Engineering with Computers (2009) 25:97–114
DOI 10.1007/s00366-008-0108-z

ORIGINAL ARTICLE

Hexahedral mesh generation for biomedical models in SCIRun

Jason F. Shepherd · Chris R. Johnson

Received: 30 May 2007 / Accepted: 24 August 2007 / Published online: 27 August 2008

Abstract Biomedical simulations are often dependent on numerical approximation methods, including finite element, finite difference, and finite volume methods, to model the varied phenomena of interest. An important requirement of the numerical approximation methods above is the need to create a discrete decomposition of the model geometry into a 'mesh'. Historically, the generation of these meshes has been a critical bottleneck in efforts to efficiently generate biomedical simulations which can be utilized in understanding, planning, and diagnosing biomedical conditions. In this paper we discuss a methodology for generating hexahedral meshes for biomedical models using an algorithm implemented in the SCIRun Problem Solving Environment. The method is flexible and can be utilized to build up conformal hexahedral meshes ranging from models defined by single isosurfaces to more complex geometries with multi-surface boundaries.

1 Introduction

Advanced techniques in biomedical computing, imaging, and visualization are changing the face of biology and medicine in both research and clinical practice. The goals of biomedical computing, imaging and visualization are multifaceted. While some images and visualizations facilitate diagnosis, others help physicians plan surgery. Biomedical simulations can help to acquire a better understanding of human physiology. Still other biomedical computing and visualization techniques are used for medical training. Within biomedical research, new computational technologies allow us to "see" into and understand our bodies with unprecedented depth and detail. As a result of these advances, biomedical imaging, simulation, and visualization will help produce exciting new biomedical scientific discoveries and clinical treatments.

Biomedical simulations are dependent on numerical approximation methods, including finite element, finite difference, and finite volume methods, to model the varied phenomena of interest. An important requirement of the numerical approximation methods above is the need to create a discrete decomposition of the model geometry into a 'mesh'. The meshes produced are used as input for computational simulation, as well as, the geometric basis for which many of the visualization results are displayed. Historically, the generation of these meshes has been a critical bottleneck in efforts to efficiently generate biomedical simulations which can be utilized in understanding, planning, and diagnosing biomedical conditions.

The most common types of elements utilized in numerical approximations are triangles or quadrilaterals in two-dimensions and tetrahedral or hexahedral elements in three-dimensions. To reduce the amount of time to prepare a model, automated meshing algorithms have been developed for creating triangular, quadrilateral, and tetrahedral meshes for a very generalized class of geometries. In the case of tetrahedral meshing, algorithms are available that can generate greater than 400 thousand tetrahedra per minute [16]. However, automated hexahedral mesh generation algorithms are available for a more limited class of geometries. Because of the limited class of geometries for which hexahedral meshes can be built, a significant amount

J. F. Shepherd (✉) · C. R. Johnson
Scientific Computing and Imaging Institute,
Salt Lake City, UT, USA
e-mail: jfsheph@sci.utah.edu

C. R. Johnson
e-mail: crj@sci.utah.edu

of time in generating a hexahedral mesh is devoted to decomposing (cutting up) a model into pieces for which a known hexahedral mesh generation algorithm will succeed. The processing of geometry for creating a hexahedral mesh can take several months for a generalized model, whereas tetrahedral meshes can often be created in a matter of hours or days [30, 31].

In spite of the limited availability of an automated hexahedral mesh generation algorithm, hexahedral meshes are sometimes preferred over tetrahedral meshes in certain applications and situations for the following reasons:

1. For uniform mesh sizes, tetrahedral meshes typically require 4–10 times more elements than a hexahedral mesh to obtain the same level of accuracy [6, 29].
2. In some types of numerical approximations (i.e., high deformation structural finite element analysis with linear elements), tetrahedral elements will be mathematically 'stiffer' due to a reduced number of degrees of freedom associated with a tetrahedral element [2, 5]. This problem is also known as 'tet-locking'.

Hexahedral mesh generation can be difficult and time-consuming. In this paper, we will demonstrate a method for generating hexahedral meshes using a methodology similar to methods currently used for generating isosurfaces in volumetric image data. This algorithm utilizes a theory for hexahedral meshes outlined in [25], and implemented in the SCIRun Problem Solving Environment [22, 10, 18]. We will describe the algorithm utilized and show how to develop single surface and multi-surface hexahedral meshes. We will demonstrate several example hexahedral meshes generated with this algorithm. The generated meshes will be similar in appearance to meshes generated by hexahedral octree algorithms [20, 21, 24, 32, 33], but the method of sheet insertion presented in this paper will enable higher quality and finer control of the meshes than previously demonstrated (at the cost of automation) as well as a method for generation of conformal multi-material hexahedral meshes.

2 Background

2.1 SCIRun background

The methods discussed throughout the remainder of this paper have been developed in the SCIRun Problem Solving Environment (PSE) [22, 10, 18]. SCIRun is a problem solving environment that allows the interactive construction, debugging, and steering of large-scale, typically parallel, scientific computations. SCIRun provides a component model, based on dataflow programming, that allows various computational components and visualization components to be connected together. SCIRun can be envisioned as a "computational workbench," in which a scientist can design and modify simulations interactively via a component-based visual programming model. SCIRun enables scientists to modify geometric models and interactively change numerical parameters and boundary conditions, as well as to modify the level of mesh adaptation needed for an accurate numerical solution. As opposed to the typical "off-line" simulation mode - in which the scientist manually sets input parameters, computes results, visualizes the results via a separate visualization package, then starts again at the beginning - SCIRun "closes the loop" and allows interactive steering of the design, computation, and visualization phases of a simulation. An example biomedical simulation utilizing the SCIRun environment is shown in Fig. 1.

2.2 Algorithmic background

As imaging and scanning techniques continue to improve for applications including biomedical imaging (e.g., CT, MRI, confocal and light microscopy, etc.), geologic imaging, and mechanical scanning, there has been substantial effort placed in generating computer models that can be visualized and manipulated. Traditionally, the Marching Cubes algorithm [15] has been utilized to convert volumetric image data into isosurfaces that can be viewed and manipulated. However, while suitable for visualization, in many cases the meshes resulting from a Marching Cubes algorithm are not of sufficient quality for use with numerical techniques such as finite element, finite volume, or finite differences methods.

Since the boundary of a hexahedral mesh is a quadrilateral mesh, we can utilize a methodology which is similar to the marching cubes method [15] to create quadrilateral isosurfaces. The hexahedral theory which supports this methodology is outlined in [25] and the algorithm for generating these quadrilateral isosurfaces was implemented as a new module in SCIRun [18, 22]. For simplicity, the algorithm takes a set of triangles describing the isosurface along with a predefined hexahedral mesh that intersects the isosurface (typically a bounding box surrounding the isosurface with a regular structured mesh). The algorithm will be described in more detail in the next section.

3 Single surface methods

Because most isosurfacing algorithms generate triangle meshes to represent the isosurfaces, an algorithm which will convert the triangle surfaces to hexahedral meshes is a very useful algorithm in creating models which can be used

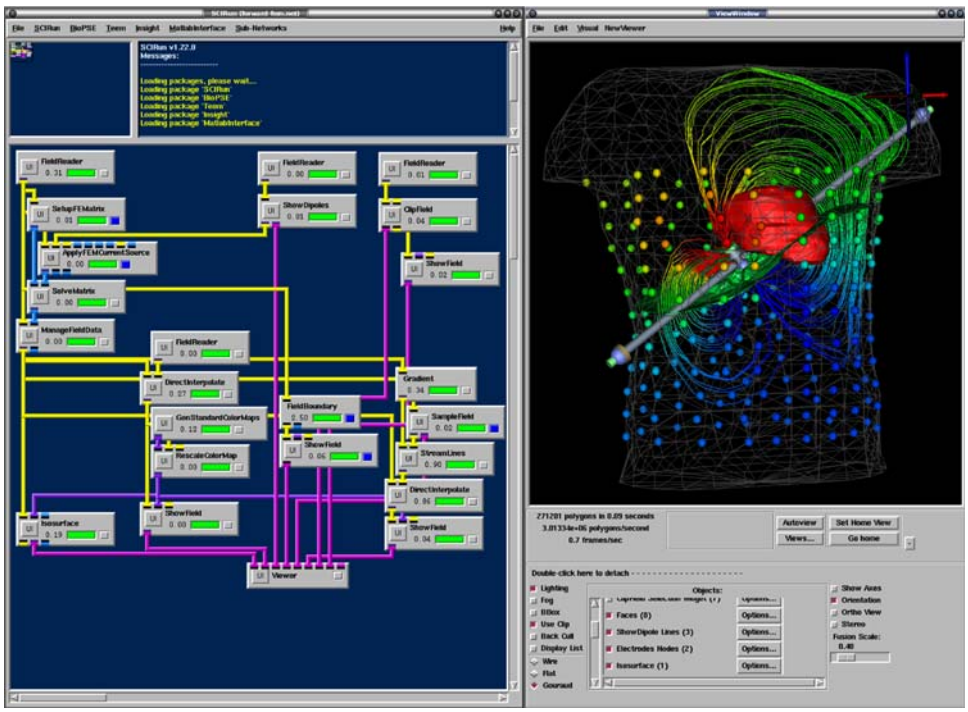

Fig. 1 The SCIRun PSE showing the module network (*middle*), the visualization window (*right*). Researchers can select UI (user interaction) buttons on many of the modules that allow control and feedback of parameters within a particular module (*left*)

in computation. Additionally, numerous algorithms exist for creating triangle meshes and most models in use for computer visualization also utilize triangle meshes making an algorithm that utilizes a triangle mesh as a starting point is readily flexible for a wide-array of preexisting models. The hexahedral meshes shown in this section were generated using the SCIRun software [18, 22] with a module (the SCIRun module name is InsertHexSheetAlongSurface) created for inserting hexahedral sheets into existing hexahedral meshes given a triangle mesh that partitions the hexahedral mesh into two regions (triangle meshes that partition the hexahedral mesh into more than two regions will be discussed in a later section). The basic algorithm takes the following form:

Given an existing hexahedral mesh and a triangle mesh representing the shape of the hexahedral sheet to be inserted, do the following:

