TOPOLOGIC AND GEOMETRIC CONSTRAINT-BASED HEXAHEDRAL MESH GENERATION

by

Jason F. Shepherd

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

School of Computing

The University of Utah

May 2007

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

SUPERVISORY COMMITTEE APPROVAL

of a dissertation submitted by

Jason F. Shepherd

This dissertation has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

	Chair: Christopher R. Johnson
	Claudio T. Silva
	R. Michael Kirby
	Krzysztof Sikorski
	Lori Freitag Diachin

THE UNIVERSITY OF UTAH GRADUATE SCHOOL

FINAL READING APPROVAL

To the Graduate Council of the University of Utah:

I have read the dissertation of Jason F. Shepherd in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

Date

Christopher R. Johnson
Chair, Supervisory Committee

Approved for the Major Department

Martin Berzins
Chair/Dean

Approved for the Graduate Council

David S. Chapman
Dean of The Graduate School

ABSTRACT

Hexahedral finite element meshes have historically offered some mathematical benefit over tetrahedral finite element meshes in terms of reduced error and smaller element counts, especially with respect to finite element analyses within highly elastic, and plastic, structural domains. However, because hexahedral finite element mesh generation often requires significant geometric decomposition, generating hexahedral meshes can be extremely difficult to perform and automate and the process often takes several orders of magnitude longer in time to complete than current methods for generating tetrahedral meshes. In this dissertation, we focus on delineating known constraints associated with hexahedral meshes and formulating these constraints utilizing the dual of the hexahedral mesh. Utilizing these constraints, we show that hexahedral mesh generation can be viewed as an optimization problem. We review existing hexahedral algorithms and describe how these algorithms operate to satisfy the hexahedral mesh generation constraints. The concept of a fundamental hexahedral mesh will be introduced and it will be shown how the fundamental mesh relates to a minimal hexahedral mesh for a given geometry. We will demonstrate conversion of existing hexahedral meshes to fundamental hexahedral meshes using hexahedral flipping operations to convert boundary sheets to fundamental sheets. Building on existing algorithms for generating hexahedral meshes from volumetric image data, we will show significant improvement in hexahedral mesh quality through the introduction of a single fundamental sheet into hexahedral meshes generated from isosurfacing techniques. We will outline a method for constructing hexahedral meshes where all hexahedra are convex and have positive volume utilizing triangle meshes of manifold surfaces to guide the placement of fundamental sheets into an existing hexahedral mesh. Finally, we demonstrate construction of hexahedral meshes for multi-surface geometric solids by introducing multiple fundamental sheets to satisfy the hexahedral mesh generation constraints for the geometric solid.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

A dissertation is rarely the work of a single individual, although a single individual often gets the credit for these efforts. My case is no exception and I wish to thank many individuals for their specific counsel and guidance as I've struggled to complete this work. With specific thanks to the following individuals:

- Special thanks to Chris Johnson, my advisor, for his efforts in helping me get from the beginning to the end, and his thoughtful counsel, insights, and concern as we met together throughout the last couple of years.
- Sandia National Laboratories for funding and maintaining my employment while I took the time to complete this dissertation.
- Marty Cole, Jeroen Stinstra, and Michael Callahan for their help in developing tools and giving me insight into SCIRun that I needed to complete some of this research. Special thanks to Marty for the many hours he has spent brainstorming, debugging, and offering advice in the various SCIRun coding projects that I have tried to complete.
- John Schreiner, Claudio Silva, and Carlos Scheidegger for their algorithmic expertise in developing a simple and robust algorithm for separating elements. Also, special thanks to John Schreiner for periodically generating new, and extremely good, triangle meshes for use in my testing.
- Tatsuhiko Suzuki and Shigeo Takahashi who allowed me to co-author a paper with them early in my research which provided me with tremendous insight into hexahedral mesh quality. Special thanks to Tatsuhiko who is one of the most thoughtful researchers with whom I have worked.
- Scott Mitchell and Pat Knupp who have been wonderful mentors for the last several years and have been good friends even while I've been away from Albuquerque. Scott has continually offered new insights and contructive feedback on my efforts, and I can't say enough about the smoothing algorithms that Pat has developed in CUBIT.
- Carlos Carbonera and I stumbled together by happenstance in what has proved to

be one of the most beneficial (and enjoyable) collaborations in which I have been involved. His mathematical and theoretical approach to hexahedral meshing has been extremely enlightening to me. Developing several proofs that demonstrate the topological existence of hexahedral meshes was 'icing on the cake' in working with him.

- Karl Merkley, Ray Meyers, Randy Morris, and Mike Stephenson for their continued support, confidence, and encouragement of this research. A very special thanks to Karl for his mentorship and also for helping me to refine my thinking in a way that is more communicable and succinct.
- Harold and Kathryn Smith for their love and support and help on the home front that has helped to give me some freedom and confidence to pursue some of these efforts.
- Bruce and Sydnee Shepherd, my parents, for last minute baby sitting and Sunday dinners.
- Samantha, Ben, and Tyler who make it a joy to come home at night.
- And, most especially to my wife, Stacey, for her love, support and management of the details while I attempted to be a student again. (I love you.)

CHAPTER 1

INTRODUCTION

Numerical techniques, such as finite element, finite difference, and finite volume methods, are used to model various scientific and engineering phenomena for a wide variety of disciplines, including structural mechanics, dynamics, heat transfer, and computational fluid dynamics. Because of the flexibility of these methods, the problem set to which these methods are applied is growing and expanding daily, and is currently being utilized in fields as diverse in scale as cellular microbiology and quantum chromodynamics to star and galaxy formation studies (for example, see [58, 63, 59]).

An important requirement of the numerical approximations above is the need to create a discrete decomposition of the model geometry into a 'mesh' composed of smaller elements. The meshes produced are used for computational simulation, as well as the geometric basis for visualization of results.

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 [56]. 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, the amount of time devoted to decomposing (cutting up) a model into pieces for which a known hexahedral mesh generation algorithm will succeed is significant. 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 [114, 115].

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. Tetrahedral meshes typically require 4-10 times more elements than a hexahedral mesh to obtain the same level of accuracy [111, 19].

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 [5, 16]. This problem is also known as 'tet-locking'.

Generating hexahedral meshes is often difficult and time-consuming, and obtaining a quality hexahedral mesh (i.e., a hexahedral mesh where all hexahedra are convex and have positive volume) is even more challenging. In this dissertation, we will show that hexahedral mesh generation is equivalent to a pseudo-linear programming optimization and focus on delineating the underlying criteria that must be satisfied (i.e., constraints) in order to produce a hexahedral mesh for a given geometric model. Specifically, this dissertation will show progress on three distinct goals related to hexahedral mesh generation. These goals are:

1. Define the necessary constraints for producing quality hexahedral meshes in arbitrary solids.

2. Define the basic geometric and topologic structures needed to create a hexahedral mesh in an arbitrary solid, and demonstrate minimization of hexahedral meshes by manipulating these structures.

3. Demonstrate all-hexahedral mesh generation for a collection of arbitrary solids of increasing complexity.

The remainder of this chapter will discuss these goals and outline the remainder of the dissertation.

1.1 Goal: Necessary Constraints

Define the necessary constraints for producing quality hexahedral meshes in arbitrary solids.

Practical algorithms for generating hexahedral meshes are often limited to a subclass of geometric solids to which the algorithm can be applied. Currently, the geometric classes

to which a hexahedral algorithm can be applied are so small that excessive decomposition of the solid is required until each of the subvolumes of the original solid is recognizable as a primitive object for which a mesh template can be constructed. Proofs have been constructed by Mitchell [64] and Eppstein [27] indicating that topologies for hexahedral meshes exist in a very general class of volumes provided some very minor conditions are met (i.e., the solid's boundary is meshed with an even parity of quadrilaterals). While it has been demonstrated that topological constraints can be met given these conditions [29, 18], the resulting meshes have no guarantees on the resulting hexahedral quality (i.e., some of the elements may be concave or may be inverted) making the algorithms unusable in a practical sense. Developing a full list of the topological, geometric, quality and algorithmic constraints necessary to produce a useable (i.e., all elements must be convex and have positive volume) hexahedral mesh is needed. Given a full list of constraints, it will be possible to evaluate existing algorithms to determine what assumptions have been made that limit the algorithms applicability to a wider class of solid geometries.

Chapter 2 of this dissertation will outline a list of necessary constraints for generating hexahedral meshes. This list will utilize constraints developed by others and also explicitly define new constraints. Both primal and dual forms of a hexahedral mesh will be discussed. Chapter 3 will review existing hexahedral mesh generation algorithms and provide discussion regarding how specific algorithmic paradigms strongly enforce some of the listed constraints while weakly enforcing, or ignoring, other constraints. We will also discuss how the algorithmic paradigm often prevents flexibility in enforcing constraints resulting in limits on the classes of geometric objects that can be meshed by that algorithm.

1.2 Goal: Fundamental Meshes

Define the basic geometric and topologic structures needed to create a hexahedral mesh in an arbitrary solid, and demonstrate minimization of hexahedral meshes by manipulating these structures.

