Engineering with Computers (2009) 25:221–235
DOI 10.1007/s00366-008-0114-1

ORIGINAL ARTICLE

Reconstruction and simplification of high-quality multiple-region models from planar sections

R. H. Moore · G. S. Rohrer · S. Saigal

Received: 18 April 2007 / Accepted: 11 July 2008 / Published online: 10 December 2008

Abstract This paper proposes an accurate and efficient process that reconstructs, smoothes, and simplifies large-scale, three-dimensional models with multiple regions from serial sections. Models are reconstructed using a generally classed marching tetrahedra method that is less complex than the recent generally classed marching cubes algorithms, yet still preserves interface conformability between the regions in the model. Surfaces are smoothed using a volume preserving Laplacian filter that also preserves the region interfaces and topologies. Models are simplified using an efficient and accurate quadric-based edge contraction scheme that maintains the interfaces between regions and preserves the topology of the model. The edge contraction process is constrained to produce surface meshes that have high-quality facet shapes. The process both reconstructs as well as simplifies these models on-the-fly in one pass so that huge models may be processed within limited computer memory. The process does not require the entire original model to fit in the memory at one time. Example results of multiple-region models from the fields of materials science and medicine are presented.

R. H. Moore (✉)
Department of Civil and Environmental Engineering,
Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: CompSciEng@gmail.com

G. S. Rohrer
Department of Materials Science and Engineering,
Carnegie Mellon University, Pittsburgh, PA 15213, USA
e-mail: gr20@andrew.cmu.edu

S. Saigal
Department of Civil and Environmental Engineering,
New Jersey Institute of Technology, Newark, NJ 07102, USA
e-mail: Saigal@adm.njit.edu

Keywords Reconstruction · Simplification · Decimation · Smoothing · Surface meshes

1 Introduction

Computational models are used in many fields to estimate the performance of materials, structures, and systems. Often the data that forms the basis of three-dimensional (3D) models originates from a series of 2D images or readings called planar sections. The process of stitching these 2D serial sections together into a 3D model is called reconstruction. Many real-life models contain multiple separate regions, such as organs, tissue, and bone for medical applications; silt, rock, and clay for geological applications; and separate grains of the same material for polycrystalline materials for materials science applications. Multiple-region reconstructions present three specific challenges. First, reconstructions of multiple-region models must construct internal interfaces between the regions in addition to the external bounding surfaces. Most of the common reconstruction techniques do not construct correct interfaces between regions and, thereby, introduce undesirable gaps between the regions. A second challenge to the development of 3D models with multiple regions is the high face count that often results from reconstructions, since a large number of internal faces are needed to separate the regions inside the model. Model simplification is needed to reduce the face count to a size that is tractable on target computers. Many simplification strategies require the entire model to be in memory before simplification, which is not a practical possibility for very large data sets. Furthermore, most simplification schemes do not maintain the model interface integrity and do not preserve the topology of the model. Interface conformability and correct model topology are crucial for accurate analyses. A third challenge is the rough shape that results from most reconstruction methods. Many models have un-natural stair-steps and jagged features that are not present in the original data set. Few smoothing methods are able to remove the stair-steps while preserving critical interfaces and maintaining accurate region shapes.

This paper proposes an accurate and efficient process that reconstructs large-scale, 3D models with multiple regions from serial sections. The process both reconstructs and simplifies these models on-the-fly in one pass so that huge models may be processed within limited computer memory. The process does not require that the entire original model be in memory at one time, but produces final models that are small enough to fit in memory. The process provides a powerful smoothing procedure that maintains the model interfaces while removing the stair-steps and artificially jagged features. Furthermore, the simplification maintains the interfaces between regions, preserves the topology of the model, and generates well-shaped surface facets.

2 Background and related work

Recent advances in numerical methods and in computer speeds have enabled not only the visualization of complex 3D domains, but also their modeling and analyses. In building a model the integrity of the source of the input data must be maintained. This section describes common sources that are used to develop 3D models, reviews relevant reconstruction tools, and provides a brief and critical review of the available smoothing and simplification approaches.

2.1 Data sources

Medical applications often use computed tomography (CT) and magnetic resonance imaging (MRI) scans to evaluate patients. The results are usually formatted as a series of cross-sectional 2D images that are called serial sections. The images are segmented into separate regions based in part on the density values. Within materials science, most microstructural characterizations use destructive serial sectioning, where a material is consecutively polished (or ground) and photographed (or scanned) through a number of planar depths. Alkemper and Voorhees [1] presented a semi-automated method to acquire and photograph 20 serial sections per hour. The method was applied to an Al–Si alloy, where the material segmentation was straightforward, since the aluminum and silicon regions were represented as bright and dark regions, respectively, in the photographs. Segmentation of polycrystalline materials is more complex, since the regions are not easily distinguished by their relative brightness. Vedula et al. [2] used orientation imaging microscopy (OIM) to measure the angular orientation for each of the grains in a serial section and thereby distinguished the grains from one another. Groeber et al. [3] used a dual-beam focused ion beam machining system coupled with a scanning electron microscope (FIB–SEM) to provide automated serial sectioning with an OIM scan at each depth. A promising nondestructive approach has been advanced recently to measure 3D microstructures using X-rays [4]. This X-ray technique is especially valuable for polycrystalline materials, since it detects the boundaries between grains and measures the orientations of the grains. Within geophysics, seismic surveys and borehole studies are used to gather the data to reconstruct 3D terrain and sediment maps. These surveys and studies are not always based on regular grids, so reconstruction methods powerful enough to handle unstructured domains are especially desired.

2.2 Reconstruction methods

Reconstruction is the process of stitching 2D images into a 3D model. For multiple-region reconstructions, the method should construct 3D models without overlapping regions, mismatched faces, and artificial gaps. This section reviews relevant past efforts in light of these criteria.

Standard reconstruction tools process one material or region at a time and are termed “binary classified,” where separation surfaces are inserted between only two regions. For 3D regular grids, the marching cubes (MC) [5] method places separation surfaces at a predetermined density level or around one region. For data pre-formatted by contours, Boissonnat [6] and Boissonnat and Geiger [7] used Delaunay triangulation to reconstruct planar contours into 3D models. However, all these contours were assumed to be for the same region. Many models have more than one region of interest and are called “generally classified.” Binary classified reconstruction tools are insufficient for reconstructing generally classed models, where more than two distinct regions may be present. Holes, gaps, and overlaps will occur in most cases. That is, if one were to reconstruct each region separately, the reconstructed objects would not fit together, and the interfaces would not match. Thus, there is a need for reconstruction methods that handle generally classified data.

Generally classed reconstructions have been proposed to overcome many of the shortcomings of the original MC method when applied to multiple-region models. To date, these methods have had varying levels of success. Wallisch [8] used an octree method to subdivide each voxel, but this generalized MC method suffered from holes and incomplete surface coverage. Hege et al. [9] also extended the MC to handle generally classed cells. Voxels with more than two values were subdivided into inner structures and then processed in a way that ensures interface consistency. When each vertex/corner of a voxel may have one of three different values (or types), there are 6,561 cases to consider, which contrasts to the 256 cases in the standard MC. For four types of vertices, 65,536 possible configurations exist. In general each of the eight nodes in a cube may be different from the others, so the number of possible configurations is 8^8. To keep the computational requirements low, lookup tables were used for the standard MC cases and for some of the common three-vertex cases. Subdivision was performed on any remaining cases that were not covered by the stored cases. Overall, the implementation of [9] is extremely complex. Ju et al. [10] presented a multiple-material reconstruction technique that handles Hermite data and can be implemented in an octree form. This method places new vertices at positions that are based on minimizing a quadratic function, and these vertices are not constrained to fall on the edges of the cubes. These extra vertices are expected to fall near the intersection of the separation surfaces, but the points could possibly fall outside the cube and thereby generate models with intersecting faces. Ju and Udeshi [11] extended the work of Ju et al. [10] and presented a method that prevents intersections, but was not extended to work with multiple materials. Wu and Sullivan [12] presented a generally classed MC procedure that handles multiple materials and preserves interfaces. Each face of each voxel is examined and divided consistently. The region inside each voxel is then subdivided to match with the previously subdivided faces. This method is far less complex than that of [9], but it generates more right angle faces that contribute to the jagged stair-step appearance.