1. *Locate all of the hexahedra that are intersected by one or more triangles in the triangle mesh*. A kdtree containing all of the triangles is utilized to improve the efficiency of this search. If there is a triangle in the vicinity of a given hexahedron, each edge of the hexahedron is tested for intersection with the triangles in the region. Each of the intersected hexes is marked as being intersected.
2. *Separate the hexahedra into three groups*: *Side1, Side2, and Intersected*. Starting with an unmarked hexahedron (i.e., a nonintersected hexahedron from the previous step), use a flood-fill algorithm to group all of the hexahedra that are connected to this hexahedron and not marked (i.e., intersected by a triangle). This group will be known as 'Side1'. All of the marked, or intersected, hexahedra are placed in a second group, known as 'Intersected', and the remaining hexahedra are placed in a third group, known as 'Side2'. An example of this process is shown in Fig. 2 where a hemispherically-shaped triangle mesh is place in a hexahedral grid. The boundary of the triangle mesh is shown in black, and the 'Intersected' hexes are drawn in yellow. 'Side1' is drawn in green and the remaining hexahedra are placed in 'Side2' (shown in blue).
3. *Collate the 'Intersected' hexahedra with either 'Side1' or 'Side2' and insert two hexahedral sheets between these two groups of hexahedra*. The 'Intersected'

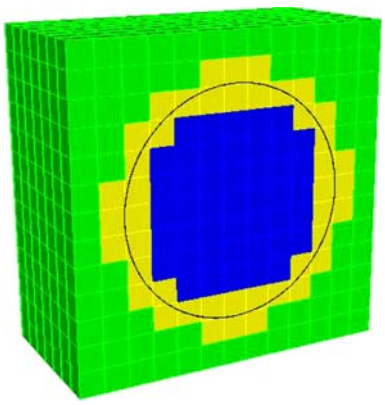

Fig. 2 A hemispherically-shaped triangle mesh (the boundary of the triangle mesh is shown in *black*) is placed in a hexahedral grid. The hexahedra intersected by the triangle mesh are shown in *yellow*, while 'Side1' is drawn in *green* and 'Side2' is shown in *blue*

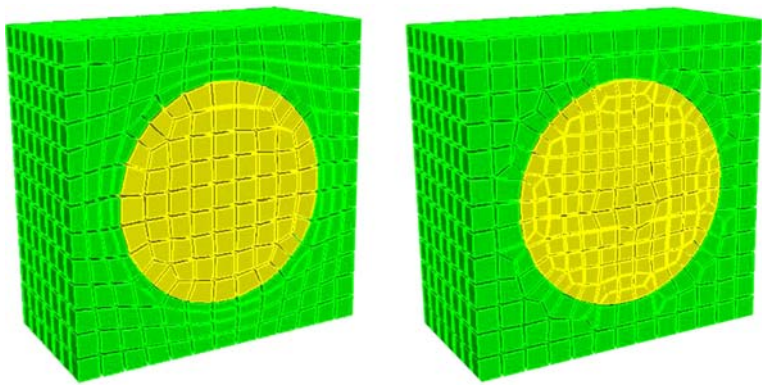

Fig. 3 Slightly different meshes result depending on which side the intersected hexes are grouped. The image on the *left* shows the resulting mesh after sheet insertion if the intersected hexes are placed with Side1's hexes, while the image on the *right* has the intersected hexes being grouped with Side2

hexahedra are subsequently added to either 'Side1' or 'Side2', and two sheets of hexahedra around these two groups. For the example highlighted in Fig. 2, depending on which side the intersected hexahedra are grouped, one of the meshes shown in Fig. 3 will result. The hexahedral sheets are inserted by (refer to Fig. 4):

a. First, determining the quadrilateral boundary between the two sides of the mesh,
b. separating the two meshes by shrinking the elements at this interface,
c. then, for each node on the separated boundary, project a new node to the triangle mesh. A map to each node is retained by both sides of the mesh, and once all of the projected nodes have been created on the boundary, the hexahedral connectivity for the two sheets can be developed by using the quadrilaterals on the interface boundary from both sides and the map to each of the newly projected nodes.

4. *Export the two new groups of hexahedra.*

The shrinking process often forces some element inversion, so it is necessary to smooth the mesh to obtain the mesh quality desired. In addition, the projection of the nodes to the triangle mesh often results in nonuniform sizing of the quadrilateral elements on the boundary. This is also remedied using a smoothing operation.

Smoothing on these meshes was accomplished in one of two ways. The first option is to use the MESQUITE [4, 17] suite of smoothing algorithms available from a mesh smoothing module implemented in SCIRun [18, 22]. These smoothers include Laplacian smoothing, a hybrid smoothing/optimization algorithm known as Smart Laplacian [8], and a mesh optimization algorithm for improving the 'shape' metric, called Shape Improvement Optimization [12]. In SCIRun, these smoothing/optimization algorithms are available for smoothing quadrilaterals or hexahedral meshes (as well as triangle and tetrahedral meshes).

The second option is to export the mesh created in SCIRun, and load it into the CUBIT Mesh Generation Toolkit [7]. CUBIT has the mesh smoothing and optimization algorithms listed above, along with some additional smoothing algorithms, including centroidal area smoothing [11], condition number optimization [12], and untangling [9, 12, 14, 28]. Additionally, CUBIT optionally allows

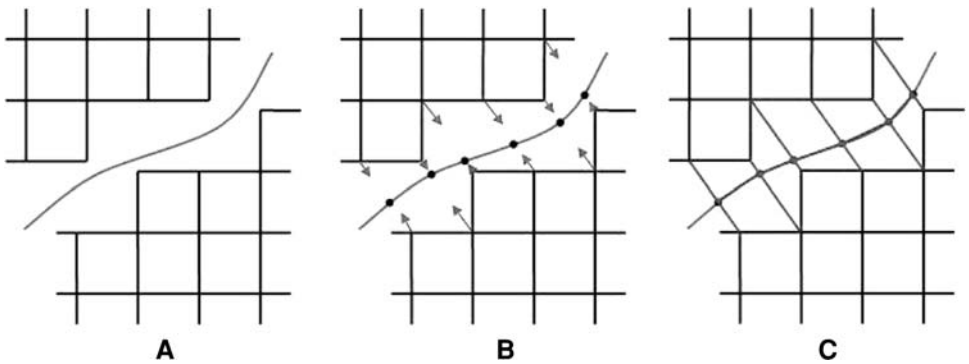

Fig. 4 Image **a** shows the shrunken hexahedra with the triangle mesh shown in between the hexahedra. Image **b** shows a newly projected node to the triangle mesh for each node on the boundary of the shrunken mesh (note that a single node on the triangle mesh corresponds to one node on each of the shrunken boundaries). Image **c** shows the newly created hexahedron by mapping the quadrilaterals on the boundary to the appropriate nodes (recently projected) on the triangle surface mesh

smoothing to occur on a focused-set of elements that can dramatically reduce the amount of time needed for optimization of specific hexahedral elements.

All mesh quality results are reported using the scaled Jacobian metric as calculated by the Verdict library of mesh quality metrics [27].

4 Single surface examples

In this section, we demonstrate the methods outlined in the previous section to generate several hexahedral meshes for models consisting of single surfaces. We demonstrate geometric conformity to the original geometry, and highlight the resulting hexahedral element quality inherent with this methodology.

4.1 Hand model

The original triangle mesh for the hand model is provided courtesy of INRIA by the AIM@SHAPE Shape Repository (http://shapes.aim-at-shape.net/index.php).

The hexahedral mesh of the hand model, shown in of Fig. 5 contains 202,974 hexahedra and was generated in SCIRun and optimized in CUBIT. The mesh was generated by using the process described in the heading to this section, namely first creating a regular grid of hexahedra that was 5% larger than a tight bounding box around the hand geometry. The size of the elements was uniform throughout the grid, and was chosen to be roughly the same size as the elements in the original triangle mesh. To obtain a sharper boundary near the wrist, the original bounding box was moved slightly to allow the original triangle mesh to extend past the boundary of the regular hexahedral grid. Two hexahedral sheets were then placed in the grid using the original triangle mesh as a guide.

Fig. 5 Front and back view of the hexahedral mesh of the hand. The mesh contains 202,974 elements

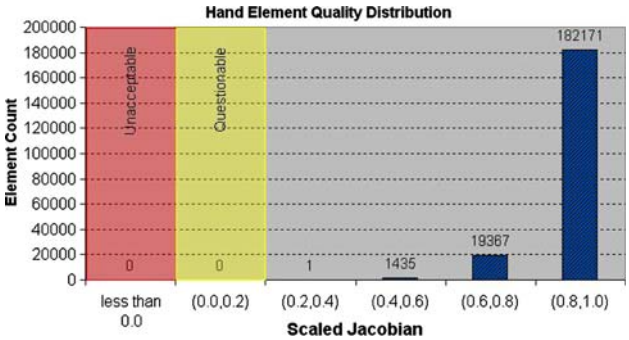

Fig. 6 Distribution of element quality for the hand model

Smoothing and optimization of this mesh was completed in CUBIT [7]. The quadrilateral mesh on the boundary was smoothed with a centroidal-area smoother to improve the quality of the surface mesh. After smoothing the quadrilaterals, the boundary nodes were fixed and the hexahedral elements were smoothed with a Laplacian smoother, followed by a mesh untangling operation on any hexahedra that may have been inverted by the Laplacian smooth. Upon completion of the untangling operation, an optimization algorithm to improve the condition number of each of the elements was performed to give the final results shown in Fig. 6.

Figure 7 displays the geometry for both the original geometry and the hexahedral mesh [the facets of the original triangle mesh are shown in red (on the left) and the facets from the hexahedral mesh are shown in green (in the middle)]. In this model, the geometric fidelity of the hexahedral mesh is very satisfactory as evidenced by the completely mottled appearance of the overlapping facets shown in the left image of both figures. The solid red area at the base of the wrist indicates the region where the triangle mesh was allowed to extend past the original hexahedral grid. Table 1 gives a listing of the original volume enclosed by the triangles and the final volume enclosed by the hexahedral mesh. The volume in the hexahedral mesh is 3.17% smaller than the volume enclosed by the original triangles. The bulk of the volume lost is due to the region at the wrist of the model where the triangle mesh was extended past the hexahedral mesh.

4.2 Mouse model

The original triangle mesh was generated from CT data and was provided courtesy of Jeroen Stinstra from the Scientific Computing and Imaging Institute at the University of Utah.