Given a list of the necessary and sufficient constraints for producing a hexahedral mesh in an arbitrary solid, it can be inferred that there exists a minimal set of hexahedral elements (i.e., a hexahedral mesh which conforms to the original geometry but has the fewest number of hexahedral elements necessary to conform to the geometry) that will satisfy these conditions to fill the geometric solid. Identification of the neces-

sary structures needed to create a hexahedral mesh for a given solid might enable the creation of algorithms for automated decomposition, hexahedral mesh coarsening, and simplification of existing algorithms. Additionally, given an initial coarse mesh of an object, very efficient algorithms for producing isotropic and anisotropic meshes could be developed using existing refinement algorithms [104, 38] that might be better suited for high-performance computing and parallelization.

Chapter 4 will discuss the concept of a 'fundamental' hexahedral mesh for an arbitrary geometric solid. We will demonstrate techniques for converting an existing hexahedral mesh to a fundamental hexahedral mesh, and will also discuss the relationship of the fundamental hexahedral mesh to the minimal mesh for a given geometric solid.

1.3 Goal: Hexahedral Mesh Generation

Demonstrate hexahedral mesh generation for a collection of arbitrary solids of increasing complexity.

This goal is broken into two distinct sections. First, a wide array of geometric models have been generated using isosurfacing techniques. In Chapter 5, an algorithm will be demonstrated for generating hexahedral meshes utilizing isosurfaces. Secondly, we will demonstrate mesh generation for models whose boundaries are not defined by isosurfaces, but rather by a collection of connected surfaces. In both sections, the algorithms will work to resolve the constraints outlined in Chapter 2 with insights developed from the fundamental mesh concepts detailed in Chapter 4.

CHAPTER 2

HEXAHEDRAL MESH GENERATION CONSTRAINTS

Numerous algorithms exist for producing hexahedral meshes [75]. However, no one algorithm is provably successful at generating hexahedral meshes in a wide class of geometric objects. In this chapter, we will define terminology related to geometry and meshes that will be utilized throughout the remainder of the dissertation. We will also review some basic terminology from linear programming and optimization, and then recast hexahedral mesh generation as a pseudo-optimization problem. We will define and outline hexahedral mesh generation constraints in both the primal space and the dual space.

2.1 Solid Geometry

The goal of mesh generation is to convert a given geometric representation into an alternative geometric representation that can be readily utilized in a variety of computational methods. Because there can be a variety of original geometric representations (including point sets, distance fields, facet representations, functional representations, etc.), for simplicity we will utilize a solid geometry representation in our discussions. We will attempt to keep the definitions for a solid geometry as general as possible in an effort to make it feasible to convert from one geometric representation to a solid geometric representation.

For clarity, we need to define what is meant by a solid geometry. A solid geometry consists of five main entity types, namely vertices, curves, surfaces, solids (or volumes), and collections of volumes (these may be referred to as an 'assembly'). These entities are often arranged in a hierarchical structure, as shown in Table 2.1. This hierarchical structure is often referred to as the topology of the solid geometry.

Definition: *Geometric topology* is the terminology used to described how geometric entities are connected to other geometric entities. *Mesh topology* is the terminology used

Table 2.1. Hierarchical arrangement of geometric entities.

Geometric Entity	Bounding Entities
Vertex	None
Curve	Zero or two vertices
Surface	Zero or more curves
Volume	One or more surfaces

to describe how mesh entities are connected with other mesh entities.

There are certainly special cases to the hierarchy shown in the table (i.e., curves without endpoints, surfaces with zero area, etc.). Most of these special cases can be ignored, or remedied by introducing the necessary entities to match the definitions shown in the table and minimally affecting the solid geometry representation.

The solid geometry also defines a 'space'. Each of the geometric entities described above may reside in an $\Re^3$ space. A volume consists of a closed set of surfaces. Therefore, a volume also divides space into two (or more) distinct regions, or subspaces, within the space of their domain (i.e., a closed volume separates $\Re^3$ space into two distinct regions: the space on one side of the volume (inside) as opposed to the space on the other side of the volume (outside)). A subspace can be geometrically defined by a boundary composed of 0-dimensional vertices, 1-dimensional curves, and 2-dimensional surfaces(residing in $\Re^3$). Therefore, in $\Re^3$ a volume can be utilized to describe a 'sub-space' of a given space, and the sub-spaces defined by a collection of volumes taken together can also be utilized to define a larger space in $\Re^3$.

Because a mesh is an alternative representation of the original geometry, it is necessary to define the various mesh entities and their correspondence to the original geometric entities. In essence, a mesh is a discretization of a space into a collection of entities that approximate the space defined by the original geometric representation. We will utilize the following terminology, shown in Table 2.2, in order to distinguish between mesh entities and geometric entities. From this table, we can also note the hierarchical relationship inherent in the mesh entities, which is similar to the relationship between geometric entities noted earlier.

2.2 The 'Primal' of a Hexahedral Mesh

Since a mesh is an alternative and approximate geometric representation of a solid geometry, there is a need to impose some requirements to ensure that the features in the

Table 2.2. Relationships between geometric entities and mesh entities.

Dimensionality	Geometric Entity	Mesh Entity
0D	Vertex	Node
1D	Curve	Edge
2D	Surface	Quadrilateral (or Triangle, etc.)
3D	Volume	Hexahedron (or Tetrahedron, etc.)

solid geometry are present in the mesh. In optimization theory, the term 'constraints' is utilized to denote the requirements that must be satisfied for the solution to be taken as acceptable. We will be utilizing several similar optimization paradigms and definitions (specifically from linear programming), and need to define some of this terminology here.

A linear program is an optimization problem that is stated in terms of an objective function and a set of linear constraints. The objective function is minimized (or maximized) subject to the given set of linear constraints. The constraints are the boundary of a feasible region of solutions from which an optimal solution has to be found (see Figure 2.1).

We can pose a hexahedral mesh generation problem as a pseudo-optimization problem utilizing a similar format to a linear programming problem and outlining an objective function and constraints that must be satisfied to generate a feasible alternate geometric representation to the original geometry. For clarity, we will separate the constraints into

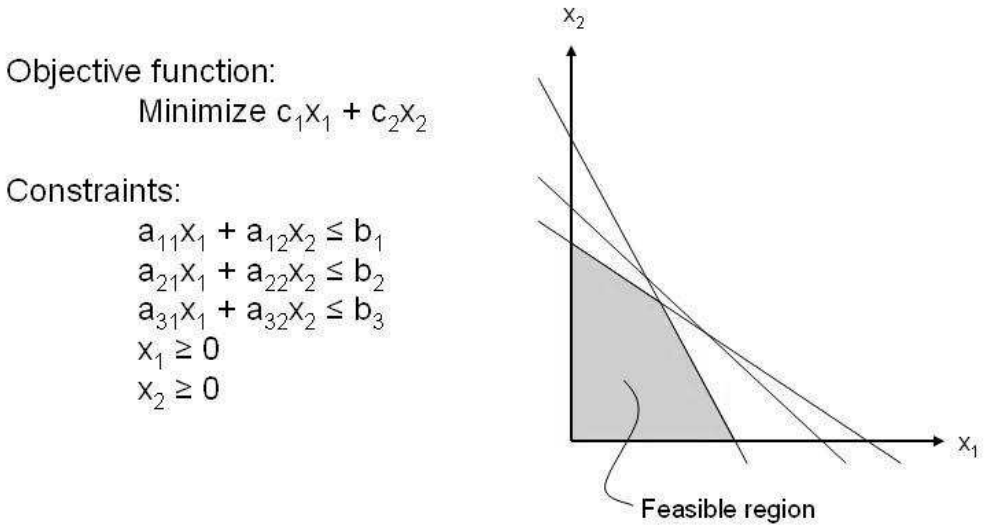

Figure 2.1. Example format of a linear programming optimization problem.

three categories: topologic, boundary, and geometric (or quality) constraints.

2.2.1 Topologic Constraints in the Primal Mesh

The topologic constraints define how the previously described mesh entities in a hexahedral mesh are connected to one another. For our purposes, a hexahedral mesh **M = {N, E, Q, H}** is defined as a geometric cell complex composed of 0-dimensional nodes **N**, 1-dimensional edges **E**, 2-dimensional quadrilaterals **Q** (residing in $\Re^3$), and 3-dimensional hexahedra **H**, such that

A. Each node is contained by at least two edges.

B. Each edge contains two distinct (i.e., different) nodes.

C. If two edges contain the same nodes, the edges are identical.

D. Each quadrilateral is bounded by a cycle of four distinct edges.

E. Two quadrilaterals share at most one edge.

F. If two quadrilaterals share four edges, they are identical.

G. Each hexahedra is bounded by six distinct quadrilaterals.

H. Every quadrilateral is contained by at least one hexahedra and no more than 2.

I. Two hexahedra share at most one quadrilateral (preventing doublets which imposes a stricter requirement on element topology).

It should be noted that items I and L may prescribe a stricter version of a hexahedral mesh than other hexahedral mesh definitions. Specifically, these constraints prohibit an element in the hexahedral mesh known as a 'doublet' [67], that may be acceptable in other mesh definitions. (A doublet occurs when two adjacent faces in the mesh share two or more edges, as shown in Figure 2.2.)