Reconstruction of multiple-region models from unstructured data sets was handled by Weinstein [13], who developed a method to reconstruct 3D models from generally classed planar contours. After the serial sections were aligned and contoured, the contours were extruded to voxels. The voxels were next subdivided into tetrahedrons followed by the extraction of separating surfaces. The resulting resolution is related to the resolution used to regrid the contours. It was noted that the contour point positions are not preserved, but converge towards the original values as the grid is refined. Nielson and Franke [14] developed a reconstruction approach based on dividing a voxel into six tetrahedra and then generating separation surfaces between regions. The division and surface generation were performed so that surfaces align with neighboring regions to produce closed and solid regions. Unlike the huge number of combinations of nodal values in the generalized MC approach, a generalized tetrahedral scheme has only five specific cases, so the coding complexity is minimal. Another strong advantage of this tetrahedra scheme is that the method can be applied to arbitrary unstructured domains, such as a Delaunay tessellation. A cubic input grid is not needed, so the method could be applied, for example, to geophysical borehole studies.

Since the reconstructions often produce rough and jagged models, smoothing is needed to remove the artificial stair-steps. Since the reconstructions produce a high number of facets, simplification is needed to reduce the model size to a more reasonable size. A few popular smoothing and simplification approaches are described next.

2.3 Smoothing approaches

The specific requirements for an effective smoothing process are that it must preserve the major features of the model, retain an accurate shape, and maintain interfaces and region topologies. This section discusses related work in the area of model smoothing.

The simplest Laplacian filters move each vertex, v_i', to a new position that is the average of adjacent vertices:

$$v_i' = \frac{1}{n}\sum_{j=1}^{n} w_j v_j \tag{1}$$

where there are n adjacent vertices, and the local weights, w_j, are typically set to one. An alternate filter moves each vertex to a new position that is related to its present position and average position of adjacent vertices:

$$v_i' = v_i + \frac{\lambda}{n}\sum_{j=1}^{n}(v_j - v_i) \tag{2}$$

where λ is in the range $(0 \le \lambda \le 1)$.

For effective smoothing, several filter passes or iterations are needed. These filters have undesirable features that they round over corners and reduce the volume of closed regions.

More advanced approaches have been proposed to preserve features and are based on surface mesh features beyond the adjacent vertices. Jones et al. [15] proposed a feature-preserving method that operates in one pass that they called a non-iterative feature-preserving mesh smoothing (NIFPMS) method. The new position for each vertex is based on projections of the tangent planes of each facet in the model. The contribution from each face is weighted by a Gaussian function that penalizes projections that make drastic moves in vertex locations. A further Gaussian weighting function minimizes the contributions of faces that are far from the vertex under consideration. While the NIFPMS method smoothes the models, no consideration was presented for optimizing the quality of the shape of the resulting faces or for handling multiple regions.

Improvements to the Laplacian filters have also been proposed, where the weighting functions are selected based on optimizing the shapes of the triangles that are connected to each vertex [16]. Well-shaped triangles are helpful for subsequent finite element meshing, since the accuracy of a finite element simulation is related to the shape of the elements. Further optimizations are targeted towards moving vertices to minimize the loss of volume of convex regions. Taubin [17] presented a Gaussian filter that roughly preserves the volume through bi-directional smoothing. The relaxation factor, λ, is alternated between positive and negative values between successive smoothing operations. Bade et al. [18] suggested that the positive value of λ be set within the range of 0.5–0.7, and the negative value should be -1.02 times λ.

Multiple-region models require special care in that the interfaces between regions present certain challenges. One basic approach is to smooth the faces that separate two specific regions using only faces and vertices that fall on the interface between these two regions. Wu and Sullivan [12] smoothed similar faces using a Laplacian filter and smoothed boundary line vertices using the two vertices that are directly before and directly after the vertex when traveling along a boundary line. While powerful, this method will reduce the volume on the concave sides of shared faces. The increased movement of regions also makes this approach more susceptible to invalid facet intersections. Kuprat et al. [19] presented a volume preserving smoothing method that respects interfaces and junctions between three regions. Their method both smoothes nodes and preserves the volume nearly down to machine precision.

2.4 Simplification approaches

The specific requirements for an effective simplification process are that it must handle multiple regions, preserve topology, maintain interfaces, and possess high accuracy. It must also not require that the entire original model fit in memory, so it can execute on-the-fly. Since the method is expected to process a large number of faces, the approach must be efficient and have a high processing speed. This section discusses related existing works in the area of model simplification in light of these criteria.

2.4.1 On-the-fly approaches

Early simplification techniques required the entire model to fit in the memory for two reasons. First, the techniques search the model to remove the features that result in the smallest change in shape. This comprehensive search implies that the entire model is already known and in memory. Second, when a feature is removed, the surrounding regions need to be re-stitched together, so those regions that are adjacent to a removed feature need to be present.

Approaches that handle partial models recognize the need to process meshes that are too large to fit in memory in their entirety. Grabner [20] presented an on-the-fly simplification technique that processed 2½-D models. This method took special care of features at the input boundary where faces were about to be read. The edge-dependency graph data structure used in [20] does not apply to multiple-region bodies, and the shortest-edge-length heuristic is not as accurate as other metrics. Furthermore, this method did not discuss topology preservation and was not applied to 3D meshes. An alternate approach for huge meshes is an out-of-core method. Lindstrom [21] presented an out-of-core approach that was based on a vertex clustering scheme. The model decimated faces and vertices at a very high speed (100,000 faces/s in the year 2000) and used an accurate quadric error metric to maintain the shape of the models. (One common quadric error metric considers faces that surround a vertex. When a vertex is relocated, the quadric error metric is equal to the sum of the squared distances between the new location of the vertex and the planes formed by the original faces that surrounded this point. If the squared distances are further scaled by the squared areas of the original faces, then the quadric error metric is related to a change in squared volume. Vertices are relocated to minimize these measures. More details may be found in Garland and Heckbert [22].)

Ju et al. [10] presented a method to reconstruct and partially simplify models that was based on an octree structure. Their approach simplified the mesh locally within a cube to reduce the size of the overall results. Their method preserved the topology of multiple-material models, but the placement of new vertices often resulted in intersected surfaces. Ju and Udeshi [11] extended the method of Ju et al. [10] to eliminate intersected facets, but their extension did not handle multiple materials.

Lindstrom and Silva [23] extended this out-of-core approach so that the simplified mesh did not need to fit in memory, and extended the boundary surface treatment so that features on the boundary of the leading-edge side would be preserved. The vertex clustering approach used in [21, 23] does not preserve the topology of models, since the vertices are merged within a cell and the fine connectivity is destroyed. Wu and Kobbelt [24] presented a stream algorithm for massive meshes that used the quadric error metric along with edge contraction to simplify a mesh. Candidate edges were contracted based on a randomized multiple-choice optimization, rather than sorting all available edges in a heap. This approach provided high speed and low memory requirements.

Isenburg et al. [25] extended the approach of Wu and Kobbelt [24] with a processing sequence paradigm. Only a small portion of the mesh is in the core memory while the much larger potion of the mesh resides on disk. The mesh is simplified as it streams through the in-core memory. The streaming mesh concept is described in great depth in [25–31].

To date, none of the streaming approaches preserve the topology and interfaces for multiple-region models, since the decimation methods employed were not designed for topology preservation. Special care must be exercised to preserve topology of even single-region bodies for edge contraction and vertex clustering methods. However, the topology and interface preserving algorithms could be used along with streaming meshes for a powerful and efficient process.

2.4.2 Speed and accuracy

A major goal of model simplification is to maintain the accuracy of the shape within a reasonable processing time. The different schemes often amount to finding the features to remove, and in some cases, determining where to relocate features during the removal. Vertex removal [32] scans and removes vertices when the resulting shape error is below a certain error tolerance. Since vertices are removed, all simplified meshes have vertices that are a subset of the original model. Wu and Sullivan [12] extended the vertex removal method to handle multiple regions while preserving interface conformability and region topology. Vertices were classified as simple vertices, in that they separated two regions, or edge vertices, in that they fell along edges that divided three or more regions. The method decimated the models until no more vertices met the error criteria or when the target decimation levels were reached. Vertex removal schemes are considered less accurate than more modern schemes for three reasons. First, the error metric is not as advanced and accurate as more modern metrics, such as the quadric error metric of Garland and Heckbert [22]. Second, the vertices are simply removed, rather than adjusting the location of features in the model to minimize the shape error. Third, most schemes do not pre-order the candidate vertices according to error, so vertices having greater error may be deleted before lower error vertices.