The hexahedral mesh of the mouse model, shown in of Fig. 8 contains 74,828 hexahedra and was generated in SCIRun and optimized in CUBIT. The mesh was generated first creating a regular grid of hexahedra that was larger than a tight bounding box around the mouse geometry. The

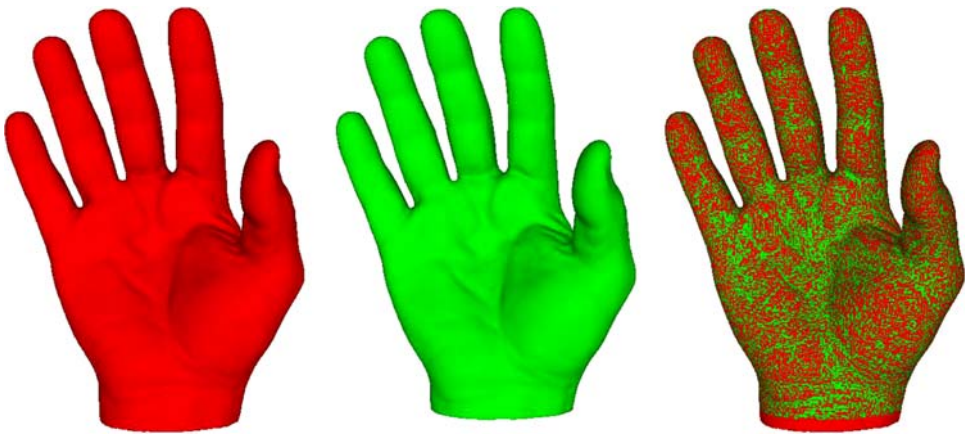

Fig. 7 Geometry generated from original triangle facets shown in *red* (on the *left*), and the geometry generated from the hexahedral facets is shown in *green* (in the *middle*). An image where both sets of facets are overlapped is given on the right to give an indication of the overall geometric fidelity of the hexahedral mesh

Table 1 Table indicating volume changes resulting from conversion of the original triangle mesh to a hexahedral mesh for the hand model

Volume (triangle mesh)	Volume (hexahedral mesh)	Difference	Percent change
0.430513	0.416846	−0.01367	−3.17%

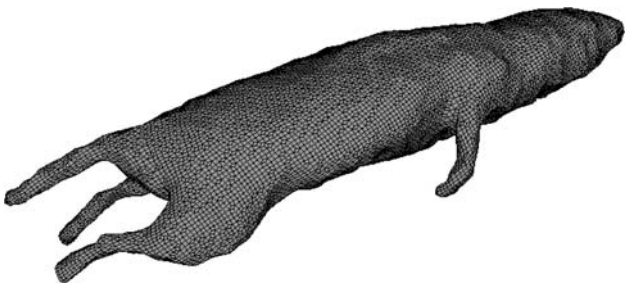

Fig. 8 Hexahedral mesh of a mouse generated from CT data. The mesh contains 74,828 elements

size of the elements within this mesh was chosen based on a percentage of length of each of the sides of the bounding box. The element size is uniform throughout the model, which is not conducive to high element quality near the feet and tail, but these locations were deemed unimportant for subsequent numerical analysis. Two hexahedral sheets were then placed in the hexahedral grid using the original triangle mesh as a guide. After the sheet insertion process, the hexahedral elements exterior to the mouse model were discarded.

Mesh quality optimization for this model was performed in CUBIT, where the quadrilateral mesh on the boundary was smoothed with a centroidal-area smoother to improve the quality of the surface mesh. Because of the nonsmooth nature in some areas of the original triangle mesh (shown in Fig. 9), some additional quadrilateral smoothing was done on some of the quadrilaterals whose nodes collected in areas of discontinuity of the original triangle mesh. After obtaining a reasonable quadrilateral mesh, the boundary

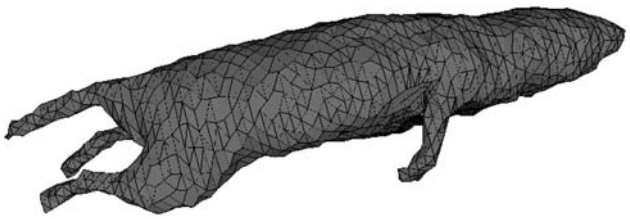

Fig. 9 Original triangle mesh of the mouse model

nodes were fixed and the hexahedral elements were smoothed with a Laplacian smoother, followed by a mesh untangling operation on any hexahedra that may have been inverted by the Laplacian smooth. Upon completion of the untangling operation, an optimization algorithm to improve the condition number of each of the elements was performed to give the final results shown in Fig. 10.

Figure 11 displays the geometry for both the original geometry and the hexahedral mesh [the facets of the original triangle mesh are shown in red (left) and the facets from the hexahedral mesh are shown in green (middle)]. In this model, the geometric fidelity of the hexahedral mesh is satisfactory with a fair amount of mottling over the entire mouse, although evidence of the original segmentation process is evident by the layering seen in the mottling. Some additional refinement around the arm may also be

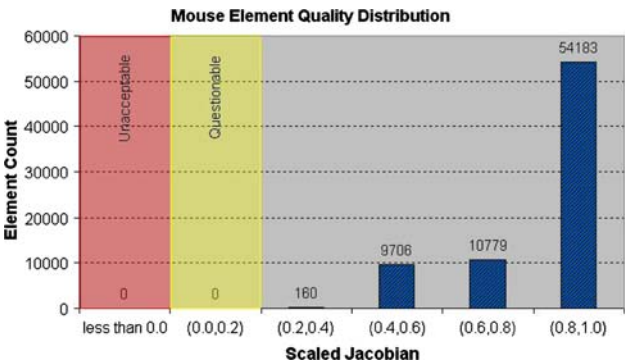

Fig. 10 Distribution of element quality for the mouse model

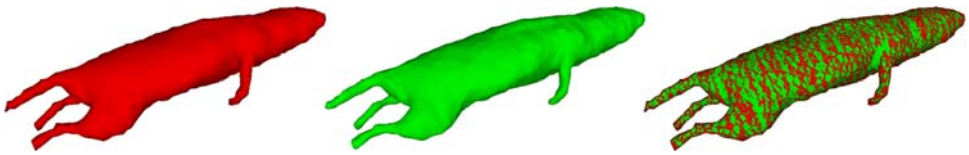

Fig. 11 Geometry generated from original triangle facets shown in *red* (*upper left*), and the geometry generated from the hexahedral facets is shown in *green* (*upper right*). An image where both sets of facets are overlapped is given on the bottom to give an indication of the overall geometric fidelity of the hexahedral mesh

Table 2 Table indicating volume changes resulting from conversion of the original triangle mesh to a hexahedral mesh for the mouse model

Volume (triangle mesh)	Volume (hexahedral mesh)	Difference	Percent change
16.845	16.873	0.028	0.17%

necessary to remove a blending in this region, as well (note the area of green surrounding the joint near the top of the arm). Additionally, utilizing a smoother initial triangle mesh, or presmoothing the triangle mesh, would enable additional geometric fidelity to be obtained. Table 2 gives a listing of the original volume enclosed by the triangles and the final volume enclosed by the hexahedral mesh. The volume in the hexahedral mesh is 0.17% larger than the volume enclosed by the original triangles with the additional volume gained mainly in the concave regions near the arms and legs.

4.3 Bunny model

The original triangle mesh for the bunny model was generated by John Schreiner using the 'afront' software [19].

The hexahedral mesh of the bunny model, shown in Fig. 12 contains 125,183 hexahedra and was generated in SCIRun and optimized in CUBIT. The mesh was generated first creating a regular grid of hexahedra that was 5% larger than a tight bounding box around the bunny geometry. The size of the elements within this mesh was chosen using a size from the original triangle mesh in an area with a moderate amount of detail. Two hexahedral sheets were then placed in the hexahedral grid using the original triangle mesh as a guide, and the mesh was then exported from SCIRun and translated into a file format readable by CUBIT.

In CUBIT, centroidal-area smoothing was used on the quadrilateral boundary. After smoothing the quadrilateral mesh, the boundary nodes were fixed and the hexahedral elements were smoothed with a Laplacian smoother, followed by a mesh untangling operation on any hexahedra that may have been inverted by the Laplacian smooth. Upon completion of the untangling operation, an optimization algorithm to improve the condition number of each

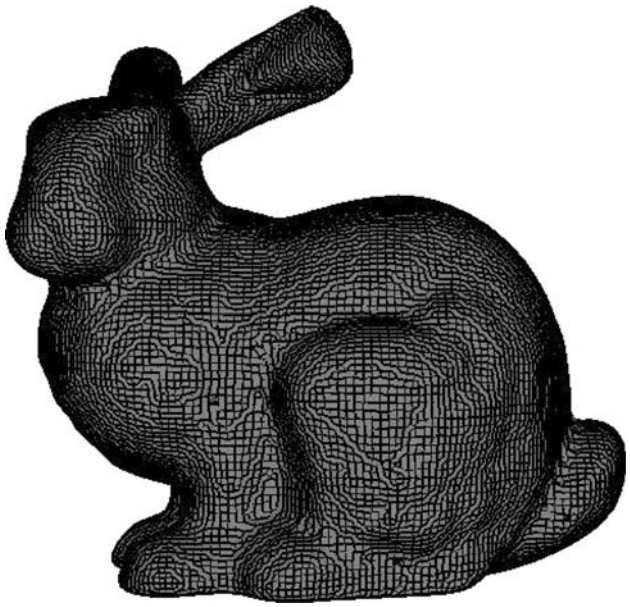

Fig. 12 Hexahedral mesh of the bunny model, containing 125,183 elements

of the elements was performed to give the final results shown in Fig. 13.