2.2.2 Boundary Constraints in the Primal Mesh

The boundary constraints define a correspondence between the boundary of the hexahedral mesh and the boundary of the solid geometry. In a valid hexahedral mesh, it is possible to determine the boundary of a hexahedral mesh topologically by testing elements for connectedness to other elements in the mesh. This fact was originally utilized

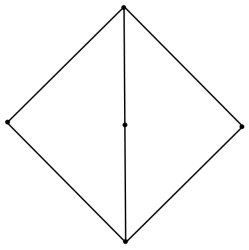

Figure 2.2. A quadrilateral doublet, where two adjacent quadrilaterals share two edges. Similar types of doublets occur in 3D with adjacent hexes sharing two or more quadrilaterals.

by Euler in formulating topologic characteristics for simplicial complexes. For example, a quadrilateral in a hexahedral mesh must be connected to either two hexahedra or a single hexahedron. If the quadrilateral is connected to a single hexahedron, it is by definition on the boundary of the hexahedral mesh.

We can constrain the boundary of the hexahedral mesh to be topologically similar to the geometric boundary with the following constraints:

J. Every quadrilateral on the boundary of the hexahedral mesh must correspond to a surface on the geometric boundary.

K. Every surface on the geometric boundary must have a collection of simply connected quadrilaterals on the boundary of the hexahedral mesh that approximates the surface.

L. Every curve on the geometric boundary must have a collection of simply connected edges on the boundary of the hexahedral mesh that approximates the curve.

M. Every vertex on the geometric boundary must be associated with a node in the hexahedral mesh.

2.2.3 Geometric Constraints in the Primal Mesh

The geometric constraints define acceptable element shapes in the hexahedral mesh. The particular constraints are typically imposed by the upstream application that intends to utilize these meshes (i.e., numerical analysis via finite element or finite difference techniques make assumptions that these element shapes are enforced in the geometric representation presented by the mesh.) Other upstream methods may impose additional

constraints (or even reduce or remove the constraints imposed below). We specify the following constraint as being generally recognized as necessary for most hexahedral meshes.

N. All quadrilaterals and hexahedra should be convex and have positive volume.

Satisfaction of this constraint is typically verified by calculating the Jacobian determinant for an individual hexahedron. In mechanics, the Jacobian matrix for a set of three equations is defined as:

$$J = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \frac{\partial y_3}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_3}{\partial x_2} \\ \frac{\partial y_1}{\partial x_3} & \frac{\partial y_2}{\partial x_3} & \frac{\partial y_3}{\partial x_3} \end{bmatrix}$$

To simplify the analytical expressions for elements of complex shapes, reference elements are typically utilized with the solution being calculated in a reference space and then mapped to the actual geometry of the elements in the mesh [25]. For a hexahedral element, a unit cube is used as the reference element and the Jacobian matrix describes the mapping from the reference unit cube to the actual geometric shape of the hexahedral element. For a single node of a hexahedron, the Jacobian matrix for mapping the hexahedral element to the unit cube can be written as:

$$J_h = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}$$

where x, y, z describe the axes of the space where the hexahedral element resides and $\xi, \eta, and \zeta$ describe the axes of the reference element space (refer to Figure 2.3).

Then, because the reference element is a unit cube, the Jacobian matrix at a single node of a hexahedra (e.g., node 0 in Figure 2.3) can be rewritten as:

$$A_0 = \begin{bmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{bmatrix}$$

For a single hexahedron, there will be eight such matrices (for additional discussion on elements with multiple Jacobian matrices see [47]). The minimal determinant of these eight matrices is known as the 'Jacobian' metric of the hexahedral element. The lengths of the sides of the hexahedron can be normalized to provide a 'scaled' version of this metric that will have values between -1 and 1 [46]. A scaled Jacobian determinant between -1 and 0 indicates non-convexity of the element. A scaled Jacobian value of 1.0 indicates a hexahedron with interior angles between common edges of 90 degrees. Generally acceptable scaled Jacobian values range between 0.2 and 1.0.

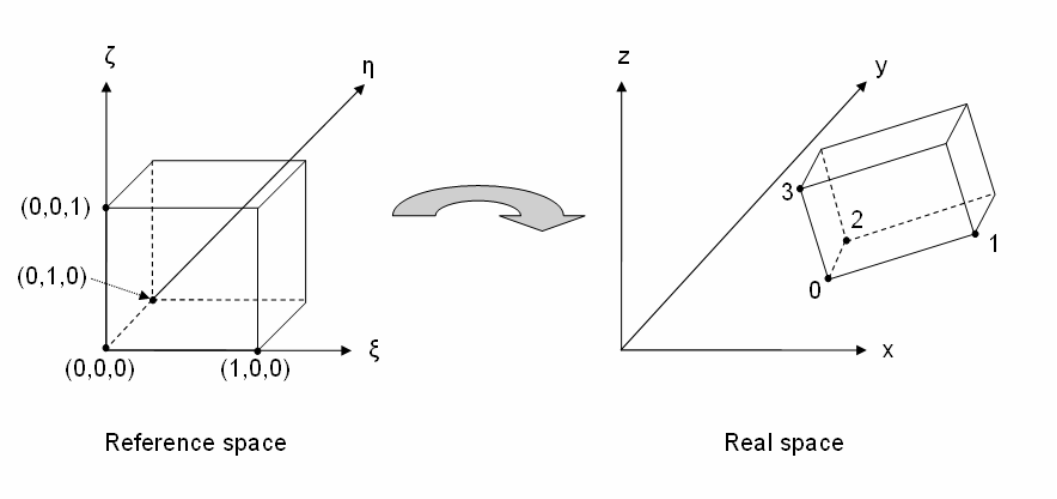

Figure 2.3. Mapping a reference unit cube to a real space hexahedron using the Jacobian.

2.3 The 'Dual' of a Hexahedral Mesh

In linear programming, it is common to look at the dual of the problem as well as the primal problem where the objective function is transformed into a set of constraints and the constraints are transformed into the objective function (see Figure 2.4). There are problems that can be easily solved in the dual that are far more difficult to solve in the primal. The duality theorem of linear programming states that there is a direct computational equivalence between the primal and the dual optimization problem.

A similar equivalence exists between the primal and dual representations of hexahedral

Primal Objective function:
Maximize $\mathbf{c}^T\mathbf{x}$
Constraints:
$A\mathbf{x} \leq \mathbf{b}$
$\mathbf{x} \geq 0$

Dual objective function:
Minimize $\mathbf{b}^T\mathbf{y}$
Constraints:
$A^T\mathbf{y} \geq \mathbf{c}$
$\mathbf{y} \geq 0$

Figure 2.4. Converting a primal linear program to a dual linear program

meshes. The use of the dual in hexahedral mesh generation was introduced in late 1993 through insights provided by Thurston [105] and Murdoch [70, 71]. Each hexahedron in the primal mesh can be represented by a single point in the dual representation (this point was defined to be a 'centroid' by Murdoch). Each hexahedron in the primal mesh is connected to at most six other hexahedra through the quadrilaterals bounding the hexahedron, so a line segment between connected points in the dual representation is analogous to a quadrilateral in the primal mesh (Murdoch defined the line segment between connected centroids to be a 'chord').

For purposes of aiding the reader's intuition, we will construct several examples of the dual of a mesh given the primal meshes. We start with a quadrilateral mesh and move to hexahedral meshes afterwards.

For a given quadrilateral mesh on a surface, the topologic structure of the dual of the mesh can be visualized by first creating a point (centroid) within the subspace defined by each quadrilateral. A line segment is drawn between each edge of the quadrilateral to the centroid of each adjacent quadrilateral sharing that edge (refer to Figure 2.5). Notice that for each quadrilateral there are always four chords emanating from the centroid; any more or less than four chords would indicate a polytope that is not quadrilateral. Additionally, because there are always four chords emanating from a quadrilateral we can infer some additional structure to the arrangement of these elements in the dual representation. Every chord for a quadrilateral can be paired with a symmetrically opposite chord in the quadrilateral (i.e., each quad has four sides, and for each side there is a corresponding 'opposite' side of the quadrilateral). Because of this symmetry in quadrilateral elements, we can show that all of the chord segments can be simply connected with other chord segments to form either a closed curve, or a curve that has end points at the surface boundary. An example of a mesh (also known as the 'primal'), with its dual, is shown in Figure 2.5.

Several important items can be gleaned from the dual representation of the quadrilateral mesh:

1. The intersection point of two chords is known as a 'centroid', and a centroid is dual to a quadrilateral in primal space.

2. Each chord represents a 'stack' of quadrilaterals in the primal mesh.

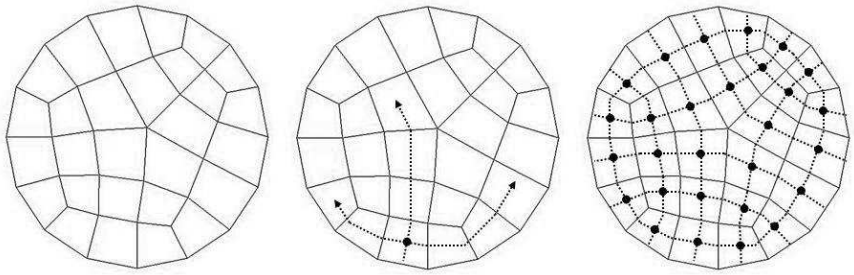

Figure 2.5. A quadrilateral mesh on a circular disk (left). The dual of a quadrilateral mesh is created by line segments (chords) connecting adjacent centroids (middle). The complete dual is on the right.

