighbors of this voxel \( i \) as shown in Figure 4(b), if none of them have the same material ID with voxel \( i \) (line 4 in Algorithm 1) and none of them are background (line 5 in Algorithm 1), then we replace the material ID of that voxel with the material ID of voxel \( i \) (line 6 in Algorithm 1). If all of them have material IDs other than background, then we randomly change one voxel with the material ID of voxel \( i \) (line 8 in Algorithm 1).

Lines 13-31 in Algorithm 1 detect and curate Case 2 non-manifold vertices. Lines 33-43, 44-52 in Algorithm 1 detect and curate Case 3 and Case 4 non-manifold vertices, respectively.

In Figure 5, we show two examples of non-manifold vertices detection and curate for the Brodmann area data (region 16, see Subsection 5.1), where (a) and (c) show two non-manifold vertices in the red circle. Figures (b) and (d) show the curate results of (a) and (c) respectively.

Comparing to the non-manifold vertex case, the detection and curation of non-manifold edges are simpler. In Figure 6, we show the scheme of detecting and curating a potential non-manifold edge. In figure (a), the red edge is detected to generate potential non-manifold edges and we curate it as shown in figure (b). Algorithm 2 gives details.

4.2. Fixing the tiny-region problem

A straightforward though cumbersome way to remove the tiny regions of the multi-material partitioned meshes, are to collapse them to a neighboring material region, after measuring their respective sizes (usually volume or \( H^3 \)-measure) relative to the desired partition mesh. Here, as above, we propose a method that operates directly on the classification map to very efficiently remove all isolated tiny material regions. We call our approach as voxel cluster growing (VCG). Based on the continuity of the material ID (including background) of each grid point, we classify them into multiple regions. If the region size is less than a threshold, then we change the material ID of this region.
with its neighbor. This is done voxel by voxel in this region. For example, for each voxel, we count the duplications of the material ID of its 6 neighbor voxels. We change the material ID of the voxel to the material ID with biggest duplication. This process is iteratively carried out, which means the region size is changed gradually. We utilize a geometric series, that is, the region size threshold is set to be $1, 2, 4, 8, \ldots$ in sequence. During all our experiments, we find that our VCG method works very effectively in removing tiny individual regions, without leaving any holes or voids.

In Figure 7, we show one example for the Brodmann area data (region 16, see Subsection 5.1). In figure (a), we show the cutview of the partitioned mesh, where one can see there are multiple tiny regions, all separately classified. In figure (b), we show the cutview after our tiny regions removal (curation) process. Note not only are there no tiny regions, the artifacts caused by the density map segmentation have been additionally eliminated. Figure (c) show the cutview of the result.

![Fig. 5. Non-manifold vertices detection and curate of real data. (a) and (c) The vertices in the red circle are non-manifold vertices. (b) and (d) The vertices in the red circle are manifold vertices after curation.](image)

![Fig. 6. Non-manifold edges detection and cure. (a) The red edge is a non-manifold edge. (b) The voxel neighbors of the red edge. (c) After cure, the red edge is a manifold edge. (d) Neighborhood status after curation. The two blue points are two grid points.](image)

![Fig. 7. Tiny components detection and removal (curation). (a) The red small regions are detected. (b) Cutview after tiny regions removal. (c) The resulting mesh after the final smoothing and quality improvement step.](image)
4.3. Partitioned mesh generation

After fixing all the non-manifold and tiny-region problems, we can generate the boundary representation of each material by partitioning the classification map. For a grid point $i \times j \times k$ with material ID $f_{i \times j \times k}$, we produce a dual cube with six faces with face ID $f_{i \times j \times k}$. If two dual cubes have common faces with the same face ID, say $f_{i \times j \times k}$, then we combine these two cubes into one bigger cuboid, and so on. If two dual cubes have common faces with different face IDs, we then produce different surfaces. We call these surfaces indented partitioned meshes or just partitioned meshes. Now each partitioned mesh is topologically a 2-dimensional manifold. One should be reminded that each partitioned mesh may include several components. See Figures 5 (b) and (d), Figure 7 (b) for some examples of partition meshes.