Figure 14 displays the geometry derived from the facets of the original triangle mesh and the hexahedral mesh [the facets of the original triangle mesh are shown in red (left) and the facets from the hexahedral mesh are shown in green (middle)]. In this model, the geometric fidelity of the hexahedral mesh is reasonable with a fair amount of mottling over the entire bunny, although evidence of blending

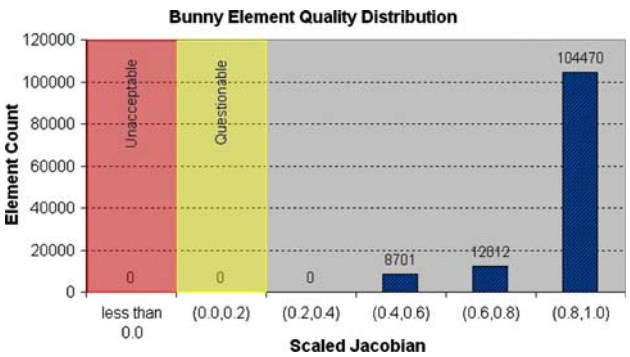

Fig. 13 Distribution of element quality for the bunny model

Fig. 14 Geometry generated from original triangle facets shown in *red* (*upper left*), and the geometry generated from the hexahedral facets is shown in *green* (*upper right*). An image where both sets of facets are overlapped is given at the bottom to give an indication of the overall geometric fidelity of the hexahedral mesh

Table 3 Table indicating volume changes resulting from conversion of the original triangle mesh to a hexahedral mesh for the bunny model

Volume (triangle mesh)	Volume (hexahedral mesh)	Difference	Percent change
753507.7	753918.2	410.5	0.05%

by the hexahedral elements is evident in areas of higher curvature, specifically around the neck, tail, ear, and thigh of the rabbit. Some refinement of the original grid in these regions should improve the geometric fidelity of the hexahedral mesh. Table 3 gives a listing of the original volume enclosed by the triangles and the final volume enclosed by the hexahedral mesh. The volume in the hexahedral mesh is 0.05% larger than the volume enclosed by the original triangles with the additional volume gained mainly in the concave regions near the legs, tail and neck.

4.4 Dragon model

The original triangle mesh for the dragon model was generated by John Schreiner using the 'afront' software [19].

The hexahedral mesh of the dragon model, shown in two separate images in Fig. 15 contains 465,527 hexahedra and was generated in SCIRun and optimized in CUBIT. The mesh was generated first creating a regular grid of hexahedra that was 5% larger than a tight bounding box around the dragon geometry. The size of the elements within this mesh was chosen using a comparable size from the original triangle mesh in an area with a moderate amount of detail (the original trimesh of the dragon is shown in Fig. 16). This size was made uniform throughout the original hexahedral grid.

Two hexahedral sheets were then placed in the hexahedral grid using the original triangle mesh as a guide. After placement of the new sheets, the hexahedral elements exterior to the dragon model were discarded.

Mesh optimization for the mesh was performed in CUBIT using a centroidal-area smoothing on the boundary, followed by Laplacian smoothing for all interior nodes was performed. This was followed by a mesh untangling operation on any hexahedra that may have been inverted by the Laplacian smooth, and, finally, mesh optimization to improve the condition number of each of the elements was performed to give the final results shown in Fig. 17.

Figure 18 displays the geometry derived from the facets of the original triangle and hexahedral meshes [the facets of the original triangle mesh are shown in red (on the left) and the facets from the hexahedral mesh are shown in green (in the middle)]. In this model, the geometric fidelity of the hexahedral mesh is reasonable over the entire model as demonstrated by a fair amount of mottling over the entire dragon. Some blending of the hexahedral mesh over original detail in the triangle mesh is evident in areas of high concavity, specifically the solid green areas near the joints around the legs, feet, and face, as well as along the

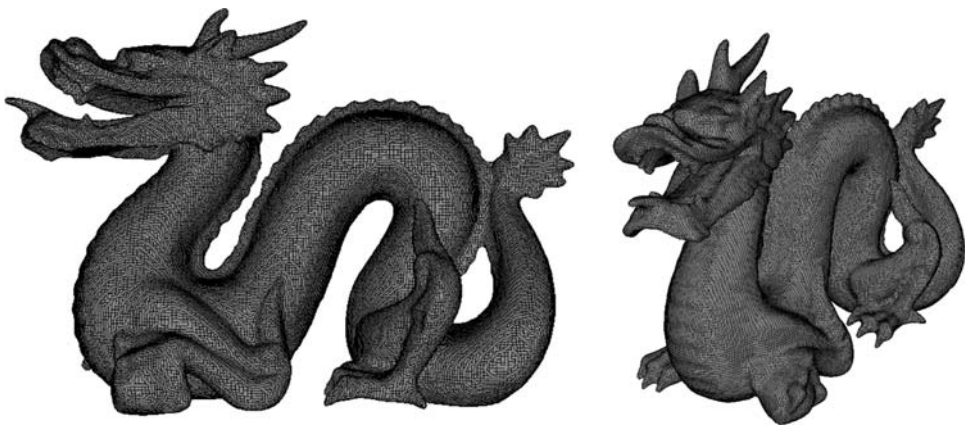

Fig. 15 Two views of the hexahedral mesh of the dragon model. The mesh contains 465,527 elements

Fig. 16 Original triangle mesh of the dragon model

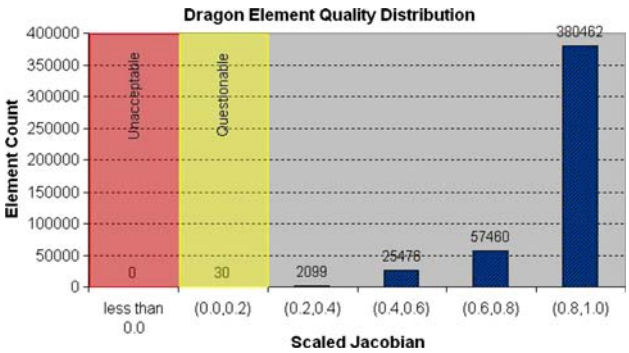

Fig. 17 Distribution of element quality for the dragon model

scales along the back of the dragon. Some of these artifacts may be due to the grouping of the intersected hexes discussed in the heading of this section. To improve the geometric fidelity of the hexahedral mesh in regions of high convexity, the hexahedra that were intersected by the triangle mesh were added to the group of hexahedra located in the interior of the triangle mesh to capture additional geometric detail in the horns, facial fans, and teeth. Including all the intersected hexes with the hexes interior to the dragon enables an improved mesh that better captures high convexity details, but can be deleterious to some features in areas of high concavity. While this process did not greatly impact the resulting mesh, improvements to this algorithm can be made that may improve the overall geometric fidelity of the model, as well as reduce the amount of questionable hexahedral elements in the final model.

Table 4 gives a listing of the original volume enclosed by the triangles and the final volume enclosed by the hexahedral mesh. The volume in the hexahedral mesh is 0.16% larger than the volume enclosed by the original triangles with the additional volume being gained largely in the concave regions around the leg joints, mouth, and along the spines on the back of the dragon.

4.5 Brain model

The original triangle mesh for the brain model is provided courtesy of INRIA by the AIM@SHAPE Shape Repository (http://shapes.aim-at-shape.net/index.php).

The hexahedral mesh of the brain model, shown in Fig. 19 contains 644,221 hexahedra and was generated in SCIRun and optimized in CUBIT. The mesh was generated first creating a regular grid of hexahedra that was 5% larger than a tight bounding box around the brain geometry. A uniform element size was chosen using a comparable size from the original triangle mesh. Two hexahedral sheets were then placed in the hexahedral grid using the original triangle mesh (shown in Fig. 20) as a guide, and the mesh was then imported in CUBIT for mesh optimization.

In CUBIT, the quadrilateral mesh on the boundary was smoothed with a centroidal-area smoother to improve the quality of the surface mesh. After smoothing the quadrilateral mesh, the boundary nodes were fixed and the hexahedral elements were smoothed with a Laplacian smoother, followed by a mesh untangling operation on any hexahedra that may have been inverted by the Laplacian smooth. Upon completion of the untangling operation, an optimization algorithm to improve the condition number of each of the elements was performed to give the final results shown in Fig. 21.

Table 5 gives a listing of the original volume enclosed by the triangles and the final volume enclosed by the hexahedral mesh. The volume in the hexahedral mesh is

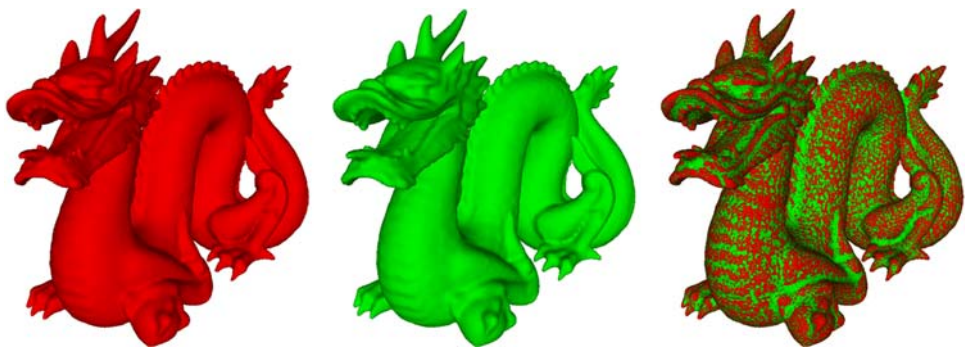

Fig. 18 Geometry generated from original triangle facets shown in *red* (on the *left*), and the geometry generated from the hexahedral facets is shown in *green* (in the *middle*). An image where both sets of facets are overlapped is given on the right to give an indication of the overall geometric fidelity of the hexahedral mesh

Table 4 Table indicating volume changes resulting from conversion of the original triangle mesh to a hexahedral mesh for the dragon model

Volume (triangle mesh)	Volume (hexahedral mesh)	Difference	Percent change
11140235.34	11158459.36	18224.02	0.16%

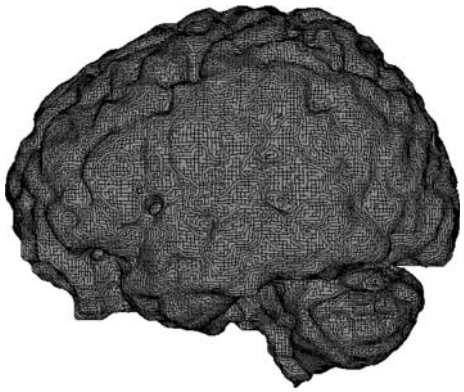

Fig. 19 Hexahedral mesh of the brain model, containing 644,221 elements

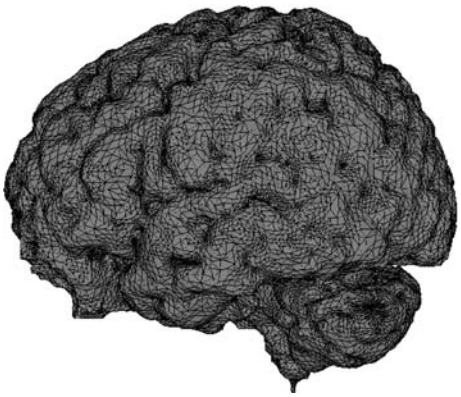

Fig. 20 Original triangle mesh of the brain model

1.85% larger than the volume enclosed by the original triangles. The additional volume gain is noticeable in areas of concavity of the original triangle mesh with the bulk of the additional volume being gained in the loss of internal cavities in the brain.