3. All chords must either have endpoints on the boundary of the surface, or they must form a closed curve within the boundary of the surface. (i.e., there is no chord with an endpoint in the interior of the surface.)

4. Chords cannot be tangent to other chords.

5. The 'size' of the primal mesh and the number of elements local to an area of the surface is a function of the density of chords and chord intersections relative to that locale on the surface.

6. As a consequence of observation three above, it can be shown that the parity of the edges around a surface admitting a quadrilateral mesh must be even (a proof of this can be found in [64]).

Extending these observations of quadrilateral meshes to hexahedral meshes, we can formulate the dual of a hexahedral mesh. Since each hexahedral element will consist of three pairs of opposing quadrilaterals (similar to the two opposing edges for quadrilateral elements), we can draw line segments between the centers of each of the opposing faces of the hexahedra. We observe that, in similar fashion to the chords of a quadrilateral mesh, the chords within a hexahedral mesh define a stack of hexahedra. However, we also observe that these stacks now interact in two directions resulting in 'layers' of elements. A layer of elements corresponds to a surface in the dual space of the mesh. This dual surface has been referred to as a 'sheet' [64], an 'interior surface' [27], or as a 'twist plane' [71, 70]. For the purposes of this dissertation, we will utilize the term 'sheet' to apply to these dual surfaces. An example dual subdivision of a hexahedral mesh is shown in Figure 2.6.

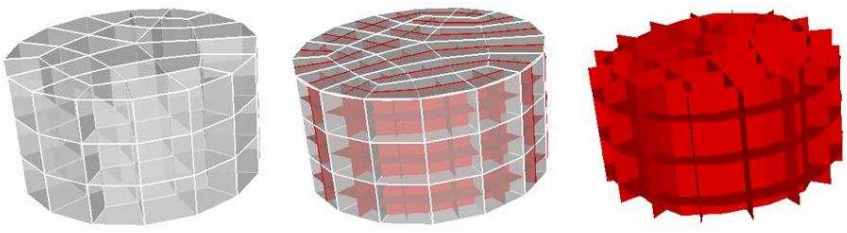

Figure 2.6. Example of a cylinder meshed with hexahedra (left). The picture in the middle shows the hexahedral mesh with its dual subdivision. The sheets of the mesh are shown on the right.

Reviewing a valid hexahedral mesh with its dual subdivision, the following observations can be made:

1. Only three sheets can intersect at a single point. The intersection point of three sheets has been identified as a 'centroid' [70]. A centroid is dual to a single hexahedron in primal mesh.

2. Each sheet in the dual space represents a 'layer' of hexahedral elements in the primal mesh.

3. All sheets either form a closed shell within the mesh space, or have a terminating boundary at the boundary of the mesh (i.e., there is no sheet that terminates in the interior of the hexahedral mesh.)

4. Sheets will not be tangent to other sheets.

5. The 'size' of the primal mesh and the number of elements local to an area of the surface is a function of the density of sheets and the triple intersections of sheets relative to that locale within the mesh.

Utilizing these observations along with theorems of topology, Thurston theorized [105] that for a given solid, any quadrilateral mesh composed of an even number of quadrilaterals should admit a compatible hexahedral mesh within the solid. This theory was later proved by Mitchell in [64], and shown to have linear complexity by Eppstein in [27]. A constructive proof and upper bound on this theorem was given recently by Carbonera, et al. [18] (see also [74]). These proofs, however, have not resulted in any practical algorithms for generating hexahedral meshes in an arbitrary solid that are suitable for analytic use.

2.3.1 Topologic Constraints in the Dual Mesh

The dual of the mesh is a topological structure; however, inherent geometric properties can be assigned to the each of the dual entities from the inherent geometric properties of the mesh that the dual entities describe.

Definition: A **sheet** is a regular, 2-manifold that is continuously differentiable (i.e., continuous and smooth), and may be defined parametrically.

Definition: A **chord** is a regular curve created by the line of intersection of one or more sheets.

Definition: A **centroid** is a point of intersection created by the triple intersection of one or more sheets.

Definition: The dual of the mesh **M** is the set of intersecting 2-manifolds **M*** such that:

A*. At any point there can be at most three intersecting 2-manifolds.

B*. A set of 2-manifolds cannot meet at a tangency

C*. A single intersection of any 2-manifold (including self-intersection) must result in a 1-manifold. The 1-manifold may overlap but it does not self-intersect.

D*. The intersection of a 1-manifold and a 2-manifold result in a triple-point intersection.

E*. Every 2-manifold must contain at least one triple-point intersection.

F*. Every 2-manifold must contain more than one triple point intersection.

2.3.2 Boundary Constraints in the Dual Mesh

Additionally, a collection of sheets satisfying the topology constraints outlined above creates a new arrangement of cell complexes. We define two of these additional cell complexes below (see also Figure 2.7). For simplicity the geometric boundary is considered to be part of this arrangement, although distinctly identified from other structures identified. (By convention, boundary sheets are typically extended past the extent of the mesh or through the adjacent element across the opposite face for visualization purposes.)

Def. A **2-cell** is a subset of a sheet whose area is bounded by a set of chords and/or the geometric boundary.

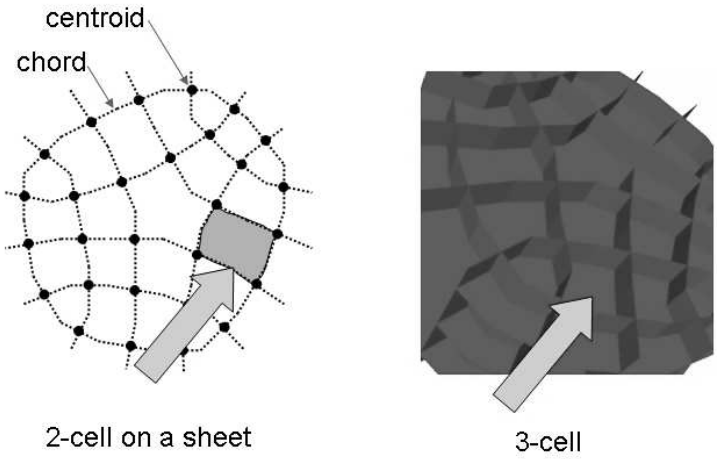

Figure 2.7. Entities in the dual.

Def. A **3-cell** is a subset of the geometric space, bounded by 2-cells and/or the geometric boundary.

It is now possible to construct a table (Table 2.3) indicating entity correspondence between the dual and primal spaces for hexahedral meshes.

The boundary constraints in the dual are analogous to the constraints the primal.

G*. Every geometric surface in the model must have one or more sheets associated with it and patches in the sheet must map to every surface quadrilateral.

H*. Every geometric curve in the model must map to the intersection of two sheets. The intersection forms a chord of hexes with two or more quadrilateral faces on the boundary.

I*. Every geometric vertex in the model must map to at least $\boldsymbol{c}$ intersections of three sheets, where $\boldsymbol{c}$ is equal to:

Table 2.3. Conversions between dual and primal entities.

Primal Entity	Dimension	Dual Entity	Dimension
Hex	3	Centroid	0
Quad	2	Chord	1
Edge	1	2-Cell	2
Node	0	3-Cell	3

$$max(v-2,1)$$

where v is the vertex valence. (Vertex valence is the number of geometric curves connected to the vertex. A vertex with 3 curves has a valence of 3.)

2.3.3 Geometric Constraints in the Dual Mesh

In terms of hexahedra, the ideal element for which finite element basis functions are formulated is a cube of six quadrilaterals of equal area, with edges of equal length, and that are mutually orthogonal to each other. This element also has a scaled Jacobian of 1.0. We will therefore formulate the geometric constraints in the dual of a hexahedral mesh around this basis. The arrangement of the sheets in dual space for such an ideal mesh is simply a collection of Cartesian planes.

J*. **Minimize sheet curvature -** The ideal isotropic mesh contains perfectly planar sheets. Therefore, it is desirable to minimize the local curvature of a sheet. Curvature of the sheets causes 'keystoning' of the elements, where the length of one edge is substantially different from its opposite edge. This phenomenon can be readily seen in Figure 2.8, as we increase the curvature of a single sheet of elements.

K*. **Maximize orthogonality in the sheet topology -** From the topological constraints earlier, we know that at most three sheets can intersect at a single point. From the ideal mesh, it is apparent that a perfectly orthogonal intersection of the three sheets is desired. Deviations from orthogonal intersections of the sheets will produce 'skewing' of the hexahedral elements in the mesh (see Figure 2.9).

L*. **Maximize the topologic regularity of the sheets -** A regular hexahedral mesh has eight hexahedra connected to each interior node in the mesh, and results from a regular topologic arrangement of sheets in the dual space. This regular topological arrangement of the sheets, as shown in the ideal mesh, is also desirable (and is essentially required in finite difference calculations). In finite element methods this requirement has some flexibility, but maintenance of this regular topology, where possible, has some additional benefits from several algorithmic standpoints including element numbering, matrix formulation, compression, etc. Additionally, non-regular arrangements of sheets have a tendency to interfere with the optimization of constraints 1 and 2 above.