4.4. Constraint geometric flow smoothing of partition meshes

Our next step is to improve the mesh quality of the partitioned meshes while keeping the correct topology of meshes. We adapt a strategy called constrained geometric flow smoothing (CGFS) by invoking a dimensionality based traversal of our partition mesh graph. The basic idea is to reposition the vertices of individual meshes according to their connectivity property. We first classify the vertices into three classes:

- single-material vertex: the vertex only belongs to one material region.
- double-material vertex: the vertex is only shared by two materials.
- triple-over-material vertex: the vertex is shared by three or more materials.

Our smoothing strategy is accordingly:

- For single-material vertex, we adapt surface diffusion flow with tangential regularization to smooth it.
- For double-material vertex, we need to further classify them into two cases. Usually two adjacent materials share one common surface patch, which is a two-dimensional manifold with boundary. Then we classify double-material vertices into two categories:
  - Boundary vertices: the vertices which are on the boundary of the two-dimensional manifold patch. These vertices construct a curve without endpoints. We smooth this indent curve by a simple way, that is by taking a weighted average of three vertices. The weight for the central vertex we adopted is 0.6 and the weight for the other two vertices are 0.2. This class of vertices is processed prior to the interior vertices.
  - Interior vertices: the vertices which are not on the boundary of the two-dimensional manifold patch. For this class of vertices, we apply the same smoothing method as for single-material vertex.
- Triple-over-material vertices are fixed without repositioning.

In the CGFS strategy, the geometric flow adopted is the surface diffusion flow with tangential regularization [4,34], that is

$$\frac{\partial x}{\partial t} = \Delta H n + \nu t,$$

(1)

where $n$, $t$ are the normal and tangent vectors to the vertex $x$, respectively. $H$ is the mean curvature, $\Delta$ is the Laplace-Beltrami operator, $\nu$ is the velocity in the tangent direction and $t$ is a marching parameter. We solve this geometric partial differential equation (1) over a triangular mesh in a discrete scheme. In the temporal dimension, we use a forward Euler scheme. In spatial dimension, each term is discretized, including mean curvature $H$ and the Laplace-Beltrami operator [35,36]. This geometric flow is volume-preserving, that means, the volume enclosed by the surface will be invariant during evolution. Even though many other schemes for smoothing exist, such as Laplacian smoothing [37–40], the surface diffusion flow with tangential regularization works best.

5. Experimental Results

The pipeline proposed in this paper has been implemented and tested on several examples.
5.1. Brodmann areas

A Brodmann area is a region of the cerebral cortex defined on its cytoarchitectonics, or structure and organization of cells ([41], English version translated by Garey [42]). It’s about 50 areas for human and non-human primates. Our data includes 41 regions. In [4], we have previously build multi-material meshes with gaps between each material region. In Figure 1 and Figure 8, we show two different views of the boundary representation of the Brodmann areas.

![Fig. 8. Brodmann areas of brain. (a) Original partition map. (b) Solid rendering of the multiple material mesh. (c) Multiple material mesh representation with boundary vertices. (d) Boundary vertices. (e) Solid rendering with wireframe of the multiple material mesh. (f) Zoom in view of a portion of meshes in (e).](image)

We list a number of tiny regions (components) and non-manifold vertices and edges number in Table 1. In the first iteration, we needed 13 loops to make the size of each component over a threshold and then we needed 5 loops to curate the non-manifold vertices and edges. Note the random selection in our curation algorithm. In the second iteration, we needed 13 loops to make the size of each component to be under the same threshold as with 70 components (earlier). No non-manifold vertices and edges were detected, i.e. they had all been curated.

We also computed the area and volume for each area and the total volume to show volumetric preservation properties of our constrained geometric flow optimization. The statistics are reported in Table 2. The normal volume of a human brain is $1300 - 1500 \text{cm}^3$. 
Table 1. Brodman areas tiny component fixing and non-manifold fixing statistics.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Loop</th>
<th># components</th>
<th>Loop</th>
<th># non-manifold vertices and edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. The volumes of individual Brodman areas. Unit mm^3.