Figure 22 displays the geometry derived from the facets of the original triangle and hexahedral meshes [the facets of the original triangle mesh are shown in red (on the left) and the facets from the hexahedral mesh are shown in green (in the middle)]. In this model, the geometric fidelity of the hexahedral mesh is reasonable over the entire model as demonstrated by a fair amount of mottling over the entire brain. However, loss of detail is apparent in many of the brain folds and especially in some of the interior

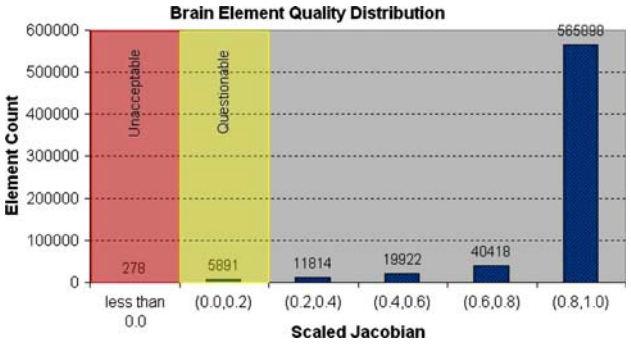

Fig. 21 Distribution of element quality for the brain model

Table 5 Table indicating volume changes resulting from conversion of the original triangle mesh to a hexahedral mesh for the brain model

Volume (triangle mesh)	Volume (hexahedral mesh)	Difference	Percent change
67855.98	69109.39	1253.41	1.85%

structure of the brain as evident in Fig. 23. The loss of the internal cavities accounts for the bulk of the volume gain shown in Table 5.

The loss of detail and negative element quality can be attributed to a couple of basic assumptions made in the current sheet insertion algorithm. First, the assumption is made that the shrunken hexahedral grid is directly homeomorphic to the original triangle mesh. With proper element sizing, this assumption is reasonable assuming that nonmanifold connections between elements do not exist in the groups of elements around which the sheets are inserted (i.e., regions where the boundary of the group of elements being pillowed is 'pinched' together). The algorithm implemented in SCIRun detects locations where nonmanifold edges exist, but does not detect nonmanifold nodes. Normally, decreasing the size of the mesh will improve the geometric fidelity and remove many, if not all, of the nonmanifold nodes. However, decreasing the size comes at a cost, and because we utilized uniform sizing and based on the the number of elements in the original grid adding additional elements to the mesh was not an option without increasing the amount of memory on the machine generating the meshes. In the brain mesh, the element sizing utilized resulted in 150 nonmanifold nodes that account for nearly all of the negative Jacobian elements in the resulting mesh (shown in Fig. 24).

A second assumption was that the intersected hexahedra should all go to one side or the other. As discussed in the dragon example, depending on which side the group of intersected hexes is added, dramatic improvements in geometric fidelity can be realized. Because the isomorphism to the triangle mesh is only requirement, better

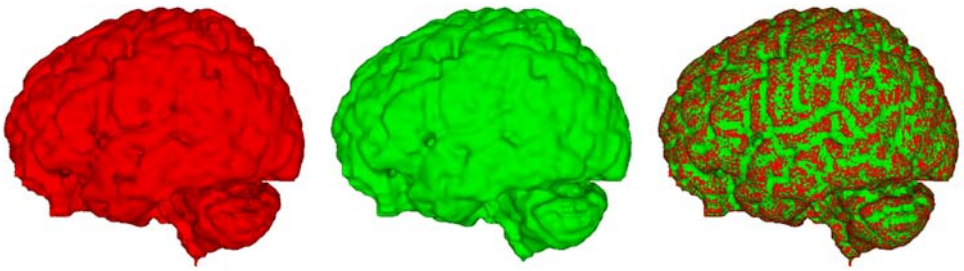

Fig. 22 Geometry generated from original triangle facets shown in *red* (on the *left*), and the geometry generated from the hexahedral facets is shown in *green* (in the *middle*). An image where both sets of facets are overlapped is given on the right to give an indication of the overall geometric fidelity of the hexahedral mesh

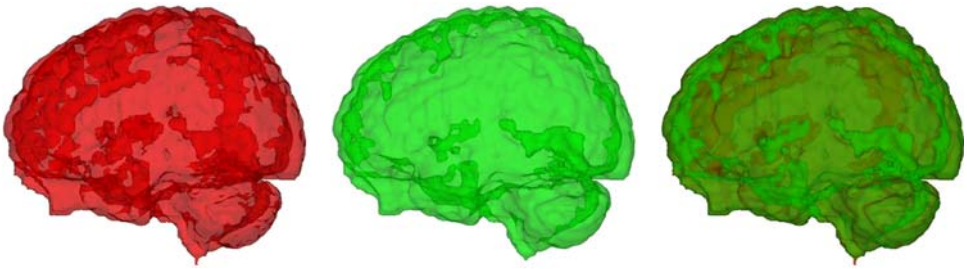

Fig. 23 Transparent view to show internal structure of the geometry generated from original triangle facets shown in *red* (on the *left*), and the geometry generated from the hexahedral facets is shown in *green* (in the *middle*). An image where both sets of facets are overlapped is given on the right to give an indication of the overall geometric fidelity of the hexahedral mesh

separation of the intersected hexes between groups may be a more viable solution, especially if regions of high convexity and concavity can be distinguished from the original triangles.

5 Multisurface methods

In the previous section, we demonstrated hexahedral meshing of complex geometric solids that are defined by a single surface. In this section, we demonstrate hexahedral mesh creation on geometric solids that are bounded by more than one surface using a similar methodology as used earlier.

In solid modeling, a volume is bounded by one or more surfaces and a surface is bounded by zero or more curves. A volume that is bounded by a single surface with zero curves can be meshed using an isosurfacing methodology as described in the previous section. In this section, we will focus on volumes that are bounded by more than one surface, which in turn is bounded by one or more curves. We will consider these curves to be discontinuities in the boundary of the mesh. These curves can be categorized as follows (see Fig. 25):

Definition: A ***soft curve*** on a volume is a curve on the boundary of the volume where the transition from one surface to the next surface across the curve is smooth, or nearly smooth.

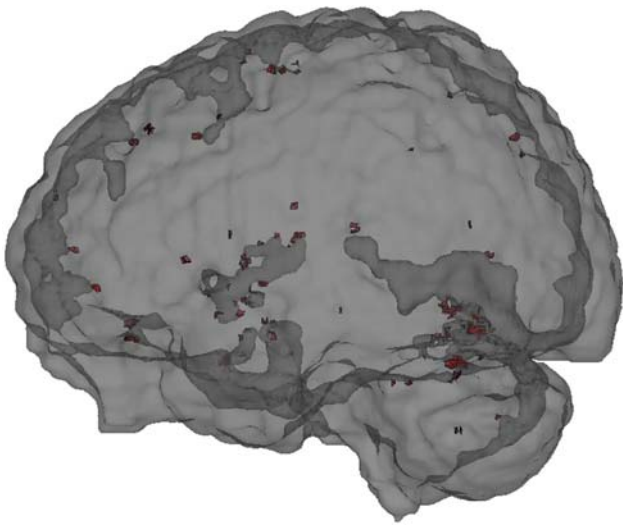

Fig. 24 Locations of negative scaled Jacobian elements in the brain model

Definition: A ***hard curve*** on a volume is a curve on the boundary of the volume, where the transition from one surface to the next surface across the curve is not smooth.

These definitions are somewhat ambiguous, and it is left to the reader to determine when a transition is smooth versus when the transition is not smooth. The techniques presented in this section will be general enough to account

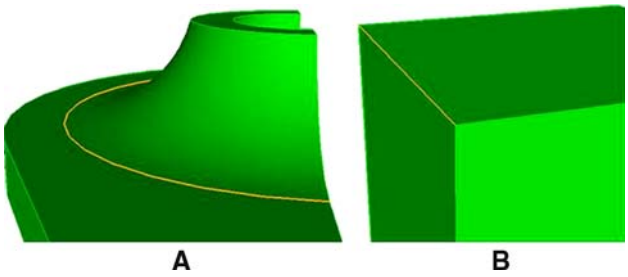

Fig. 25 The highlighted curve in the image on the left could be considered a soft curve, while the highlighted curve in the image on the right would be considered a hard curve

for this ambiguity; however, the examples presented in this section will, for the most part, ignore the cases of curves that can be defined nearly unambiguously as being 'soft curves' on the boundary of the geometry.

5.1 Sharp feature capture

Whenever a hard curve is present in the boundary of a volume, it is advantageous to the quality of the mesh and to the fidelity of the geometry to have a string of mesh edges that align themselves with the curve. In a hexahedral mesh, a string of mesh edges results whenever two sheets intersect. We can control the placement of the edges resulting from the intersection of the two sheets by controlling the locations of the sheet intersections. For example, in Fig. 26 we place a planar sheet behind the face in the head model. The intersection of the planar sheet (which also captures a new planar surface) with the boundary sheet inserted earlier, produces a string of mesh edges that are nicely aligned to create the sharp *corner*. This allows the face to be *cut* from the head model.

By controlling where the sheet intersections occur, or manipulating the conformation of the sheets such that intersections occur in the proximity of hard curves, we can manipulate the hexahedral mesh to obtain a mesh topology that mimics the geometric topology with the hard curves. Therefore, we can use this methodology to enable hexahedral meshing of multisurface geometries by strategic insertion of the fundamental sheets needed to capture the geometric surfaces, curves and vertices of the orginal model.

Additionally, by inserting multiple sheets (similar to the procedure used for isosurfacing), we can construct complex geometries by performing *Boolean*-like operations in the hexahedral mesh while still maintaining conformity with all of the split-off pieces. In Fig. 27 we demonstrate several successive spherical cuts from a single hexahedral mesh of a cubical geometry. Where two sheets intersect, the result is a string of mesh edges that align with the cut enabling the sharp features in the resulting model to be recognized. Figure 28 lists the resulting quality of each of the elements demonstrating the overall high quality of the resulting mesh.