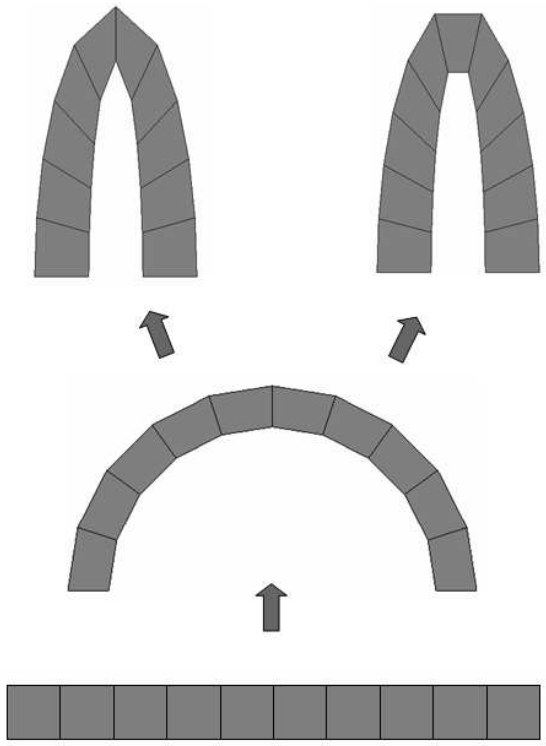

Figure 2.8. Increasing the curvature of a single sheet results in 'keystoning' of elements where one edge shrinks in size while the other grows as the curvature of the sheet increases.

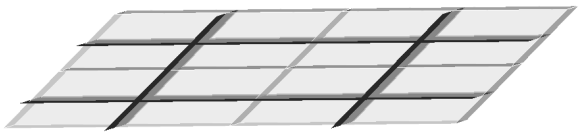

Figure 2.9. As sheet intersections deviate from orthogonality, the skew of the mesh increases. (Sheets are shown in black, primal mesh is in grey.)

2.4 Summary of Constraints

In this last section, we summarize the list of constraints for generating hexahedral meshes in both the dual and the primal space.

2.4.1 Primal Mesh Constraints

A hexahedral mesh $\mathbf{M = \{N, E, Q, H\}}$ is defined as a geometric cell complex composed of 0-dimensional nodes **N**, 1-dimensional edges **E**, 2-dimensional quadrilaterals **Q** (residing in $\Re^3$), and 3-dimensional hexahedra **H**, such that

A. Each node is contained by at least two edges.

B. Each edge contains two distinct (i.e., different) nodes.

C. If two edges contain the same nodes, the edges are identical.

D. Each quadrilateral is bounded by a cycle of four distinct edges.

E. Two quadrilaterals share at most one edge.

F. If two quadrilaterals share four edges, they are identical.

G. Each hexahedron is bounded by six distinct quadrilaterals.

H. Every quadrilateral is contained by at least one hexahedron and no more than 2.

I. Two hexahedra share at most one quadrilateral (preventing doublets which imposes a stricter requirement on element topology).

J. Every quadrilateral on the boundary of the hexahedral mesh must correspond to a surface on the geometric boundary.

K. Every surface on the geometric boundary must have a collection of simply connected quadrilaterals on the boundary of the hexahedral mesh that approximates the surface.

L. Every curve on the geometric boundary must have a collection of simply connected edges on the boundary of the hexahedral mesh that approximates the curve.

M. Every vertex on the geometric boundary must be associated with a node in the hexahedral mesh.

N. All quadrilaterals and hexahedra should be convex and have positive volume.

2.4.2 Dual Mesh Constraints

The dual of the mesh **M** is the set of intersecting 2-manifolds **M*** such that:

A*. At any point there can be at most three intersecting 2-manifolds.

B*. A set of 2-manifolds cannot meet at a tangency

C*. A single intersection of any 2-manifold (including self-intersection) must result in a 1-manifold. The 1-manifold may overlap but it does not self-intersect.

D*. The intersection of a 1-manifold and a 2-manifold results in a triple-point intersection.

E*. Every 2-manifold must contain at least one triple-point intersection.

F*. Every compact 2-manifold must contain more than one triple point intersection.

G*. Every geometric surface in the model must have one or more sheets associated with it and patches in the sheet must map to every surface quadrilateral.

H*. Every geometric curve in the model must map to the intersection of two sheets. The intersection forms a chord of hexes with two or more quadrilateral faces on the boundary.

I*. Every geometric vertex in the model must map to at least $\boldsymbol{c}$ intersection of three sheets, where $\boldsymbol{c}$ is equal to:

$$max(v-2,1)$$

where v is the valence of the geometric vertex.

J*. Minimize sheet curvature.

K*. Maximize orthogonality in the sheet topology.

L*. Maximize the topologic regularity of the sheets.

CHAPTER 3

REVIEW OF HEXAHEDRAL MESH GENERATION ALGORITHMS

In this chapter, we will review existing hexahedral mesh generation algorithms and discuss the methods these algorithms utilize to satisfy the hexahedral mesh generation constraints outlined in Chapter 2. We will also discuss historical methods for evaluating the performance of a given algorithm.

3.1 Introduction

In a presentation to attendees at the 9th International Meshing Roundtable, Blacker [10] reiterated several algorithmic considerations that are utilized in comparatively evaluating hexahedral mesh generation algorithms. I reiterate them here for their value in evaluating algorithms, and also for definitional value in understanding advantages and disadvantages inherent in reviewing algorithms. In addition to these characteristics, I will also review several hexahedral mesh generation algorithms with regard to the paradigm that each algorithm utilizes to satisfy the topologic, boundary and geometric constraints outlined in Chapter 2. Blacker outlined seven characteristics of a good hexahedral algorithm, namely:

1. *Geometric Generality* - The algorithm should handle a large class of geometries with arbitrary complexity and detail.

   In hexahedral mesh generation, most algorithms will assume that one set of constraints are fixed and then work to optimize the other two constraint sets. By fixing one set of constraints, an algorithm will only work for the geometric set of problems for which the constraint set is fixed.

2. *Geometric Matching* - The algorithm should contain the geometric features identified in the solid being meshed.

In some sense, this consideration asks how well an algorithm satisfies the boundary constraints of a hexahedral mesh. In some cases, an algorithm working under the assumption of a fixed mesh topology may have difficulty capturing specific boundary features since the mesh topology utilized will not be readily homeomorphic to the geometric boundary.

3. *Boundary Sensitivity* - The algorithm should produce high-quality elements at the domain boundary.

   Historically, this consideration dealt with the regularity of the mesh topology at the boundary coupled with the how nearly orthogonal the sheets at the local boundary locations are with respect to the boundary surface.

4. *Orientation Insensitivity* - The orientation of the geometry should not affect the mesh generated by the algorithm.

   Another way of saying this would be: "The algorithm should produce identically (or nearly so) meshes independent of rigid body transformations (e.g., rotations or translations) of the geometry." This consideration can normally be satisfied by anchoring the mesh topology to the geometry.

5. *'Bad' Geometry Tolerant* - The algorithm should be able to operate despite gaps, overlaps, holes, etc. in the geometry.

   Upon determination of a mesh topology which generally satisfies the boundary and quality constraints, a general operation in most algorithms is to assign each individual mesh entity to an owning geometric entity. With 'bad geometry' these assignments can be ambiguous at times. The algorithms' ability to deal with these ambiguities is the subject of this consideration.

6. *Size Control* - The algorithm should match the desired element sizing constraints throughout the domain.

   Because the topological constraints of hexahedral mesh generation require a sheet to be a manifold within the mesh space (that is the sheet must divide the space into at least two regions); at times it can be difficult to control the sheet density in local areas while still satisfying the mesh quality constraints. This consideration pertains to how well an algorithm is able to balance these oft-times conflicting requirements.

7. *Speed* - Generate reasonably large meshes (>1M elements) in 'interactive' time. Tetrahedral meshing speeds are desirable.

   Tetrahedral mesh generation algorithms are currently available which are generating on the order of several hundred thousand elements per minute on single processor machines. A desirable hexahedral algorithm would be able to generate hexes in the same order of time, while still adequately meeting the previous algorithmic considerations for its operation class of geometries.

In Chapter 2, we noted that the dual sheet infers element connectivity for a layer of hexahedra. Because it is possible to infer mesh properties for a large collection of elements utilizing the dual description of the mesh, we will use the dual form of the constraints listed in Chapter 2 as the basis for evaluating generation algorithms. In this chapter, we will review how these constraints are incorporated into several current hexahedral mesh generation algorithms. In some cases, these constraints are weakly satisfied by the algorithm, and we can then demonstrate how an algorithm may be extended to support a wider class of geometries by a stronger incorporation of the constraint capturing techniques. In the remainder of this chapter, we will review several existing algorithms for generating hexahedral meshes, and, then, compare these algorithms to the constraint sets and evaluate their ability to capture these constraints. Owen [75] categorized the classes of hexahedral mesh generation algorithms as follows: Structured Methods, Direct Methods, and Indirect Methods. We will examine a few algorithms in each category and suggest how better knowledge or incorporation of the constraints detailed above may extend the class of geometries that the algorithm can successfully mesh.

By way of reminder, the three classes of constraints for hexahedral mesh generation are outlined in Table 3.1.