The quadric-based edge contraction procedure in [22] is considered as the best trade-off between accuracy and speed [33]. Two major aspects work to achieve this. First, the quadric error metric provides an upper bound for the more accurate Hausdorff error [33]. Also, the quadric error metric is often scaled to produce physical measures of error that are related to the square of the change in volume when an edge is removed. When an edge is contracted, a merged vertex is placed at a location that minimizes this error. Second, the method uses a heap that contains edges that are keyed with contraction cost. The lowest cost edges are removed in sequence, so the simplification progresses in terms of making the smallest changes in shape first. Furthermore, heaps are very fast and have $O(N\log N)$ performance. Simplification envelopes [34] provide a different approach to maintaining the shape of a model. Two envelope surfaces are placed next to the original surface. One envelope is a small distance inside the original surface and the other envelope is a small distance outside the original surface. Model simplification is constrained such that the final surface falls within these bounding envelopes. This technique may provide the highest level of accuracy for single-bodied objects, and it preserves the topology of single bodies, but is very complicated to code. No extension was offered for multiple-region models. Moore [35] presented a technique that preserves the topology and interface conformability for multiple-region models, yet uses a quadric error metric coupled with edge contraction to maintain the accuracy of the model. Candidate edges are only contracted if no topology change would result and if interfaces remain conformable. (A conforming interface is the shared interface region between two touching elements, where every pair of touching elements must share the same three vertices of a single face, or share the same two vertices of the same edge, or share one and the same vertex.)

The approach of Moore [35] may be described within the framework of boundary representation solid modeling (B-Rep), where an Euler operator removes an edge and maintains the topology of the modified model.

An alternate approach to model simplification was discussed by Dey et al. [36], where a link-test was applied to each edge under consideration for removal. If the link-test was passed, the edge removal would not alter the topology of the model. Their discussion proved the correctness of the link-test for single-region bodies and for bodies that had extra edges or boundaries. Vivodtzev et al. [37] adapted the link-test for multiple-region models and demonstrated that the topology and interfaces were maintained during simplification.

Tetrahedron mesh simplification was discussed by Cutler et al. [38] as an alternate approach to the surface mesh simplification. Multiple tetrahedron regions were simplified in their process.

This paper extends the approach of [35] to further handle on-the-fly model simplifications that are coupled directly to model input and reconstruction. The process is accurate, fast, and can provide highly reduced models that preserve topology and interfaces. Since an immediate goal is to produce models suitable for finite element meshing and modeling on workstation-class computers, the models are reduced to a size that fits within in-core memory, and the mesh face shapes are constructed to have high quality.

3 Multiple-region model reconstruction and simplification

This section describes the proposed process for building and simplifying large-scale models from serial sections. The proposed reconstruction and decimation method starts at one end of a model and repeatedly reads the input data and simplifies the in-core model until the entire model is processed. After each input reading step, the newly input readings are reconstructed into a 3D surface model, the data structures associated with the simplification method are updated, and the in-core model is simplified until a termination criterion is satisfied. The following sections describe these steps in more detail. The strategy proposed here is fast, efficient, memory conscious, and it preserves topology and interface conformability.

3.1 Read in a strip

For our implementation, we assume that the input data is pre-segmented in that each data point is numbered to represent a region. For input data sets arranged on a rectangular grid, the process sequentially reads in one layer of voxels, as shown in Fig. 1. The memory required to read a strip of voxels is reduced when the model is oriented such that face with the smallest number of voxels is the one that is read. For example, if the x-dimension in Fig. 1 is very large, then the y–z face would have fewer voxels than either of the x–y and x–z faces.

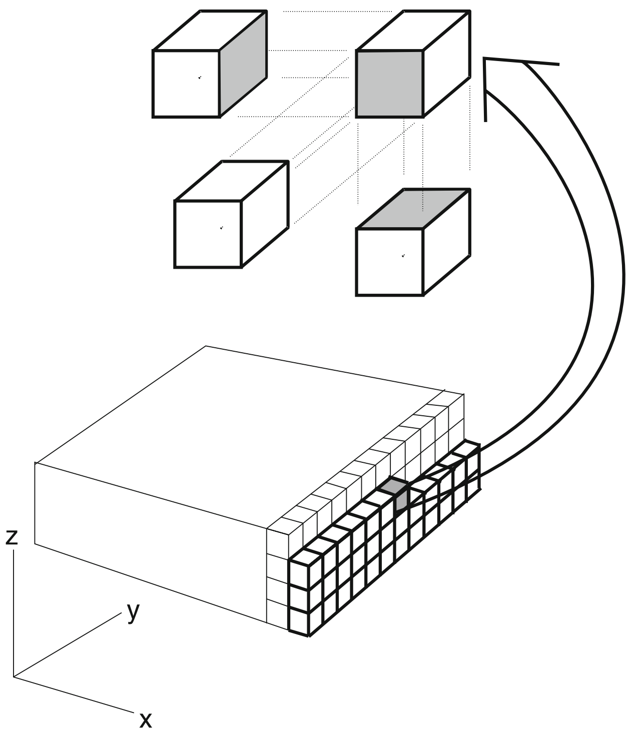

Fig. 1 Reading new strips and voxels

3.2 Reconstruct using generally classed marching tetrahedra

Region boundary faces are generated for each voxel in the freshly read strips using the method of [14]. Each voxel in the strip is first divided into six tetrahedra, as shown in Fig. 2, which ensures that all faces of the tetrahedra align, match, and conform with each other within the voxel and also with the tetrahedra of the neighboring voxels. Each of the six tetrahedra is then divided into separate regions, based on the region numbers of its vertices. This dividing surface scheme produces additional vertices both on the surface of the voxel and internal within the voxel, as shown in Fig. 3, for the case when only one vertex has a different region number than the other three vertices that share a common region number. Through the subdivision of a voxel into tetrahedra, and the subsequent generation of separation surfaces, the boundary surface model is advanced as the strips are read.

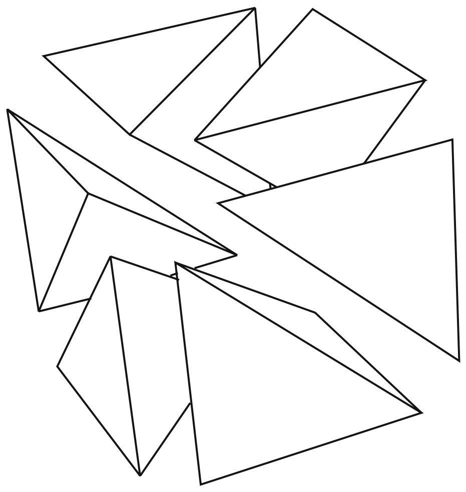
Fig. 2 Voxel divided into six tetrahedra

The generally classed marching tetrahedra style method proposed in [14] has a few advantages over other schemes. First, the method is simpler than generally classed MC, since each tetrahedron has only five different cases for separation surfaces, including the trivial case when all vertices are of the same region number and no separation surfaces are generated. Second, topological ambiguities associated with MC are avoided. Third, tetrahedron-based reconstructions can be used on arbitrary unstructured domains and do not require structured rectangular grids. A disadvantage of the marching tetrahedra method is that the subdivision of each voxel into six tetrahedra can result in a larger number of vertices than the number produced using generally classed MC. There is a tradeoff between the

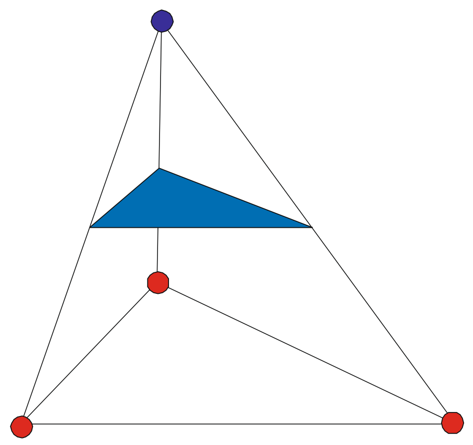

Fig. 3 Separation face that separates a tip of a tetrahedron

increased need for mesh simplification with the marching tetrahedra method versus the complexity of the generally classed MC methods. This disadvantage is not severe, since we simplify the mesh after each strip of voxels is input, as will be discussed in Sect. 3.5.

Generally classed reconstruction schemes must store additional data to distinguish among the different regions. All faces separate two regions or bound a single material, and the one or two face numbers are stored along with the face structure. Likewise, vertices may touch more than one region, and for convenience, these region numbers are stored along with the vertex structure.

3.3 Link points and faces to the global model

After the separation faces and vertices are formed locally for a strip, these entities are added to the global model of previously processed faces and vertices. The strip information is transferred voxel by voxel, and each voxel shares maximally only three faces with voxels that have already been read and processed, as shown in Fig. 1. The expanded voxel in Fig. 1 has adjacent voxels on the left, below, and behind it. Since some of the newly created vertices fall on the faces of the cubes, there may already exist matching vertices in the global model, in which case the duplicates are purged. New vertices are added to the global vertex list either at the end or inside available memory spaces that are vacated during a previous decimation step. In this way, memory is reused and conserved, which reduces the memory requirements of the processing system.