In SCIRun [18, 22], we utilize the same algorithm that was developed for the hexahedral isosurfacing (as described previously) to affect the sheet insertion process. By ensuring that another sheet already exists in the location where we desire the hard curve to be placed, the addition of the new sheet results in a string of mesh edges that can be moved to the location of the hard curve. A similar methodology is used in the MeshCutting algorithm [3] as implemented in CUBIT.

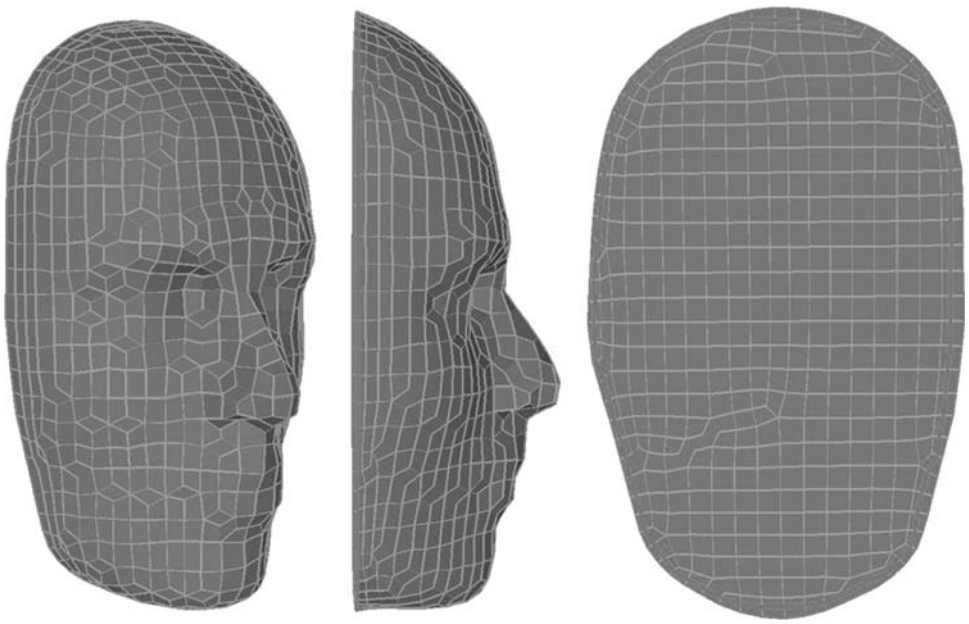

Fig. 26 In locations where two sheets intersect, the resulting mesh topology contains a string of edges that can be aligned with sharp features, or hard curves. In this image, we 'cut' the face from the head model by inserting a planar sheet behind the face. The inserted boundary sheet capturing the face (shown in the image on the *right*), along with the newly inserted planar sheet (*middle image*) behind the face, results in a mesh topology that contains a string of edges sufficient to produce a sharp boundary where the two sheets intersect, as shown in the middle image above

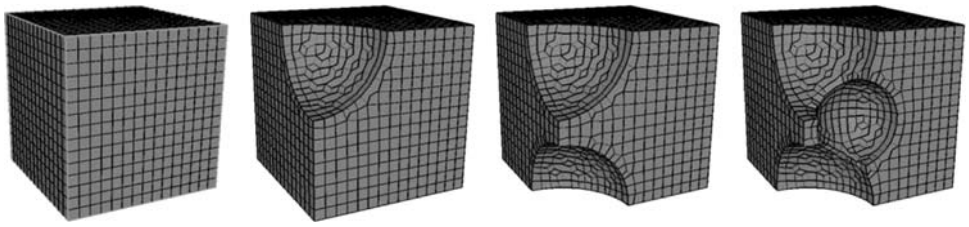

Fig. 27 By inserting spherical sheets into the geometry, we can perform Boolean-like operations in the mesh, while maintaining the integrity of the hexahedral mesh. At each of the boundary surfaces, the intersection of the spherical sheet with the original planar sheets in the cuboid mesh is sufficient to produce a hexahedral mesh with a string of mesh edges that can be utilized to capture the boundary discontinuities resulting from the spherical cuts

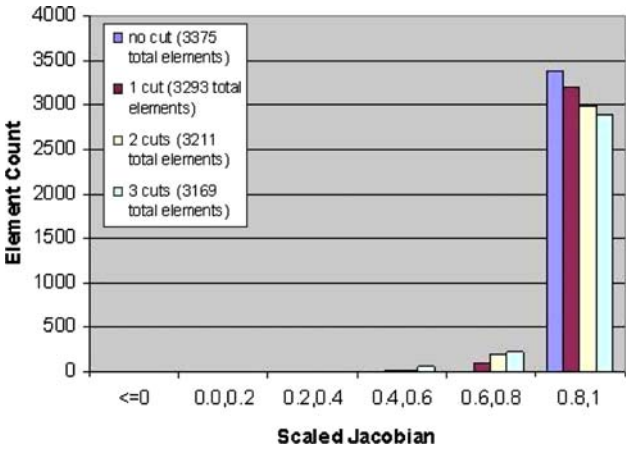

Fig. 28 The distribution of scaled Jacobian values for the cuboid geometry with the spherical cutouts shown in Fig. 27

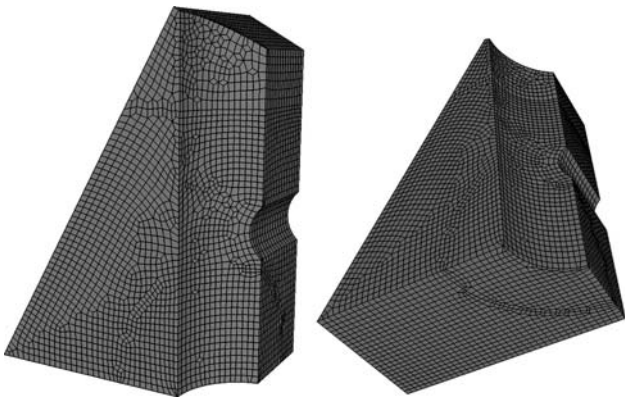

Fig. 29 Hexahedral mesh of the mechanical part model showing images from the side and bottom of the mesh

The remainder of this section demonstrates hexahedral meshing on several multisurface models. These results were obtained using both the algorithms in SCIRun and CUBIT, with the exact recipe for generating the mesh being detailed in each respective section. The first example of a mechanical part demonstrates the sheet insertion process for capturing both hard and soft curves. Later examples demonstrate increased model complexity utilizing sheet insertion and mesh cutting examples as implemented in both SCIRun and CUBIT.

6 Multisurface examples

6.1 Mechanical part

The hexahedral mesh of the mechanical part model, shown in Fig. 29, was generated in CUBIT and contains 27,486 hexahedra. The original geometry contains two soft curves that are shown in Fig. 30. In this example, we will show how these soft curves can be captured utilizing the sheet insertion techniques described earlier, while also generating a hexahedral mesh for the volume.

The difficulty in generating a hexahedral mesh on this model using traditional methods is a result of the long quarter-cylindrical cut along one of the edges of the model coupled with the quarter-circle soft curve on the base of the model. These two features of the model prevent the use of a traditional sweeping method. Other common methods for producing a hexahedral mesh with a fair amount of structure require a fair amount of decomposition to the model to develop recognizable hexahedral mesh primitives.

To generate the mesh for the mechanical part, we first generated a mesh of a simpler version of the model that could be meshed utilizing a sweeping algorithm [23, 26] (see Image a in Fig. 31). Utilizing this mesh, we inserted three additional sets of sheets into the mesh using the three triangle meshes as guides shown in Image b, c, and d in Fig. 31. These triangle meshes were created directly on the original surfaces using an advancing front triangle meshing algorithm available in CUBIT. Following the insertion of the sheets, the inserted sheets from the triangle mesh in Image B was used to cut the original geometry to produce the final geometry for the mechanical part and recover the hard curves around the cylindrical section.

Following sheet insertion, the new mesh edges that were formed to capture the soft curves were fixed in place and a centroidal area smoothing of the boundary was performed

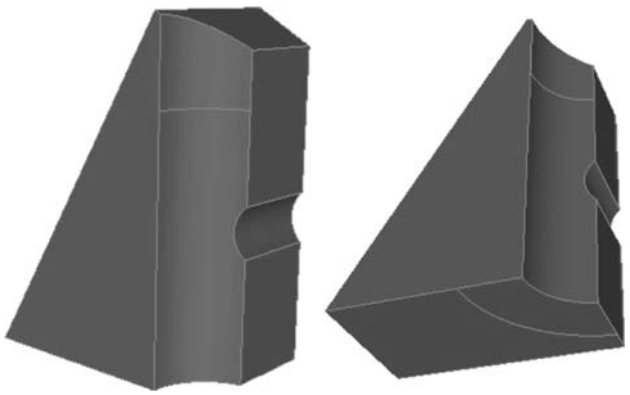

Fig. 30 Geometry for the mechanical part showing two soft curves: one in the upper cylindrical section and a second on the base of the model

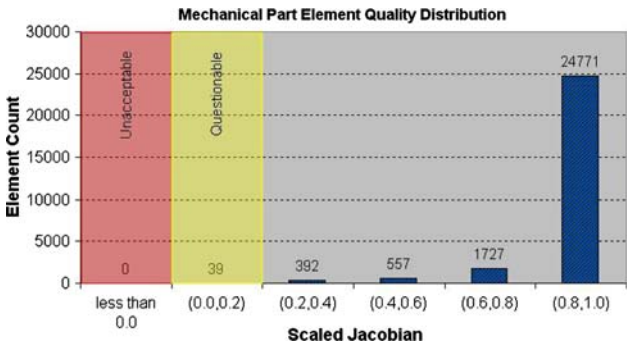

Fig. 32 Distribution of element quality for the mechanical part model

to improve quality of the surface mesh for the solid. The hexahedral elements were then smoothed with a Laplacian smoother followed by optimization via the mean-ratio metric for each of the hexahedra resulting in the mesh quality distribution shown in Fig. 32

6.2 Skull model

The skull model is provided courtesy of INRIA by the AIM@SHAPE Shape Repository (http://shapes.aim-at-shape.net/index.php). The difficulty in generating this model with traditional methods is several fold. First, the original model was constructed from a triangle mesh only, and no solid model description of this model is available. Therefore traditional decomposition with solid modeling operations is not readily accessible. Second, since there are no hard curves in the model traditional methods for determining a decomposition strategy for common hexahedral methods are not present. With commonly available methods for generating hexahedral meshes, this model would be extremely difficult to produce.