3.2 Generation Methods

3.2.1 Structured Methods

Strictly speaking, a structured mesh is characterized by all of the interior mesh nodes having an equal number of adjacent elements. Unstructured meshes, on the other hand, relax the nodal valence requirement, allowing any number of elements to meet at a single node. In this section, we will review the most common forms of structured mesh generators available today.

Table 3.1. Recapitulation of the classes of constraints for hexahedral mesh generation.

Constraint Class	Description
Topologic	Defines requirements on the arrangement of the sheets needed for producing hexahedral cells.
Boundary	Defines requirements needed to capture geometric entities defining the boundary of the solid being meshed.
Quality	Defines the allowable sheet conformations which present satisfactory quality hexahedral meshes.

Structured meshing algorithms include the octree approaches [89, 121, 94, 123], mapping [22], multiblock [24, 120, 79, 3], submapping [112, 113], and sweeping algorithms [43, 97, 84, 52, 51, 96, 9]. These algorithms are categorized as structured, or semistructured methods because of the regularity of the meshes that they typically produce.

These methods work for a well-defined set of geometric topologies. For instance, mapping algorithms require geometric topologies conforming to a cuboid (i.e., six squareish surfaces composed of four curves each to form the boundary of the solid), and sweeping algorithms require cylindrical topologies. Algorithms that automatically detect or enforce the predefined topologies have significantly improved the time required to generate hexahedral meshes [117, 65, 66, 95, 116].

The most popular, and commonly displayed, types of hexahedral meshes are those utilizing a multiblock method for generating a hexahedral grid [120, 79, 3]. The major advantages of these methods are related to the size of the meshes and relative speed with which large meshes can be constructed once the block decompositions have been defined. An individual skilled in generating multiblock-type grids can generate complex meshes with exceptional quality. Because of the control that is available in the creation and layering of the mesh, this method is the common choice for hexahedral decompositions in fluid dynamic simulations where fine control of the mesh near geometric boundaries is required.

The major disadvantage associated with these algorithms is their seeming inflexibility to incorporate additional geometric topologies to the predefined geometric topology written for the algorithm. Therefore, hexahedral meshing for nonconforming geometric topologies results in extensive decomposition of the solid into the predefined topologies

[12, 57, 103], suppression of incompatible topology [50], predefined topology overlays (used extensively in multiblock methods), refinement [110], or massaging the elements to fit the geometry [109, 125].

With these algorithms, satisfaction of the boundary constraints is only guaranteed if the geometric topology matches a predefined geometric topology. Potentially, all of these algorithms could be augmented for larger classes of geometric topologies by enabling the algorithms to capture additional boundary details. It may be possible to incorporate some additional paradigms directly into these methods that allow the hexahedral boundary constraints to be satisfied during the course of the initial mesh generation phase.

3.2.1.1 Mapping. The most popular, and most widely available, algorithm for hexahedral mesh generation is an algorithm known as mapping [22]. This algorithm serves as the foundation for the submapping and multiblocking schemes referred to later in this section. The basis of this algorithm is the mesh of a simple cube (i.e., solid consisting of six faces oriented in a box-like shape).

The mapping algorithm is a basic form of a structured grid technique where parametric coordinate planes are utilized to divide a space into cubes, or hexahedra. The coordinate planes can be scaled to meet the desired element sizing constraints. A solid that has a nearly cuboid topology and shape is 'mapped' to a parametric coordinate system for subsequent decomposition into hexahedra. This mapping allows for translations, rotations, scalings, and even nonuniform projections, while still maintaining reasonable quality in the final decomposition (see Table 3.2). Some example meshes created by a mapping algorithm are shown in Figure 3.1.

While there are several variant implementations of the mapping algorithm, the most common method is to identify the desired cubical topology from the geometric solid in a bottoms-up approach. That is, first identify the pertinent features of the primitive geometry (i.e., the eight corners of the cuboid), then ensure that the connectivity between these eight corners is correct for a cubical object. A map can then be established between the parametric coordinate planes by parameterizing the geometry into the three coordinate axes. The boundary curves are decomposed to the desired size, followed by the surfaces, and finally the solid is divided by using a transfinite interpolation algorithm to determine the optimal placement of the interior nodes to match the desired hexahedral decomposition.

Some additional enhancements of these algorithms in code enable additional geome-

Table 3.2. Satisfaction of the hexahedral mesh generation constraints for general geometries by a mapping algorithm.

Constraint Class	Description
Topologic	**Strong**-The subdivision of a Cartesian coordinate system utilizing coordinate planes meets all necessary topologic requirements for hexahedral mesh generation.
Boundary	**Weak**-The boundary requirements for hexahedral mesh generation are only strongly satisfied when the geometry is a topological cube by mapping each geometric boundary to one of the parametric coordinate planes utilized in the hexahedral decomposition.
Quality	**Neutral**-Quality constraints are satisfied so long as the mapping function does not require extreme transitions for each of the coordinate planes when mapped to the geometric solid.

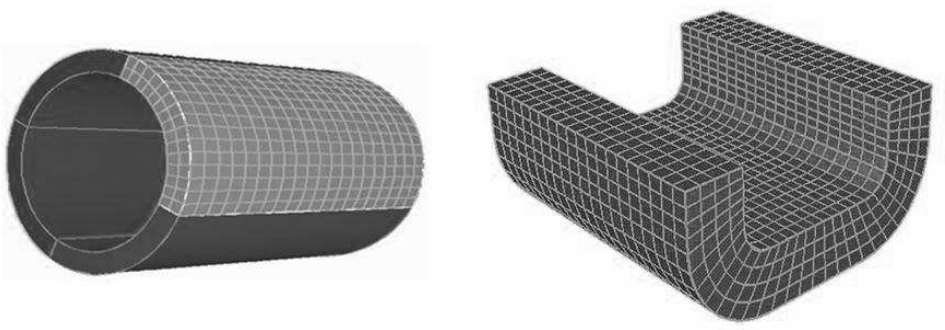

Figure 3.1. Example meshes using a mapping algorithm.

tries, which do not strictly meet the geometric primitive class, to also be meshed with this algorithm. For instance, a cubic solid, composed of more than six surfaces, which fits the general description of a cuboid primitive, but not the strict primitive definition, may still be meshed with minor modifications to the general algorithm utilizing a corner-picking algorithm [117, 65].

One of the main advantages of the mapping paradigm is that the set of parametric coordinate planes intrinsically satisfies the topology and quality constraints for formulating a hexahedral mesh. Therefore, the only truly difficult task associated with a mapping algorithm is ensuring that the geometric solid boundaries are mapped correctly into a parametric coordinate system to ensure that the boundary constraints are satisfied. Initial

identification of the six sets of surfaces forming the cuboid topology and determining a consistent and regular internal topology for the surfaces is often the most difficult area of the algorithm.

3.2.1.2 Submapping and blocking. A fairly major extension to mapping algorithm involved 'stacking' several 'mappable' solids in an effort to increase the class of geometries for which the mapping algorithms would work. There are two main approaches to identifying, or creating, these 'stacked' geometries which are highly utilized in hexahedral mesh generation: blocking and submapping.

The submapping algorithm identifies mappable regions within a volume. Assuming that a consistent mapping can be found for all regions within the volume, the submapping algorithm will then 'virtually decompose' the solid into separate mappable subregions, while ensuring that the constraints within each of the subregions is consistent with the adjacent subregions. [112, 113]

Blocking algorithms (also known as multiblock) 'wrap' solids, or portions of solids, with cuboid topologies. Each of the new cuboids is mated against other adjacent cuboids, ensuring that the mesh topology of each cuboid is consistent with each of the adjacent cuboids. This process is continued until the geometric space is filled [24]. To speed this process, several composite blocks are often applied simultaneously. Multiblocking algorithms are the most popular and commercially available hexahedral mesh generation algorithm [120, 79, 3], and are very popular with the fluid dynamics community for their ability to generate long-thin elements (i.e., high aspect ratios) that are suitable and necessary in region of a dynamic fluid where boundary layers develop.

Again, by utilization of parametric coordinate planes, which are mapped to sets of solids (instead of a single solid in the case of the original mapping algorithm) the topology and quality constraints are intrinsically satisfied for a hexahedral decomposition. The major difficulty in the development of submapping and blocking algorithms is ensuring that the boundaries can be parametrically mapped to the three parametric dimensions in order to adequately satisfy the boundary constraints for the hexahedral decomposition (see Table 3.3). Several solids meshed with a submapping algorithm are shown in Figure 3.2. It is important to notice in the figures that the topology arrangement of the parametric coordinate planes, from which the original mesh is mapped, is visible in the dual sheets of these meshes.

As an aid to identifying stacked sets of mappable solids, and ensuring consistency

Table 3.3. Satisfaction of the hexahedral mesh generation constraints by a blocking or submapping algorithm.

Constraint Class	Description
Topologic	**Strong**-The subdivision of a parametric coordinate system utilizing coordinate planes meets all necessary topologic requirements for hexahedral mesh generation.
Boundary	**Neutral**-Assuming that decomposition of the volume in to mappable subregions is feasible, the boundary requirements for hexahedral mesh generation are strongly satisfied by mapping each geometric boundary to one of parametric coordinate planes utilized in the hexahedral decomposition.
Quality	**Neutral**-Quality constraints are satisfied so long as the mapping function does not require extreme transitions for each of the coordinate planes when mapped to the geometric solid.