3.4 Update structures

The geometrical model primarily consists of vertices and faces, but additional data structures are used to make the process run more efficiently. In particular, each vertex has associated lists or arrays that store the faces, edges, neighbors, and error measures that touch or relate to the vertex. These lists are updated after each strip is read.

Our implementation stores the vertices for the presently read strip and for the previous two strips that were recently read and processed. The present strip contains vertices that are on the leading edge of the model input, and all the faces that touch these vertices are not yet known. However, the vertices for the previous strip are fully surrounded by faces (a condition called finalized in [25]), so quadric error characterizations can be made for these vertices. Furthermore, the quadric error for edges between vertices of the previous two strips can be computed, and the edges may be added to the edge heap, since their vertices have already been finalized. Edges that touch the leading-edge vertices cannot be fully characterized, so they are examined after the next strip is read.

3.5 Decimate when necessary

The previous step added faces and vertices to the model description and edges to the edge heap. When the face count threshold is exceeded, the mesh is decimated down to or below this threshold. Moore [35] outlined an edge contraction approach to decimate multiple-region bodies that preserves the interfaces and topology. A few features of the approach that apply to decimating on-the-fly are reviewed here.

Decimation is performed by iteratively removing edges until some termination condition is reached. For on-the-fly implementations, the list of edges includes only the edges in the model that have been read so far. The edge contraction scheme is similar to [22] where the candidate edges are stored in a heap and sorted by the contraction error cost. The lowest cost edges are removed and the affected edges and heap are updated until termination. Termination will occur if the scheme runs out of candidate edges to delete, if the edge contraction error is too great, or if the targeted face count is attained. When an edge is contracted, the model is usually simplified by the removal of three edges, two faces, and one vertex. In some situations, a different number of these items is removed. In all cases, the memory that was occupied by these entities is made available for subsequent model input. In this way, memory is conserved/reused as efficiently as possible.

To preserve the topology and interface conformability of the model, the decimation process must be more restrictive than to simply remove the lowest cost edge. Each edge is scrutinized in depth before contraction. The edge is contracted if it passes this scrutiny or it is rejected and removed from the edge candidate heap. The extra tests are documented in [35] and are summarized as follows:

1. Check to ensure that the edge contraction does not cause triangles to fold over themselves.
2. Check that the edge contraction does not cause a corner to pinch close and thereby produce zero-volume slivers.
3. Reject the removal of an edge that joins two vertices whose region numbers do not match. If the region number list of one vertex is a subset of the region numbers of the other vertex, then that edge may still remain as a valid candidate for edge contraction.
4. Check for resulting changes in the model after an edge is removed. If any edge results in having more than two faces of the same material number, the edge is rejected, and will not be contracted.
5. Ensure that the Euler–Poincaré characteristic does not change during an edge contraction. This test will be discussed in depth in the next section.
6. Ensure that the modified faces do not intersect or self-intersect with other faces in the model.

In addition to the topology tests, the decimation process also performs tests to maintain the quality of the shape of the facets.

1. Reject contractions that produce at least one modified face that has a quality, Q, that is below a pre-specified threshold. Our process uses the scale-invariant quality metric, Q, proposed by Shewchuk [39].

   $$Q = \frac{\text{Face area}}{[l_{\min} l_{\text{med}}(l_{\min} + 4|r_{\text{in}}|)]^{2/3}} \tag{3}$$

   where $l_{\min}$ and l_{med} are the smallest and middle triangle edge lengths, respectively, and r_{in} is the signed inradius or the radius of the inscribed circle for this face. An advantage of this quality metric is that it characterizes the quality of an element used in finite element analyses. The quality, Q, is near-zero for the poor shapes and negative when the face area is negative. Equilateral triangular faces have a quality of 0.26, right isosceles triangles have a factor of 0.24, and isosceles triangles that have an obtuse angle of 135° have a factor of 0.16.
2. For edge contractions on flat surfaces, the quadric error is zero for all merged-vertex positions that fall on the flat face. The program computes the quality for all the faces that are modified for this merged-vertex position. If allowable, the program also computes the quality for of each face that is modified for the case where the merged-vertex position is taken as the average of its surrounding vertices, as when a simple Laplacian filter is used. If the quality, Q, of the poorest shaped facet is above a minimum threshold quality, the process takes the higher quality result of either the quadric position or the Laplacian position.

These two tests maintain the quality of the model to remain above a pre-specified minimum.

3.6 Preserving topology on-the-fly

It is essential to preserve the topology of every region in the model while the model is simplified. However, for on-the-fly simplifications, at any instant only a part of the model has been read and reconstructed, and the final topology is not yet known. Moore [35] scrutinized edge contractions based on the constraint that the Euler Characteristic must be invariant during simplification to preserve the topology. The Euler Characteristic, X, is given as

$$X = 2(S - G) + L = V - E + F \tag{4}$$

where V, E, F, and S are the number of vertices, edges, faces, and solids or bodies, respectively; G is the genus, or the number of through holes; and L is the number of inner hole loops. Since V, E, and F for an object are fixed, X is invariant as seen from (4).

Simplifications that reduce V, E, and F are performed in such a way that X remains unchanged. For on-the-fly models, the final complete Euler characteristic, X, is not yet known, but some value exists after each strip is read. The on-the-fly simplification maintains this temporary value of the Euler characteristic, Xt, to be invariant only during the simplification stage at each input step. Since the on-the-fly process uses edge contraction to simplify models, and since only edges away from the leading edge are considered, the process will only simplify local regions where the local topology is complete and known.

Conformance to the Euler–Poincaré law is a necessary condition for the correctness of the topology of a closed and solid model. However, the Euler–Poincaré Law is not a sufficient condition. Insidious combinations of dangling vertices, edges, and faces can combine together to satisfy the Euler–Poincaré law. Therefore, the specification of the topology of a model is further refined to include the following constraints: (a) all vertices must have at least three edges, (b) all edges must have two, and only two, faces with the same region number, (c) each edge has two vertices, and (d) no two distinct vertices can occupy the same spatial location. These constraints must continue to be maintained as the model is simplified.

3.7 Refresh heap if necessary

If the decimation loop terminates because either the candidate error was too high or the edge heap has been depleted of edges, then the heap is refreshed with completely updated information. All edges in the model that are in memory are considered again for inclusion in the heap.

The edge heap can be depleted of low error edges for a number of reasons. The main reason is face flips, where a contracted edge causes a face to invert or fold over itself. The edge would be rejected in this case. However, after nearby edges are contracted, it may be possible that this previously rejected edge may now be a viable candidate. For severe decimations, edges need to be reconsidered after initial rejections. Some sample studies showed that 54% of the edges were rejected from face flips, although this high number includes edges that were rejected multiple times.

The need for reconsideration for edge contraction limits the applicability of some past edge contraction codes for severe decimation in one pass. Many edges may not be available within a single pass. Multiple decimation passes are often necessary to achieve low face counts. However, in the present system, the model is input in one pass, but the heap of candidate edges may be updated more than once.

3.8 Postprocess results

After the model is decimated, the simplified model is output to a disk file and is also checked for correctness. The Euler–Poincaré law is used to ensure that all regions are solid and closed. Further checks are made to ensure that the model is correct and free of dangling vertices, faces, and edges.

3.9 Comments on unstructured data

Unstructured data, such as bore-hole studies, produce data at various locations that do not fall on a regular grid. A Delaunay tessellation would be constructed from these data points, and the tetrahedra could be sorted in one direction. The marching tetrahedra method would be used to construct separation surfaces for the tetrahedra. Only the separation surfaces and boundary faces would be stored to disk. The faces could be sorted in one direction if the prior sort were omitted. Subsequent processing would read these faces from the disk file until a certain face number threshold is reached. The process would note when each vertex is finalized and would update the quadric error and heap for each edge that has two finalized vertices. The process would then alternate between simplification and face input until the entire model has been processed, as was done with the planar section data.

4 Results

The reconstruction and simplification process proposed above was applied to examples in the fields of materials science and medicine. These analyses were performed on a Dell Inspiron 8600 laptop computer with a 1.6 GHz centrino processor, 1 GB of memory, and Windows XP Professional. The source code was compiled as a console application using Microsoft Visual C++ 6.0. The defaults compiler settings were used, so the program was not optimized for speed, and the application priority was not adjusted within the Windows task manager. Thus, the execution times are expected to be typical for common use, rather than optimized for fastest performance.