<table>
<thead>
<tr>
<th>area</th>
<th>volume</th>
<th>area</th>
<th>volume</th>
<th>area</th>
<th>volume</th>
<th>area</th>
<th>volume</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24993.190980</td>
<td>12</td>
<td>83306.690610</td>
<td>22</td>
<td>18922.652301</td>
<td>32</td>
<td>4626.289581</td>
</tr>
<tr>
<td>2</td>
<td>14238.734365</td>
<td>13</td>
<td>18462.201872</td>
<td>23</td>
<td>37193.895606</td>
<td>33</td>
<td>50042.611639</td>
</tr>
<tr>
<td>3</td>
<td>19529.177004</td>
<td>14</td>
<td>1186.165355</td>
<td>24</td>
<td>34544.761322</td>
<td>34</td>
<td>46977.748187</td>
</tr>
<tr>
<td>4</td>
<td>77075.120754</td>
<td>15</td>
<td>66535.767843</td>
<td>25</td>
<td>15218.418055</td>
<td>35</td>
<td>30659.084403</td>
</tr>
<tr>
<td>5</td>
<td>161986.839854</td>
<td>16</td>
<td>6788.072364</td>
<td>26</td>
<td>27051.367103</td>
<td>36</td>
<td>4759.113220</td>
</tr>
<tr>
<td>6</td>
<td>98607.168977</td>
<td>17</td>
<td>25681.920613</td>
<td>27</td>
<td>33032.876507</td>
<td>37</td>
<td>22497.331052</td>
</tr>
<tr>
<td>7</td>
<td>13010.120724</td>
<td>18</td>
<td>1668.009589</td>
<td>28</td>
<td>28609.124907</td>
<td>38</td>
<td>32466.247486</td>
</tr>
<tr>
<td>8</td>
<td>32168.631313</td>
<td>19</td>
<td>10311.779490</td>
<td>29</td>
<td>2402.269414</td>
<td>39</td>
<td>27998.900245</td>
</tr>
<tr>
<td>9</td>
<td>81971.079039</td>
<td>20</td>
<td>7579.996076</td>
<td>30</td>
<td>5720.112915</td>
<td>40</td>
<td>36009.643504</td>
</tr>
<tr>
<td>10</td>
<td>15881.479013</td>
<td>21</td>
<td>28036.965678</td>
<td>31</td>
<td>5239.390048</td>
<td>41</td>
<td>6506.035475</td>
</tr>
<tr>
<td>11</td>
<td>90137.361823</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1349634.346</td>
</tr>
</tbody>
</table>

5.2. Penicillium stoloniferum virus slow cryo-EM data

We selected an example of the Penicillium stoloniferum virus slow (PsV-S) at 7.3 AngstromA resolution cryo-EM data (EMDB ID:1459) [43] as the second example. The virus capsids are a closed packed shell of individual proteins in a spherical arrangement. The virus capsids houses the viral genome (RNA/DNA) .. In Fig. 9, we show the original density map in (a) by shaded rendering by VolRover (http://cvcweb.ices.utexas.edu/cvcwp/?page_id=100). Figures (b) and (c) show the multi- material meshing results of the 60 monomers without boundary
points and with boundary points between two adjacent materials, respectively. Figure (d) shows the cut view of the capsid. Figure (e) shows the mesh quality and (f) shows one monomer protein mesh with boundary points.

Fig. 9. PsV-S cryo-EM data at 7.3 Angstrom resolution. (a) Original volumetric density map by shaded rendering. (b) Multiple material gap-free meshing of each monomer (60 monomers). (c) Multiple material mesh representation with boundary vertices. (d) Cut view. (e) Zoomed in for mesh quality visualization. (f) One monomer protein mesh with boundary points.

5.3. P22 data

This example shows the multiple material gap-free meshing of bacteriophage P22 at 9.5 Angstrom resolution [44] in Fig. 10. A bacteriophage is a virus that infects bacteria. Figure (a) shows the original volumetric density map by shaded rendering. Figure (b) shows the multi-material mesh result. The virus capsid is composed of 60 hexamer and 12 pentamer arrangements of proteins. Figure (c) shows the mesh representation with boundary vertices. Figures (d) and (e) show the cut view of the mesh without/with boundary vertices. Figure (f) shows three hexamers with all boundary vertices.