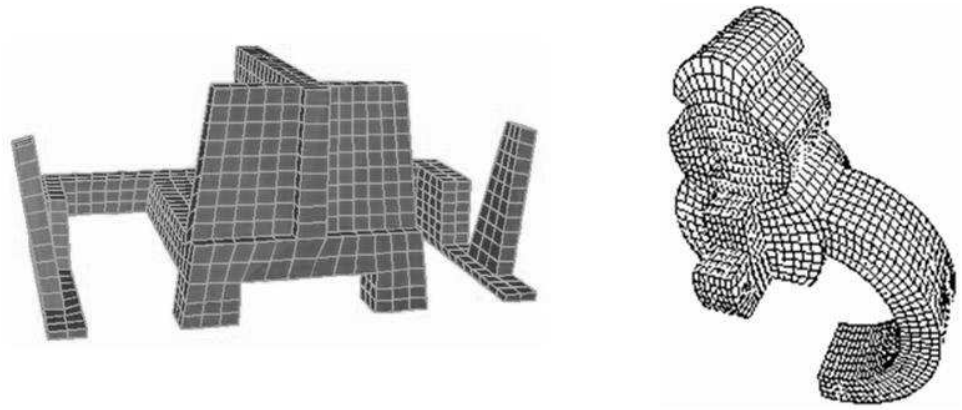

Figure 3.2. Example meshes using a submapping algorithm.

between adjacent regions, automation algorithms for 'interval matching' and 'corner-picking' have been developed. Interval matching algorithms essentially ensure that the number of sheets intersecting a geometric boundary curve is consistent across each of the subregions. Corner-picking can automate the process of identifying the eight corners of a cuboid region [117, 65, 66, 95].

3.2.1.3 Octree-based algorithms. Another algorithm which relies on the intrinsic characteristics of the set of Cartesian coordinate planes to produce a hexahedral decomposition is a class of hexahedral mesh generation algorithms known as octree-based algorithms. The most popular forms of these algorithms are detailed by Schneider's [89, 87, 110], Shephard [121, 94], and Zhang and Bajaj [124, 123].

For the octree-based algorithms, the Cartesian coordinate planes are again utilized to produce the significant portion of the hexahedral decomposition. However, instead of utilizing a transformation to map the geometry into the coordinate system, the geometry is placed directly in the coordinate frame. Then, according to the individual algorithm's heuristic rules, each element is classified as interior, or exterior, to the solid geometry and various nodal movement, or element addition, techniques are applied in attempt to improve the resulting mesh along the solid's boundary.

Several disadvantages are immediately recognized with this method. The first significant disadvantage is the potential orientation-insensitivity of the method. Because the solid is not tied to a local coordinate system, but is rather placed in an existing coordinate system, small rotations to the solid can produce significant changes in the final mesh. Additionally, misalignments of the geometry to the coordinate system can have significant impact on the resulting quality of the meshes produced.

However, the most significant disadvantage associated with this method is the seeming inability to adequately capture geometric features that are not aligned with the coordinate planes. Many varied attempts to alleviate this problem include boundary element combination [109], improved smoothing [125], and refinement techniques [110]. The inability for this paradigm to capture geometric boundaries well highlights the failure to consider the hexahedral mesh generation boundary constraints as shown in Table 3.4.

The insight that one set of the hexahedral mesh generation constraints is largely ignored signals several possible extensions to this algorithm to extend the class of solids to which this algorithm can be successfully employed. In particular, modifications which explicitly enable the algorithm to incorporate better boundary capture techniques will significantly improve this algorithm's success, as we will demonstrate in later chapters.

3.2.1.4 Semistructured. A major extension to the algorithms listed above involves incorporating a major modification to the topology perpendicular to one set of the coordinate planes. The basic idea with sweeping (sometimes known as extrusion) is to create a quadrilateral mesh and 'sweep' the topology of the quadrilaterals into hexes

Table 3.4. Satisfaction of the hexahedral mesh generation constraints by an octree-based algorithm.

Constraint Class	Description
Topologic	**Strong**-The subdivision of the Cartesian coordinate system utilizing coordinate planes meets all necessary topologic requirements for hexahedral mesh generation.
Boundary	**Weak**-Weak satisfaction of boundary constraints often result in the need for refinement and smoothing algorithms. In many cases, algorithmic failure on many solids is due to inability to adequately capture boundary features of the solid.
Quality	**Strong**-Mesh quality is typically high throughout the majority of the volume, with quality degradation occurring near the boundaries where the quality constraints are relaxed while trying to capture the boundary features.

[43, 97, 84]. The quadrilateral mesh may have an unstructured topology, but the results of the extrusion are sheets which maintain a structured arrangement in an orthogonal direction to the original quadrilateral mesh. Therefore, there is a recognizable structure in the mesh topology, which is the reason that sweeping is known as a semistructured meshing scheme. It is interesting to note that if the original quadrilateral mesh is structured, then the resulting hexahedral mesh will also be structured after sweeping (see Table 3.5).

Sweeping extends the class of geometric primitives which can be meshed with hexahedra from topological cubes to topological cylinders. Additional extensions, which are similar to submapping and blocking, have also been made to sweeping algorithms in an effort to increase the class of geometries to which the sweeping algorithms are applicable. These extensions resulted in algorithms known as "many-to-one sweeping", "many-to-many sweeping" or "multisweeping", and the "Cooper tool" (a reference to the topological similarity of barrels and cylinders)[52, 51, 96, 9]. Examples of meshes using a sweeping algorithm are shown in Figure 3.3. Additionally, a significant amount of work has also been done to optimize the placement of the nodes at each layer to enhance the resulting mesh quality [93, 85, 83].

One difficulty being encountered with these extended sweeping algorithms involves the

Table 3.5. Satisfaction of the hexahedral mesh generation constraints by a sweeping algorithm.

Constraint Class	Description
Topologic	**Strong**-A valid quadrilateral mesh topology is extruded, and several layers of sheets are added orthogonally to these sheets. Assuming that none of the newly inserted sheets are allowed to intersect or form tangencies to any of the other sheets in the projection process, the mesh should be topologically sound.
Boundary	**Neutral to Moderate**-Boundary constraints are strongly satisfied when the geometric topology is cylindrical or can be 'virtually' decomposed into compatible stacks of cylindrical topologies.
Quality	**Moderate to Strong**-Quality constraints affected only when high curvature of the sheets in the original quadrilaterals is present, or when the layers deviate from parallel with each other producing element skew in each of the layers..

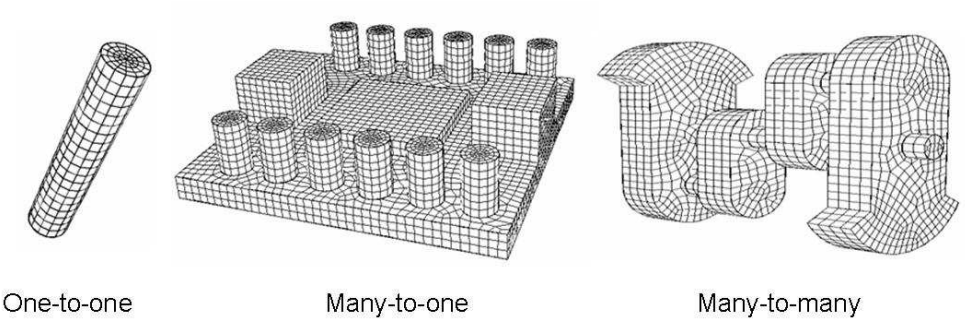

Figure 3.3. Example meshes using a sweeping algorithm.

requirement that a sheet in a hexahedral mesh must divide the mesh space. Unstructured quadrilateral meshes are often used as the beginning of a sweep because of the geometric complexity of the surface to be swept. Because the quadrilateral mesh is then extruded, artifacts of the original geometric complexity are carried by the sheets as they are projected and can be found at each layer of the sweep. If swept solids are stacked into a long cylinder, similar to what was done in multiblocking, the geometric complexity from each of the stacked solids is additive throughout the entire cylindrical chain, and

the complexity must be resolved individually within each solid in the stack. Tracking and resolving all of this induced complexity through each of the solids can become an extremely significant challenge [96].

Some additional extensions to enable the ability to sweep orthogonally to the original sweep direction have also been attempted [68, 40] with varied success.

3.2.2 Indirect Methods

The indirect methods are those algorithms that first generate a tetrahedral mesh, and then convert (or attempt to convert) the tetrahedral mesh into a hexahedral mesh. There are two common methods for converting tetrahedra into hexahedra. They are tetrahedral decomposition (also known as tet-to-hex, or t-hex, for short), and tetrahedral combination.

3.2.2.1 Tetrahedral decomposition. Provided a solid can be tetrahedrally meshed, each tetrahedron can be subdivided into four hexahedra as shown in Figure 3.4. The major advantage of tetrahedral decomposition into hexes is that it is very easy to implement and is very easy to satisfy the boundary and topologic constraints for hexahedral meshes. The disadvantage of this method is that the decomposition produces sheets that are small and spherical, resulting in relatively high curvature of all sheets within the solid.