4.1 Memory requirements

Memory studies on various models indicate that the data structures of the process require about 165 bytes/face for a memory conserving implementation and about 224 bytes/face for standard double precision and 32-bit integers. Since the process handles multiple regions, each face stores the two integer region numbers that it divides, and each vertex stores four region numbers that it may touch. Exact memory predictions cannot be obtained, since the data structures that track the neighbors, edges, and faces for each vertex are not fixed in size. On average, for the models reported here, however, each vertex keeps track of about 6.25 faces, 6.15 edges, and 6.15 neighboring vertices. These memory statistics are useful for estimating the largest model size that may be processed in-core for a given amount of computer memory, even though they may vary from model to model.

4.2 Smoothing

Smoothing is necessary to remove the jagged stair-steps found in many reconstructed models. The smoothing method should also preserve the overall shape and important features of the models. The performance results of three filters are shown in Fig. 4 for a chamfered rectangular solid. Note that repeated application of the simple Laplacian filter evolves the model to a cigar shape with a reduced volume. Repeated application of the volume preserving Laplacian filter still maintains the overall shape, but the edges are rounded. More iterations are needed to evolve to the cigar shape, although the volume is preserved. The one-pass NIFPMS filter shows slight warping, but the overall shape and sharp edges are retained.

The performance for smoothing pyramids with jagged stair-steps is shown in Fig. 5 for a double-pyramid shaped object, where the four center base edges are marked as interface edges. The simple Laplacian filter reduces the model to a flat square pancake. The volume preserving Laplacian filter retained the overall shape and size, but the edges were slightly rounded. The NIFPMS filter rounded the overall shape, yet did not remove the stair-steps.

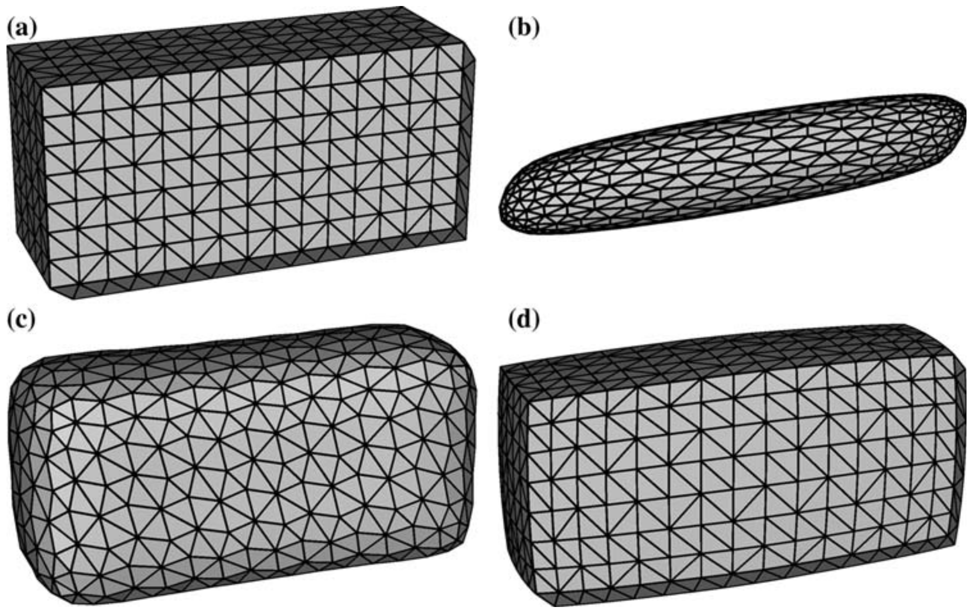

Fig. 4 Smoothing a chamfered slab: **a** original, **b** Laplacian filter, **c** volume preserving Laplacian filter and **d** NIFPMS filter

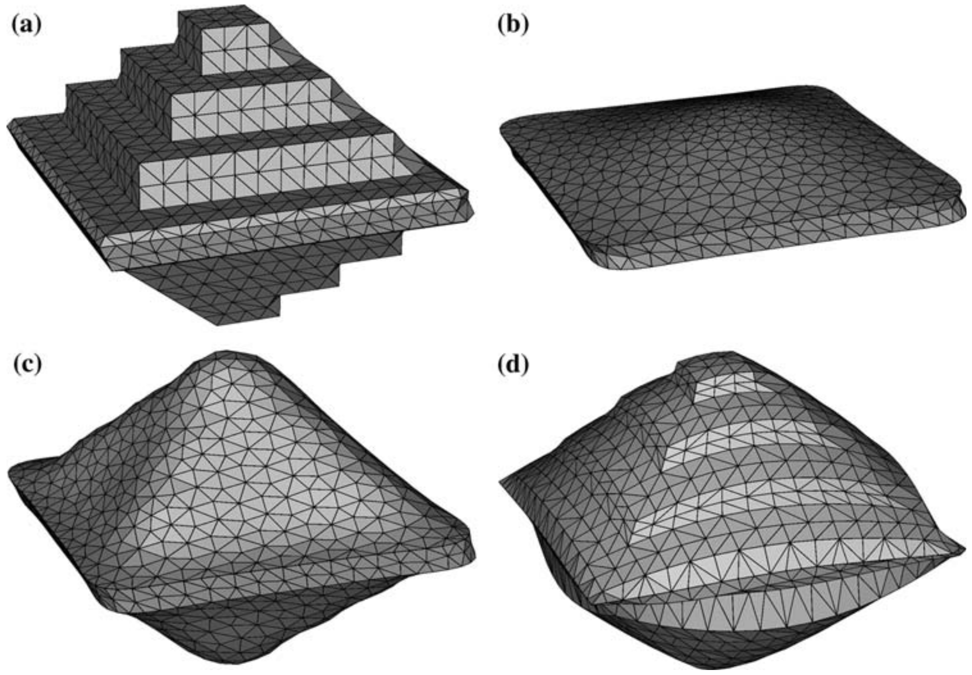

Fig. 5 Smoothing stair-steps on a double-pyramid: **a** original, **b** Laplacian filter, **c** volume preserving Laplacian filter and **d** NIFPMS filter

To preserve the topology and interfaces between regions, the volume preserving smoothing method of Taubin [17] was modified as follows. Vertices are smoothed using vertices that have similar region attributes. Formally stated, a vertex, v_i, that touches a set of regions, R_i, is smoothed based only on neighboring vertices, v_j, where each vertex in v_j has a set of regions, R_j, that are equal to R_i or are a superset of R_i. In a similar manner, the NIFPMS filter was applied separately to the surface facets between pairs of grains. The lines that separate three or more grains were smoothed separately using only neighboring data along these same lines.

The smoothing performance was also applied to polycrystalline models, where one grain was displayed individually, as shown in Fig. 6. Visually, the volume preserving Laplacian removed the stair-steps more effectively than the NIFPMS filter, although 100 passes were required. As seen in Fig. 6, fewer passes showed significant stair-steps. The NIFPMS filter also developed depressions in the flatter sections of the grain, and undesirable spikes in the upper right hand corner of Fig. 6d. The surfaces between pairs of grains are typically regarded as smooth and devoid of sharp features, so there is less need for the feature-preserving strength of the NIFPMS filter for these