6. Conclusions

We present a simple but effective algorithm for generating topologically and geometrically consistent quality triangular surface meshing of compactly packed multi-material domains. Each individual material region is represented by a triangulated 2-manifold boundary mesh, with adjacent material regions having shared boundaries. Our algorithm has been implemented, and applied to different multi-material solids, and the results are additionally presented with quantitative analysis of detection and curation of non-manifold interfaces as well as spurious small components. This MMO meshing is in the process of being parallelized on multi-core and many-core computers, and furthermore, coupled to
Fig. 10. Bacteriophage P22 cryo-EM data at 9.5 Angstrom resolution. (a) Original volumetric density map by shaded rendering. (b) Multiple material gap-free meshing of 60 hexamers and 12 pentamers. (c) Multiple material mesh representation with boundary vertices. (d) Cut view. (e) Cut view with boundary points. (f) Three hexamers with boundary points.

a boundary element Poisson-Boltzmann electrostatic solver. This meshing code has been implemented in VolRover and made freely available from our CVC software page http://cvcweb.ices.utexas.edu/cvcwp/?page_id=91.

References


Algorithm 1 Potential vertex non-manifold detection and curate for classification map.

```plaintext
1: Set Tag(ijk) = 0, Tag(lmn) = 0, Tag(t) = 0, t = 1, . . . , 6.
2: if (fijk = 0) then ▶ Voxel ijk is not background.
3:    if (fijk = flmn) then
4:       if (f1 = fijk, t = 1, . . . , 6) then ▶ Curate.
5:          if (Any t = 1, . . . , 6, ft = fijk) then
6:             Set ft = fijk, Tag(t) = 1.
7:         else
8:             Randomly choose t = 1, . . . , 6. Set ft = fijk. ▶ Curate.
9:        end if
10:     else Manifold.
11:    end if
12: else
13:    if (flmn = 0) then Manifold.
14:    else
15:       if (f1 = · · · = f6) then
16:          if (flmk || flmk = fijk || flmk = flmn) then ▶ Curate.
17:             Two domains share a point. Manifold.
18:          else
19:             if (Tag(lmn) = 0) then ▶ Curate.
20:                Set flmn = flmk.
21:             else ▶ Curate.
22:                Set fijk = flmk.
23:           end if
24:        end if
25:     end if
26: else More than two domains. Manifold.
27: end if
28: end if
29: else ▶ Voxel ijk is background.
30:    if (flmn = 0) then ▶ Curate.
31:       if (f1 = · · · = f6 & & ft = 0 & & fijk & & ft = flmn, t = 1, . . . , 6) then
32:          if (Tag(ijk) = 0) then ▶ Curate.
33:             Set fijk = flmk.
34:          else ▶ Curate.
35:             Set flmn = flmk.
36:        end if
37:     else Manifold.
38: end if
39: end if
```

Appendix A. Algorithmic Pseudocode for Non-manifold Vertex and Edge Detection, and Curation
Algorithm 2 Potential edge non-manifold detection and curate for classification map.

1: if \((f_{ijk} \neq 0)\) then \(\triangleright\) Voxel \(ijk\) is not background.
2: \hspace{1em} if \((f_{ijk} = f_{lmn} \land f_{ijk} \neq f_t, t = 1, 2)\) then
3: \hspace{2em} if (Any \(t = 1, 2, f_t = 0\)) then
4: \hspace{3em} Set \(f_i = f_{ijk}, Tag(t) = 1\). \(\triangleright\) Cure.
5: \hspace{2em} else
6: \hspace{3em} Randomly choose \(t = 1, 2\). Set \(f_i = f_{ijk}\). \(\triangleright\) Cure.
7: \hspace{2em} end if
8: \hspace{1em} else
9: \hspace{2em} Manifold.
10: \hspace{1em} end if
11: else \(\triangleright\) Voxel \(ijk\) is background.
12: Manifold.
13: end if

43: else
44: \hspace{1em} if \((f_1 = \cdots = f_6 \land f_t \neq 0, t = 1, \ldots, 6)\) then
45: \hspace{2em} if \((Tag(ijk) = 0)\) then
46: \hspace{3em} Set \(f_{ijk} = f_{lmk}\). \(\triangleright\) Cure.
47: \hspace{2em} else
48: \hspace{3em} Set \(f_{lmn} = f_{lmk}\). \(\triangleright\) Cure.
49: \hspace{2em} end if
50: \hspace{1em} else
51: Manifold.
52: \hspace{1em} end if
53: \hspace{1em} end if
54: end if