The hexahedral mesh of the skull model, shown in Fig. 33, was generated in SCIRun and contains 19,330 hexahedra in the skull bone and an additional 34,815 hexahedra in the mesh of the cranial cavity. The mesh is completely conformal throughout the model, but is separated into the two material blocks. A transparent view of the geometry showing the bone and cranial cavity is given in Fig. 34.

This model was generated by placing a triangle mesh describing the geometry of the skull bone (minus the surface describing the cranial cavity) in a regular grid of hexahedra and inserting two hexahedral sheets using the triangle mesh to guide the placement of the newly formed hexahedra. The mesh exterior to the skull was discarded, and an additional set of sheets was added using a triangle mesh describing the cranial cavity to control the placement of the new hexahedral elements belonging to the inserted sheets. This generation process is shown pictorially in Fig. 35.

These two groups of hexahedral elements were then optimized in CUBIT, by first, using a centroidal-area smoother on the exterior skull surface and the shared surface of the cranial cavity. Laplacian smoothing was then utilized on the hexahedra in both volumes. Additional mesh untangling and condition number optimization were performed on the hexahedra in the hexahedral mesh of the bone. The final mesh quality, dictated by the scaled Jacobian metric, is shown in the distribution in Fig. 36.

6.3 Goose16 model

The goose16 model is provided courtesy of ANSYS [1]. The most significant difficulty in generating a hexahedral mesh of the goose model using traditional hexahedral algorithms is the circularity created in the geometry. This

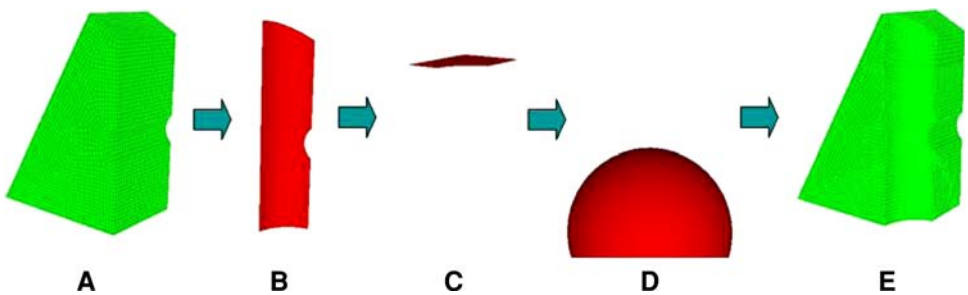

Fig. 31 Flow chart showing the sheet insertion steps to create the mesh for the mechanical part. The *red* surfaces represent hexahedral sheets that were inserted into the simplified hexahedral mesh on the left to create the final hexahedral mesh on the right

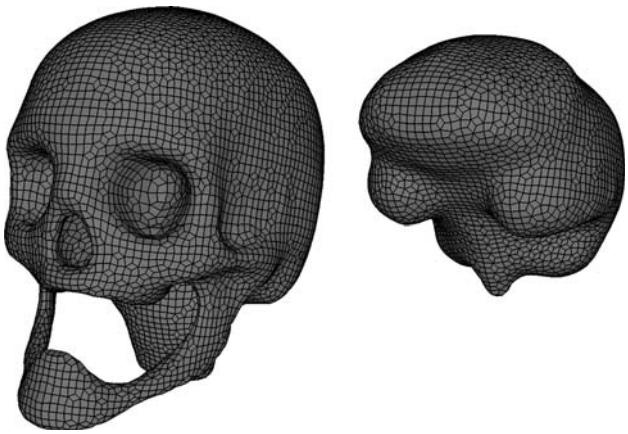

Fig. 33 Hexahedral mesh of the skull model. Bone (*left*) and cranial cavity (*right*) meshes are shown separately

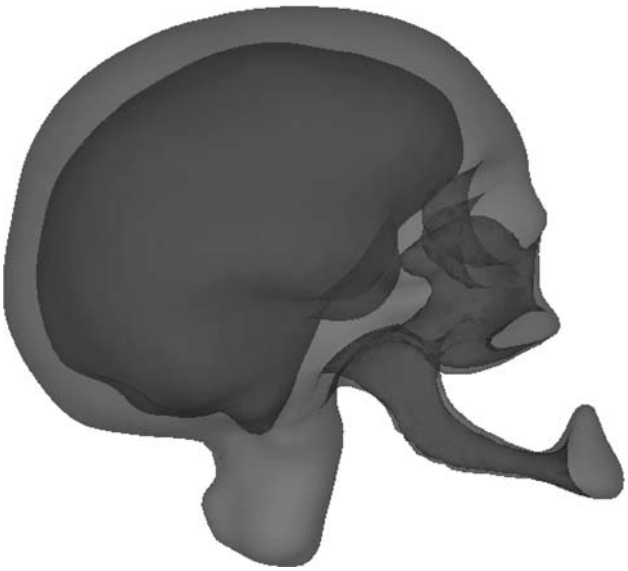

Fig. 34 Transparent view of the combined geometry generated from the facets of the hexahedral mesh of the skull model

prohibits common hexahedral mesh generation methods without doing a fair amount of precision decomposition to the original model. Additionally, the branching of the cylindrical cut-outs in the back make traditional sweeping methods impossible to use on this model without creating degenerate hexahedra at the branch points.

The hexahedral mesh of the goose16 model, shown in Fig. 37, was generated using sheet insertion algorithms in SCIRun and mesh cutting algorithms in CUBIT. A close-up view detailing the internal pipe detail is shown in Fig. 38. The final mesh contains 57,114 hexahedra. The mesh quality distribution of scaled Jacobian measures for the goose16 mesh is shown in Fig. 39.

The mesh was generated following the process flow shown in Fig. 40. Specifically, a simplified geometry that was sweepable was created and meshed in CUBIT. A set of sheets was created and inserted into this mesh to fit the geometry of the original model using a triangle mesh of this surface as a guide in placing the hexahedral sheets. The elements outside the original volume boundaries described by the newly inserted sheet were discarded.

On the backside of the goose model, a set of half-cylindrical sheets following the pipe-like arm created in the previous step was added to this mesh with the elements interior to this cylinder being discarded. An additional half-cylindrical branch to the previous sheets was added to this, discarding the elements interior to this set of sheets, resulting in the mesh shown in Image H of Fig. 40.

Finally, one last set of sheets was added near the top of the model to produce the filleted region and finalize the mesh. Because this last set of sheets was nearly tangent with the top of the surface near the middle of the model, a line of doublet elements resulted where the two sheets meet. This line of doublets was resolved with a boundary face collapse operation [25] to join the two disjoint sheets into a single continuous sheet across the top of the model.

The resulting mesh was smoothed using a Laplacian smoother on the boundary quadrilaterals and hexahedral elements, followed by a mesh untangling operation and condition number optimization. The resulting mesh quality distribution of scaled Jacobian measures is shown in Fig. 39.

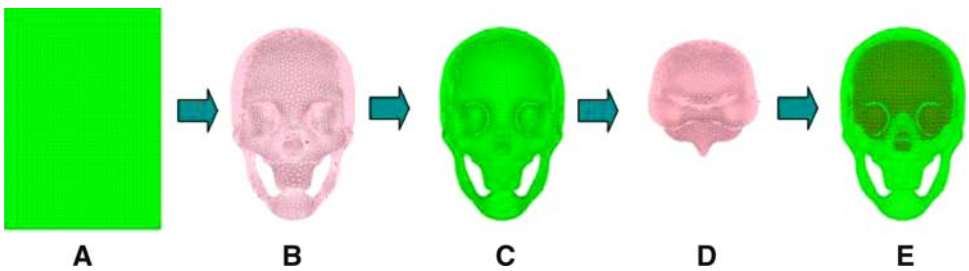

Fig. 35 Pictorial flow chart demonstrating the mesh generation process for creating the hexahedral mesh of the skull. Triangle meshes (*pink*) are utilized to guide placement of hexahedral sheets into existing hexahedral meshes to achieve new meshes that are conformal with the original solid geometry

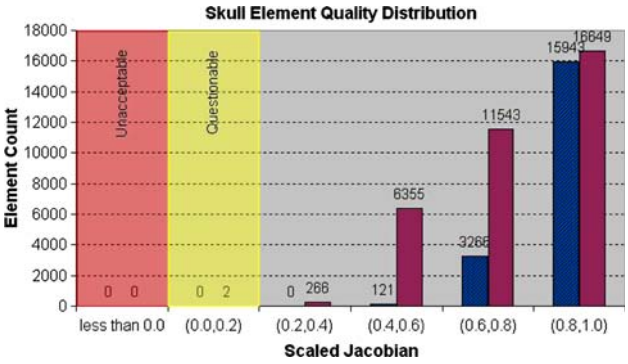

Fig. 36 Distribution of element quality for the skull model (bone is shown in *white* and cranial cavity is shown in *black*)

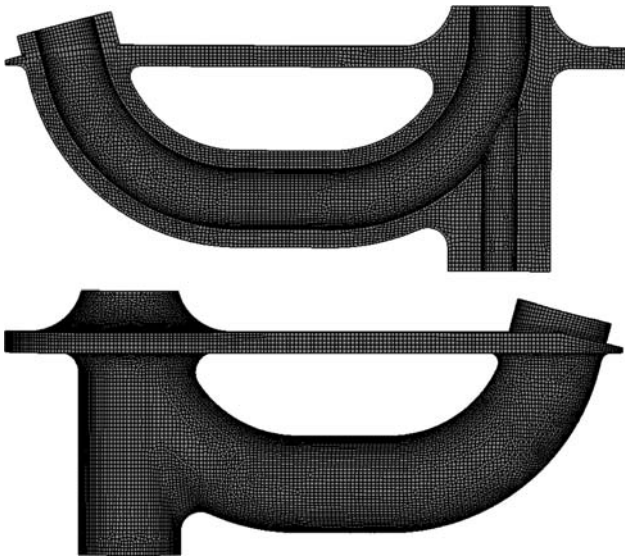

Fig. 37 Hexahedral mesh of the goose16 model showing images from the front and back of the mesh, respectively