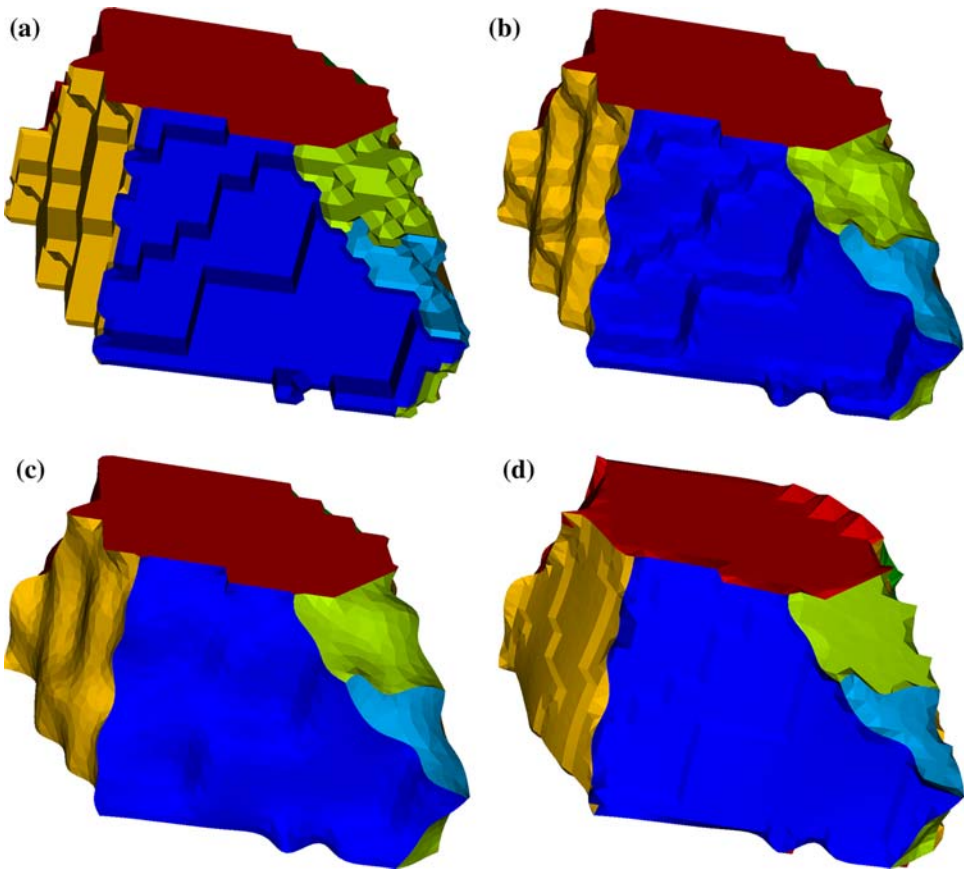

Fig. 6 Smoothing a grain in a polycrystal: **a** original, **b** 20 passes volume preserving Laplacian filter, **c** 100 passes volume preserving Laplacian filter and **d** NIFPMS filter

regions. The lines between three or more regions remained sharp and were preserved with both modified methods, as seen in Fig. 6c, d. Given the simplicity of the volume preserving Laplacian and its effectiveness, this filter was chosen for the overall processing system.

4.3 Decimated face quality

Standard edge contraction decimation techniques move vertices to maintain the shape of a model, but the resulting facet shapes may be poor. An example of a decimated grain is shown in Fig. 7 for when the face quality was not considered and also when quality factor of 0.1 and 0.16 were specified. The triangular facets in Fig. 7b, c have increasingly better shapes than the facets in Fig. 7a. We expect that models that have high-quality faces will be easier to mesh with high-quality tetrahedra.

4.4 Polycrystalline solid

The reconstruction and simplification process was applied to a sample of polycrystalline magnesia. The data was obtained from [40] and the sample contained 2,400 individual grains. The data was available for five serial sections, each on a 1,740 $\times$ 1,303 grid. Without model simplification, the model had 25,722,389 faces and 12,443,888 vertices. If this model were fully stored for in-core simplification, it would require between 4.24 and 5.76 GB of memory. This model is within the reach of large modern computers, as the input data is fairly small at only five sections. The model size did exceed the memory capacity of the processing computer used in this study, so on-the-fly simplifications were necessary.

The polycrystalline model was decimated 90% to obtain a face count of 2.5 million faces for the example shown in Fig. 8. The execution time was 5,111 s. The average processing rate is about 4,544 faces/s, which includes the time required to read the data file and reconstruct the 3D surface models. The model was checked for topology and interface errors and passed.

4.5 Heap performance

The significant number of co-planar surface faces shown in Fig. 8 can cause performance inefficiencies for the heap-based data structure that is used to store edges. Purely planar edge contractions have zero-error for the quadric error metric, so the edge heap would be loaded with a significant number of zero entries. On average, heaps have $O(N\log N)$ behavior and $O(\log N)$ access times when the

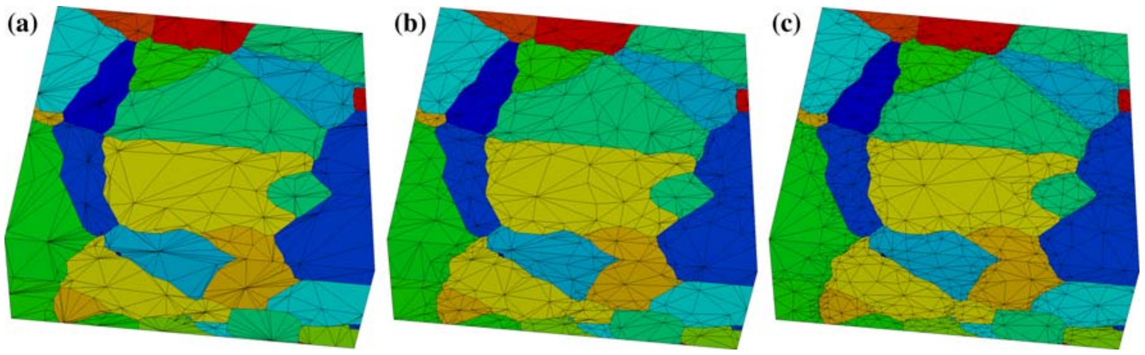

Fig. 7 Surface mesh facet shapes for three specified quality levels: **a** $Q = 0.0$ (no restriction), **b** $Q = 0.1$, **c** $Q = 0.16$

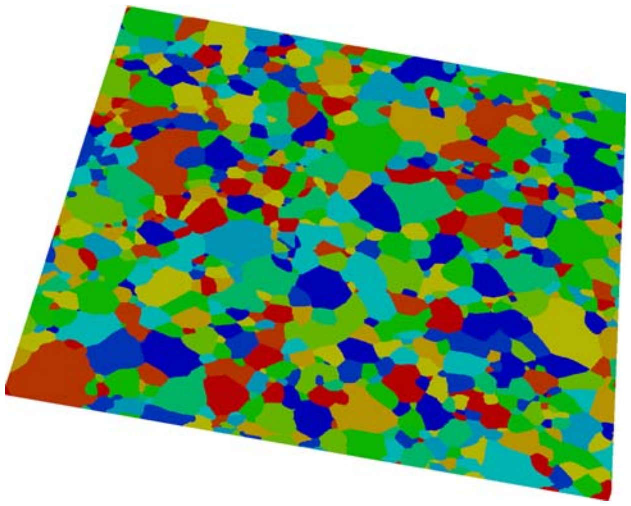

Fig. 8 Reconstructed and simplified model of polycrystalline magnesia

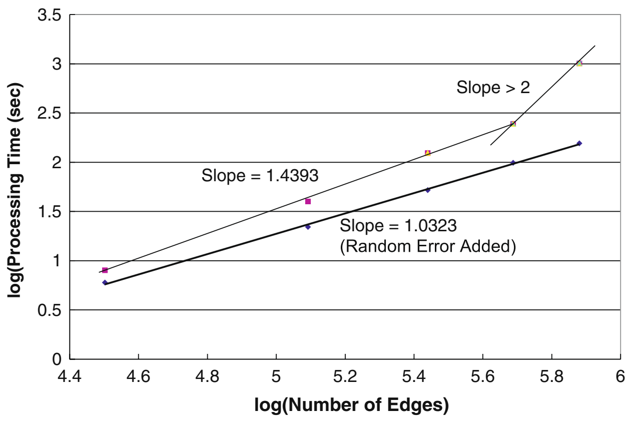

Fig. 9 The addition of a small random edge contraction error enhances process performance

data are irregular or random. When a long string of identical data is input, the heap insertion costs approach $O(N)$. Thus, the cost to build a full heap would be closer to $O(N^2)$ for a long string of identical inputs. To illustrate this effect more clearly, consider a flat rectangular slab of a single region. Most of the edges have zero contraction error. As shown in Fig. 9, the decimation time for small N is of the order $O(N^{1.4})$ and worse than order $O(N^2)$ for larger N. In a similar manner, MC and marching tetrahedra-based reconstruction algorithms may produce a significant number of discrete nonzero error values that represent common combinations of output faces. The heap-based performance is again expected to decline in this case.

Edge contractions on all zero-error edges could be performed before adding them to the heap. The coding in this case, however, becomes more complex and the other nonzero plateaus would not be handled by this approach. One novel solution to this inefficiency is to perturb each edge contraction error by an insignificant amount, say up to 1.0e-7. A random number generator adds a small error to each computer error so that the heap will see different values for each entry. The error caused by this addition is smaller than the smallest expected nonzero difference in edge contraction error. As shown in Fig. 9, the addition of the random perturbation error brought the algorithm back to the optimal $O(N\log N)$ behavior. The highest face count example in Fig. 9 ran nearly 6.5 times faster when the random error was added. More significantly, for the polycrystalline magnesia example shown in Fig. 8, the addition of the random perturbation reduced the processing time by nearly 20% (4,119 s). Unless otherwise noted, the random error perturbation was performed for all the results reported here.

4.6 MRI medical example

The reconstruction/simplification process was applied to a phantom head model available in [41] that was based on MRI data. This pre-segmented model has 128 serial slices, each of size 256 × 256. The reconstructed model would have had initial 6,691,110 faces, but was reconstructed on-the-fly and simplified to 2 million faces in 944 s. A different complete reconstruction/simplification process produced a model with 1,670,000 faces in 942 s. The nearly equal processing times demonstrate that the process balances between the time required to maintain the heap data structure and the time required to delete edges. In any case, the original model was about 50% too large to fit in memory and had to be processed on-the-fly.

Although all 61 regions were reconstructed and simplified together by the process, for simplicity, only the brain, spinal cord, cerebellum, medulla oblongata, and pons are shown in Fig. 10. The brain and cerebellum in Fig. 10 were visualized as semitransparent light gray and dark gray, respectively, to indicate that the regions fit together. A stronger visual test for interface conformability test is to plot the un-matched faces. Only the faces on the outside of the head were displayed in our results, since internal faces were properly matched, demonstrating the validity of the present approach.

4.7 Small-size performance

The performance of the proposed reconstruction/simplification system was further examined through a set of analyses on a small subset of the polycrystalline solid data.

The processing system was applied to five serial sections of a subset of the polycrystalline example from Fig. 8. Small regions of respective sizes $50 \times 50, 50 \times 100, 50 \times 200, 50 \times 300, \ldots, 50 \times 1,000$ were individually reconstructed and simplified. A nearly linear performance of the algorithm as shown in Fig. 11 was observed. The total model processing times for decimating down to 10% of the initial model sizes are shown by the higher curve, and the decimation times alone are shown by the lower curve in Fig. 11. For these small models, the entire models were loaded into memory before decimating. The nearly linear performance shown in Fig. 11 indicates that the processing system is efficient and functions at no worse than $O(N\log N)$.

The aggressiveness of the decimation level employed affects the processing time, as shown in Fig. 12. The same models were used as shown in Fig. 11, but various levels of decimation were considered in obtaining the results. The on-the-fly decimation processes more quickly than the approaches that read the entire model before simplification. Of course, this latter option is only available for models small enough to fit into memory. An interesting effect is that there is little difference between the 10, 25, and 50% decimation times for these on-the-fly examples. Even though the more aggressive decimations remove more faces, the heap construction and maintenance costs are lower, since the heaps are smaller. For less efficient data structures, or possibly for heaps that do not add the random error quantity, the more aggressive decimation models actually run faster than the models that retain the majority of their faces. There are limits to this trend, however, as will be discussed in the next section.

4.8 Limits on decimations

The effect of decimation level on processing time is seen more clearly in Fig. 13 for a single model of size $50 \times 1,000$. The overall processing times follow the expected trend that greater decimations require greater processing time. For this particular example, at levels of aggressiveness determined by percentage remaining, beyond 45% the processing time levels off until the model reaches a simplification level of about 12% where most remaining edges cannot be contracted. For model sizes

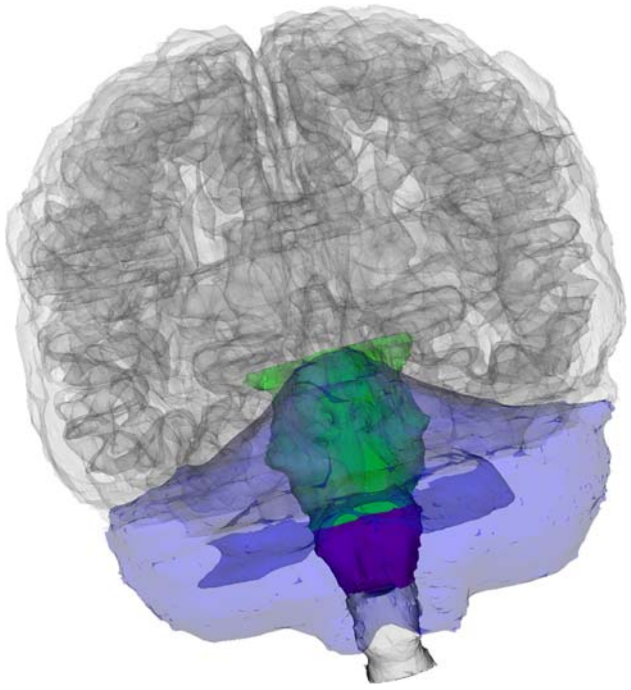
Fig. 10 Reconstructed head showing five connected regions

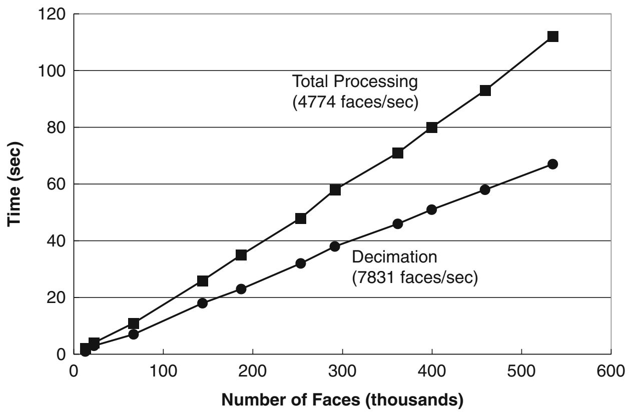

Fig. 11 Processing performance for various size models

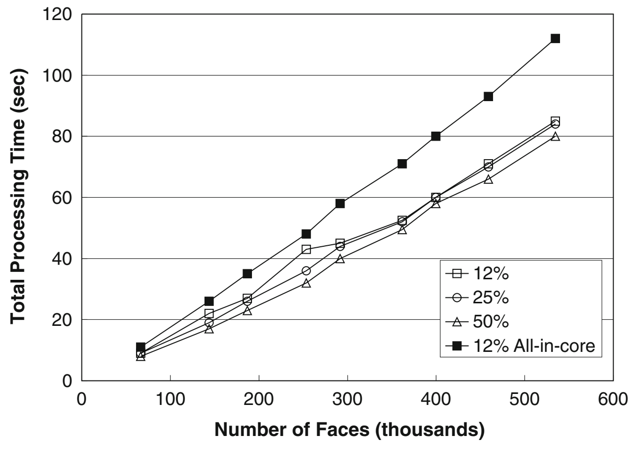

Fig. 12 Processing time versus decimation aggressiveness

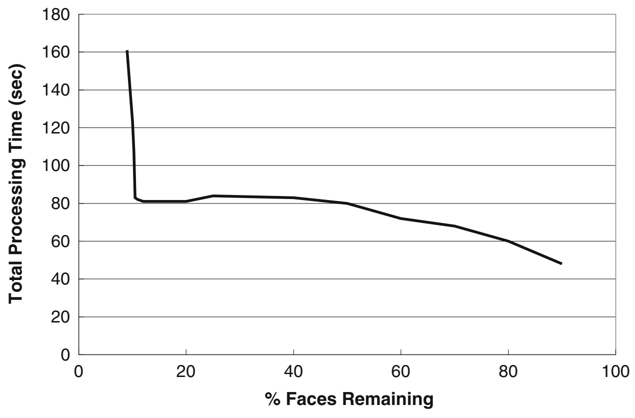

Fig. 13 Processing efficiency degrades for severely decimated models

below 12%, the process may spend more time performing computationally expensive tests on the remaining edges to ensure that the topology and interface conformability are preserved. This drop in performance would not be seen with most single-body regions that are allowed to evolve to a single tetrahedron. However, high-genus single-body models would see performance drops at extremely high levels of decimations.

5 Concluding remarks

This paper presents a process to reconstruct, smooth, and simplify multiple-region models on-the-fly. The method can be used to process models that initially are too large to fit inside the in-core memory of a computer. The quality of each facet in the mesh is constrained to exceed a pre-specified threshold. The method preserves the topology and interface conformance for all the regions and produces models that are suitable for further computational analysis, such as for finite element analyses.

Models are reconstructed using a generally classed marching tetrahedra method that is less complex than the recent generally classed MC algorithms, yet preserves interface conformability between the regions in the model. Model simplification through the use of quadric-based edge contraction is computationally efficient and produces accurate models. With the un-optimized code written for this study, the method processes files at average rates of above 4,500 faces/s. The performance was enhanced from 20 to 650% through the addition of a very small random error to the conventional quadric-based edge contraction error. This small error prevents long sequences of identical entries in the edge heap and brings the heap performance back to $O(N\log N)$ performance levels.

The results indicate that the on-the-fly performance is faster than the performance of simplifying after the entire model is in memory, since the heap maintenance times are reduced. There is a tradeoff in that aggressive decimation has shorter heap maintenance costs, but requires more computations to delete the greater number of edges. Even so, the timing differences between decimation levels of 12 and 50% are small for the examples reported here.