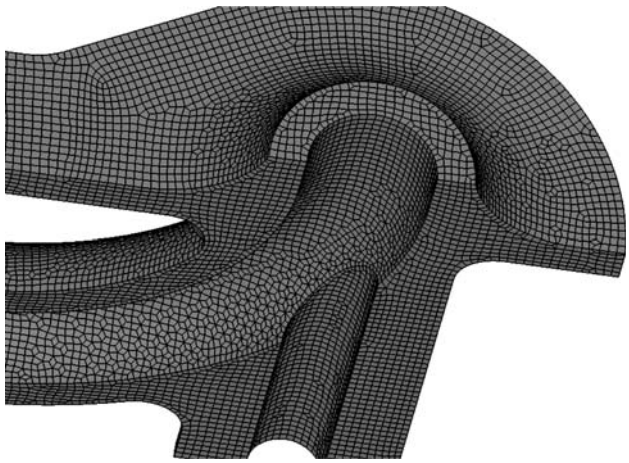

Fig. 38 Close-up view of the goose16 model showing detail in the area of several cuts

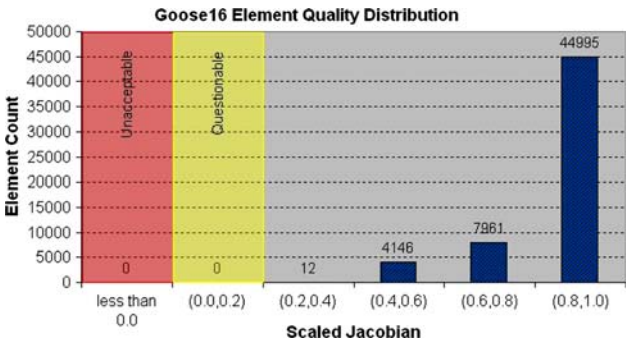

Fig. 39 Distribution of element quality for the goose16 model

7 Conclusion

Biomedical simulations are often dependent on numerical approximation methods, including finite element, finite difference, and finite volume methods, to model the varied phenomena of interest. Meshes are used as input for computational simulation, as well as, the geometric basis for which many of the visualization results are displayed. Historically, the generation of these meshes has been a critical bottleneck in efforts to efficiently generate biomedical simulations which can be utilized in understanding, planning, and diagnosing biomedical conditions.

For some types of analyses, hexahedral meshes are desirable for reduced element counts and improved analysis fidelity. However, automated hexahedral mesh generation algorithms are available for a more limited class of geometries. Because of the limited class of geometries for which hexahedral meshes can be built, a significant amount of time is often required to generating a hexahedral mesh. For many models the process of creating a hexahedral mesh can take several months for a generalized model [30, 31].

In this paper, we have built upon theory outlined in [25] to create an algorithm that can be utilized for generating hexahedral meshes for biomedical models. The algorithm uses a framework that is similar to Marching Cubes approaches for generating triangle isosurfaces. The algorithm is also flexible and can be utilized to build up more complex geometries while maintaining geometric fidelity and mesh quality. We demonstrated this flexibility by creating meshes for biomedical and mechanical models of increasing complexity. All of the results were verified using the Verdict mesh quality library [27] and the scaled Jacobian measure for a hexahedral element and shown to have acceptable quality suitable for use in subsequent simulations.

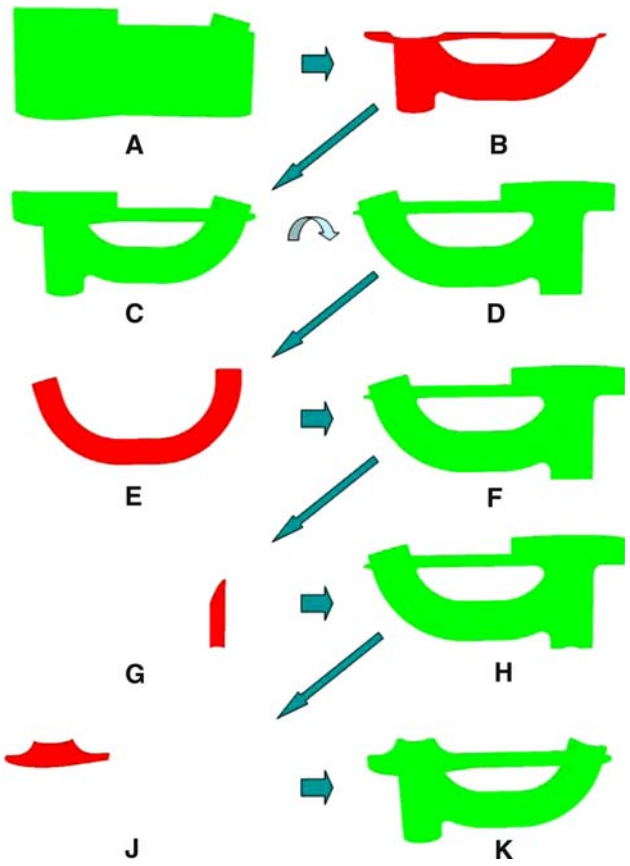

Fig. 40 Pictorial flow chart demonstrating the mesh generation process for creating the hexahedral mesh of the goose16 model. Triangle meshes (*red*) are utilized to guide placement of hexahedral sheets into the existing hexahedral mesh to achieve a new mesh that is conformal with the original solid geometry

References

1. ANSYS (2007) ANSYS, http://www.ansys.com
2. Benzley SE, Perry E, Merkley K, Clark B (1995) A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In: Proceedings, 4th international meshing roundtable. Sandia National Laboratories, pp 179–191
3. Borden MJ, Shepherd JF, Benzley SE (2002) Mesh cutting: fitting simple all-hexahedral meshes to complex geometries. In: Proceedings, 8th international society of grid generation conference
4. Brewer M, Freitag-Diachin L, Knupp P, Leurent T, Melander DJ (2003) The MESQUITE mesh quality improvement toolkit. In: Proceedings, 12th international meshing roundtable. Sandia National Laboratories, pp 239–250
5. Bussler ML, Ramesh A (1993) The eight-node hexahedral elements in FEA of part designs. In: Foundry management and technology, pp 26–28
6. Cifuentes AO, Kalbag A (1992) A performance study of tetrahedral and hexahedral elements in 3-D finite element structural analysis. Finite Elem Anal Des 12(3–4):313–318
7. The CUBIT geometry and mesh generation toolkit (2007) Sandia National Laboratories, http://cubit.sandia.gov/
8. Freitag L (1997) On combining Laplacian and optimization-based mesh smoothing techniques. AMD trends in unstructured mesh generation. ASME 220:37–43
9. Freitag LA, Plassmann P (2000) Local optimization-based simplicial mesh untangling and improvement. Int J Numer Methods Eng 49(1):109–125
10. Johnson C, MacLeod R, Parker S, Weinstein D (2004) Biomedical computing and visualization software environments. Commun ACM 47(11):64–71
11. Jones TR, Durand F, Desbrun M (2003) Non-iterative, feature-preserving mesh smoothing. ACM Trans Graph 22(3): 943–949
12. Knupp P, Mitchell SA (1999) Integration of mesh optimization with 3D all-hex mesh generation, LDRD subcase 3504340000, final report. SAND 99-2852, October 1999
13. Knupp PM (2003) Hexahedral and tetrahedral mesh shape optimization. Int J Numer Methods Eng 58(1):319–332
14. Knupp PM (2000) Hexahedral mesh untangling and algebraic mesh quality metrics. In: Proceedings, 9th international meshing roundtable. Sandia National Laboratories, pp 173–183
15. Lorenson WE, Cline HE (1987) Marching cubes: a high resolution 3D surface construction algorithm. Comput Graph 21(4):163–169. Proceedings of SIGGRAPH '87
16. Loriot M (2006) TetMesh-GHS3D v3.1 the fast, reliable, high quality tetrahedral mesh generator and optimiser, http://www.simulog.fr/mesh/tetmesh3p1d-wp.pdf
17. MESQUITE (2005) The mesh quality improvement toolkit, terascale simulation tools and technology center (TSTT), http://www.tstt-scidac.org/research/mesquite.html
18. Parker S, Weinstein D, Johnson C (1997) The SCIRun computational steering software system. In: Arge E, Bruaset A, Langtangen H (eds) Modern software tools in scientific computing. Birkhauser Press, Boston, pp 1–40
19. Scheidegger C, Schreiner J (2007) Afront, http://sourceforge.net/projects/afront/
20. Schneiders R (1996) A grid-based algorithm for the generation of hexahedral element meshes. Eng Comput 12:168–177
21. Schneiders R (1997) An algorithm for the generation of hexahedral element meshes based on an octree technique. In: Proceedings, 6th international meshing roundtable. Sandia National Laboratories, pp 183–194
22. SCIRun (2007) A scientific computing problem solving environment, scientific computing and imaging institute (SCI), download from http://software.sci.utah.edu/scirun.html
23. Scott MA, Earp MN, Benzley SE, Stephenson MB (2005) Adaptive sweeping techniques. In: Proceedings, 14th international meshing roundtable. Sandia National Laboratories, pp 417–432
24. Shephard MS, Georges MK (1991) Three-dimensional mesh generation by finite octree technique. Int J Numer Methods Eng 32:709–749
25. Shepherd JF (2007) Topologic and geometric constraint-based hexahedral mesh generation. Published Doctoral Dissertation, University of Utah
26. Shepherd JF, Mitchell SA, Knupp P, White DR (2000) Methods for multisweep automation. In: Proceedings, 9th international meshing roundtable. Sandia National Laboratories, pp 77–87
27. The verdict mesh verification library (2007) Sandia National Laboratories, http://cubit.sandia.gov/verdict.html
28. Vachal P, Garimella RV, Shashkov MJ (2002) Mesh untangling. LAU-UR-02-7271, T-7 Summer Report 2002
29. Weingarten VI (1994) The controversy over hex or tet meshing. Machine Design, pp 74–78, April 18
30. White DR, Leland RW, Saigal S, Owen SJ (2001) The meshing complexity of a solid: an introduction. In: Proceedings, 10th international meshing roundtable. Sandia National Laboratories, pp 373–384

31. White DR, Saigal S, Owen SJ (2003) Meshing complexity of single part CAD models. In: Proceedings, 12th international meshing roundtable. Sandia National Laboratories, pp 121–134
32. Yerry MA, Shephard MS (1984) Three-dimensional mesh generation by modified octree technique. Int J Numer Methods Eng 20:1965–1990
33. Zhang Y, Bajaj C (2005) Adaptive and quality quadrilateral/hexahedral meshing from volumetric imaging data. In: Proceedings, 13th international meshing roundtable. Sandia National Laboratories, pp 365–376