The model simplification performance degrades at highly severe level of decimations, since many computationally expensive tests are performed to find the small number of valid remaining edges to remove. These expensive tests are necessary to preserve the interfaces between regions and to maintain the region topologies. This drop in performance would not be seen with most single-body regions that are allowed to evolve to a single tetrahedron. However, high-genus single-body models would see performance drops at extremely high levels of decimations.

Multiple decimation passes are needed to simplify a model down to low percentages, since many edges are initially rejected to prevent faces from folding back over a model. Alternately, rejected edges should be reconsidered later when they may not cause local folding. On-the-fly processing can still be performed in one pass while allowing multiple decimation cycles by reading the model in one pass and by repeating the decimation loop whenever necessary.

A volume preserving Laplacian smoothing method that respects interfaces was applied to the model to remove stair-steps and jagged edges.

Acknowledgments The authors would like to acknowledge Dr. A. D. Rollett for his many constructive comments for improving this paper. The authors would also like to thank Dr. David Saylor and Mr. Suk-Bin Lee for providing the segmented and aligned serial section data.

References

1. Alkemper J, Voorhees PW (2001) Quantitative serial sectioning analysis. J Microsc 201(Pt 3):388–394
2. Vedula VR, Glass SJ, Saylor DM, Rohrer GS, Carter WC, Langer SA (2001) Residual stress predictions in polycrystalline alumina. J Am Ceram Soc 84:2947–2954
3. Groeber MA, Haley BK, Uchic MD, Dimiduk DM, Ghosh S (2006) 3D reconstruction and characterization of polycrystalline microstructures using a FIB-SEM system. Mater Charact 57(4–5):259–273
4. Poulsen HF, Nielsen SF, Lauridsen EM, Schmidt S, Suter RM, Lienert U, Margulies L, Lorentzen T, Jensen DJ (2001) Three-dimensional maps of grain boundaries and the stress state of individual grains in polycrystals and powders. J Appl Cryst 34:751–756
5. Lorenson W, Cline H (1987) Marching cubes: a high resolution 3D surface construction algorithm (Proceedings of SIGGRAPH ‘87). Comp Graph 21(4):163–169
6. Boissonnat JD (1988) Shape reconstruction from planar cross sections. Comput Vis Graph Image Process 44(1):1–29
7. Boissonnat JD, Geiger B (1992) Three dimensional reconstruction of complex shapes based on the Delaunay triangulation. Technical report no. 1697, INRIA, France
8. Wallisch B (2000) Internet-based visualization of basin boundaries for three-dimensional dynamical systems. The 4th Central European seminar on computer graphics in 2000. (http://www.cg.tuwien.ac.at/~wallisch/da/)
9. Hege H-C, Seebaß M, Stalling D, Zöckler M (1997) A generalized marching cubes algorithm based on non-binary classifications. Technical report SC-97-05, Konrad-Zuse-Zentrum für Informationstechnik (ZIB)
10. Ju T, Losasso F, Schaefer S, Warren J (2002) Dual contouring of hermite data. ACM Trans Graph 21(3):339–346. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002)
11. Ju T, Udeshi T (2006) Intersection-free contouring on an octree grid. In: Proceedings of Pacific graphics
12. Wu Z, Sullivan, JM Jr (2003) Multiple material marching cubes algorithm. Int J Numer Methods Eng 58(2):189–207
13. Weinstein D (2000) Scanline surfacing: building separating surfaces from planar contours. In: IEEE Visualization 2000, pp 283–289
14. Nielson GM, Franke R (1997) Computing the separating surface for segmented data. In: Proceedings of the 8th conference on visualization ‘97, Phoenix, AZ
15. Jones TR, Durand F, Desbrun M (2003) Non-iterative, feature-preserving mesh smoothing. In: ACM transactions on graphics, pp 943–949
16. Nealen A, Igarashi T, Sorkine O, Alexa M (2006) Laplacian mesh optimization. Computer graphics and interactive techniques in Australasia and South East Asia. In: Proceedings of the 4th international conference on computer graphics and interactive techniques in Australasia and Southeast Asia, Kuala Lumpur, Malaysia, pp 381–389. ISBN:1-59593-564-9
17. Taubin G (1995) A signal processing approach to fair surface design. In: Computer graphics proceedings, pp 351–358
18. Bade R, Haase J, Preim B (2006) comparison of fundamental mesh smoothing algorithms for medical surface models. In: Simulation and Visualization 2006, Magdeburg
19. Kuprat A, Khamayseh A, George D, Larkey L (2001) Volume conserving smoothing for piecewise linear curves, surfaces, and triple lines. J Comput Phys 172:99–118
20. Grabner M (2002) On-the-fly greedy mesh simplification for 2 1/2-D regular grid data acquisition systems. Vision with nontraditional sensors. In: Leberl F, Fraundorfer F (eds) Proceedings of 26th workshop of the Austrian Association for Pattern Recognition, vol 160. Austrian Computer Society, Graz
21. Lindstrom P (2000) Out-of-core simplification of large polygonal models. In: Proceedings of SIGGRAPH 2000, pp 259–262
22. Garland M, Heckbert P (1997) Surface simplification using quadric error metrics. In: SIGGRAPH 97 conference proceedings. Annual conference series, ACM SIGGRAPH. Addison-Wesley, Reading
23. Lindstrom P, Silva CT (2001) A memory insensitive technique for large model simplification. In: Proceedings of the conference on visualization ’01. IEEE Computer Society, New York, pp 121–126
24. Wu J, Kobbelt L (2003) A stream algorithm for the decimation of massive meshes. In: Graphics interface ‘03 conference proceedings, pp 185–192
25. Isenburg M, Lindstrom P, Gumhold S, Snoeyink J (2003) Large mesh simplification using processing sequences. In: Proceedings of visualization ’03, pp 465–472
26. Isenburg M, Lindstrom P (2005) Streaming meshes. In: Proceedings of visualization ’05, pp 231–238
27. Mascarenhas A, Isenburg M, Pascucci V, Snoeyink J (2004) Encoding volumetric grids for streaming isosurface extraction. In: Proceedings of 3DPVT’04, pp 665–672
28. Isenburg M, Lindstrom P, Snoeyink J (2005) Streaming compression of triangle meshes. In: Proceedings of 3rd symposium on geometry processing, pp 111–118
29. Vo H, Callahan S, Lindstrom P, Pascucci V, Silva C (2005) Streaming simplification of tetrahedral meshes. LLNL technical report UCRL-CONF-208710
30. Isenburg M, Lindstrom P, Gumhold S, Shewchuk J (2006) Streaming compression of tetrahedral volume meshes. In: Proceedings of graphics interface 2006, pp 115–121
31. Isenburg M, Gumhold S (2003) Out-of-core compression for gigantic polygon meshes. In: Proceedings of SIGGRAPH’03, pp 935–942
32. Schroeder W, Zarge J, Lorensen W (1992) Decimation of triangle meshes (Proceedings of SIGGRAPH ’92). Comput Graph 25(3)
33. Guthe M, Borodin P, Klein R (2005) Fast and accurate Hausdorff distance calculation between meshes. J WSCG 13. ISSN:1213–6964
34. Cohen J, Varshney A, Manocha D, Turk G, Weber H, Agarwal P, Brooks FP Jr, Wright W (1996) Simplification envelopes. In: Proceedings of ACM SIGGRAPH’96, pp 119–128
35. Moore RH (2007) PhD Dissertation, Carnegie Mellon University, Pittsburgh
36. Dey TK, Edelsbrunner H, Guha S, Nekhayev DV (1999) Topology preserving edge contraction. Publ Inst Math (Beograd) (N.S.) 66:23–45
37. Vivodtzev F, Bonneau GP, Letexier P (2005) Topology preserving simplification of 2D non-manifold meshes with embedded structures. Vis Comput 21(8)
38. Cutler B, Dorsey J, McMillan L (2004) Simplication and Improvement of tetrahedral models for simulation. In: Scopigno R, Zorin D (eds) Eurographics symposium on geometry processing
39. Shewchuk J (2002) What is a good linear element? Interpolation, conditioning, and quality measures. In: Eleventh international meshing roundtable Sandia National Laboratories, Ithaca, NY, pp 115–126
40. Saylor DM, Morawiec A, Rohrer GS (2003) Distribution of grain boundaries in magnesia as a function of five macroscopic parameters. Acta Mater 51:3663–3674
41. Zubal IG, Harrell CR, Smith EO, Rattner Z, Gindi GR, Hoffer PB (1994) Computerized three-dimensional segmented human anatomy. Med Phys 21(2):299–302