Tetrahedral decomposition provides a way to create hexes local to a very small region. The particular tetrahedral subdivision employed tends to create small, topologically spherical sheets surrounding each node of the tetrahedral mesh. Therefore, topologic and boundary constraints for hexahedral mesh generation are easily satisfied, but the higher curvature of each of the spherical sheets has a tendency to degrade the quality of the hexahedral mesh (see Table 3.6). The characteristic boundaries of a THex mesh give an indication of the spherical dual sheet topology contained throughout the volume (see

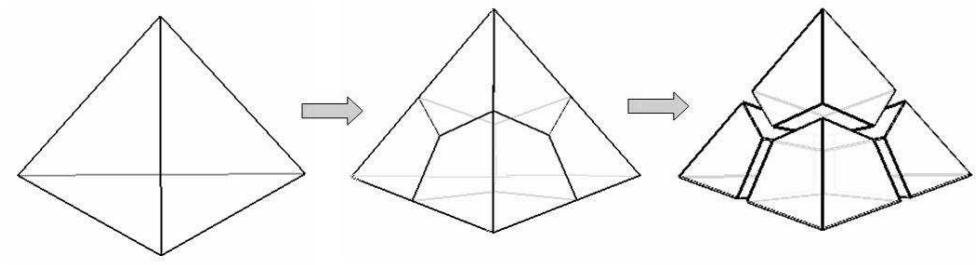

Figure 3.4. Subdivision of a tetrahedral element into four hexahedral elements.

Table 3.6. Satisfaction of the hexahedral mesh generation constraints by a tet-to-hex decomposition algorithm.

Constraint Class	Description
Topologic	**Strong**-The decomposition of each of the tetrahedra results in a set of packed, and overlapping, spherical sheets which meet all of the topologic requirements of a hexahedral mesh.
Boundary	**Strong**-Boundary requirements are satisfied by the conglomerate actions of multiple spherical sheets overlapping one another near the geometric surfaces, curves and vertices.
Quality	**Weak**-Quality constraints are weakly satisfied as the curvature of the sheets is significant. Because every sheet contains curvature, there will be few hexes which approach ideal hexahedral quality. Additionally, tetrahedral meshes tend to have high nodal valence, which is counter to the regular nodal valence desired in an ideal hexahedral mesh.

Figure 3.5). The spherical sheets coupled with high valence nodes associated with the original tetrahedral mesh result in less than ideal quality. Most finite element analysts have rejected this solution because of the poor element quality that generally results from this method.

To augment an indirect decomposition algorithm, methods are needed that reduce the curvature of each of the sheets and also reduce the high nodal valence associated with the original tetrahedral mesh. Some ideas for accomplishing this include combining adjacent sheets into single sheets thereby reducing curvature and locally coarsening the mesh (i.e., combining two slightly offset spherical sheets would produce an elliptical sheet). Iterative and strategic combinations of sheets could ultimately produce a mesh whose sheets are much more planar in nature, resulting in a higher quality mesh. Methods for combining sheets could have a powerful additional benefit in providing the foundational tools for general coarsening of existing hexahedral meshes.

Another approach to this problem is to begin with a very coarse tetrahedral mesh, split each tetrahedra into four hexahedra, and then apply a dicing algorithm [61, 60] to the resulting mesh (see [82]). The resulting mesh has a few very large spherical sheets whose local curvature is relatively low. The additional sheets produced via dicing are

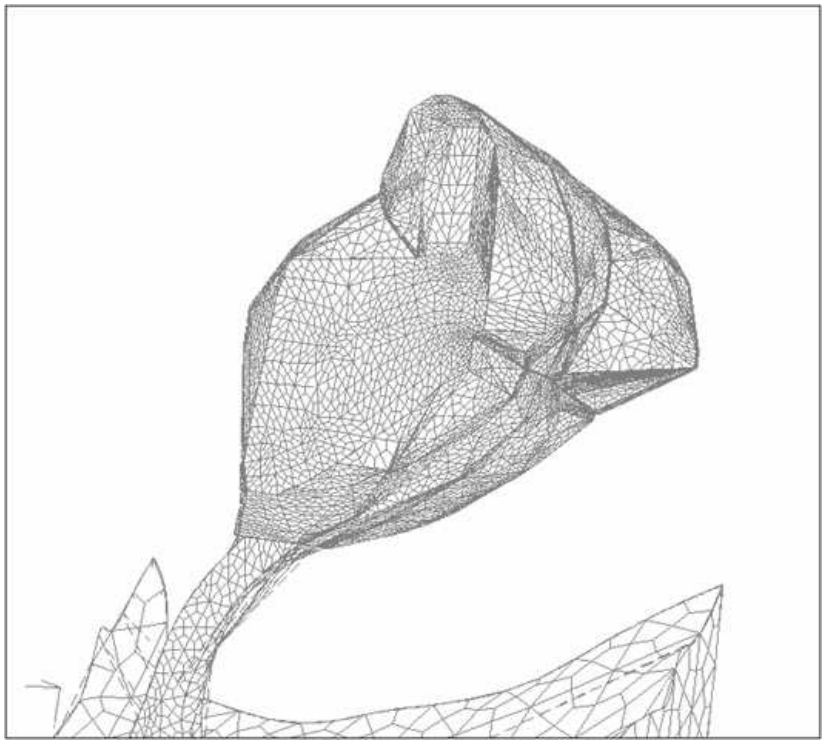

Figure 3.5. Example mesh of a mitochondrial christae using a tetrahedral-to-hexahedral decomposition algorithm.

copies of these sheets and, therefore, maintain the relatively low curvature of each of the original large spherical sheets.

3.2.2.2 Tet-to-hex combinations. H-Morph is an indirect method which utilizes an initial tetrahedral mesh coupled with tetrahedral combination techniques to transform the tetrahedral mesh into a hexahedral, or hex-dominant, mesh. H-Morph also has a two-dimensional analog, entitled Q-Morph [77]. The method is described in the H-Morph paper abstract [76] thus:

> The H-Morph method starts with an initial tetrahedral mesh and systematically transforms and combines tetrahedra into hexahedra. It uses an advancing front technique where the initial front consists of a set of prescribed quadrilateral surface facets. Fronts are individually processed by recovering each of the six quadrilateral faces of a hexahedron from the tetrahedral mesh. Recovery techniques similar to those used in boundary constrained Delaunay mesh generation are used. Tetrahedra internal to the six hexahedral faces are then removed and a hexahedron is formed. At any time during the H-Morph procedure a valid mixed hexahedral/tetrahedral mesh is in existence within the volume. The procedure continues until no tetrahedra remain within the volume, or tetrahedra remain which cannot be transformed or combined into valid hexahedral elements. Any remaining tetrahedra are

> typically towards the interior of the volume, generally a less critical region for analysis. Transition from tetrahedra to hexahedra in the final mesh is accomplished through pyramid-shaped elements. Advantages of the proposed method include its ability to conform to an existing quadrilateral surface mesh, its ability to mesh without the need to decompose or recognize special classes of geometry, and its characteristic well-aligned layers of elements parallel to the boundary.

The biggest problem that will occur in this method is an inability to always satisfy the topological constraints of hexahedral meshes. This manifests itself when the method completes with a hex-dominant mesh only. The boundary constraints are generally satisfied first due to the advancing front nature of the algorithm, although these constraints are not considered explicitly within the algorithm. The quality constraints are addressed locally with the formation of each hexahedra by heuristic methods to determine which tetrahedra (or modified tetrahedra via edge swaps) to combine. (see Table 3.7)

This method has not been made widely available because of the complexity of the heuristic algorithms utilized to form the hexahedra, and because of its inherent failure in satisfying the topology constraints for generating a fully hexahedral mesh.

Table 3.7. Satisfaction of the hexahedral mesh generation constraints by a tet-to-hex combination algorithm.

Constraint Class	Description
Topologic	**Weak to Neutral**-Topologic constraints are satisfied in a local manner, which often results in sheets that do not always divide the space. When a sheet cannot be created which spans the space, the algorithm switches to a hex-dominant mesher, and an all-hexahedral mesh is not realized.
Boundary	**Moderate to Strong**-These constraints are also satisfied in a local manner. However, since the boundary elements are often the first created, boundary constraints are often satisfied while the interior topology constraints are compromised.
Quality	**Moderate to Strong**-Hexes are formed in a local manner, which results in relatively planar sheets and high quality meshes.

3.2.3 Direct Methods

The direct methods include the following classes of algorithms: medial axis, plastering and whisker weaving. These methods generate the hexahedral mesh directly rather than indirectly in the geometry.

3.2.3.1 Medial axis algorithms. The medial-axis algorithms [53, 80, 81, 1] define decompositions of the volume into simpler primitives that can be meshed with a structured scheme. The medial surface, or axis, of the volume is first constructed. This surface can be thought of as the surface interior to the volume which is generated by the midpoint of a maximal sphere as it is rolled through the volume. It is hypothesized that the decomposition of a volume by its medial surface will produce a set of mappable regions or a set of primitives for which a mappable decomposition is known. Meshing of volumes created after medial axis decomposition is performed utilizing the structured methods outlined earlier.

While medial axis algorithms have been utilized with success on some volumes, robustness issues with generating the medial surface still tend to be problematic. Additionally, the true medial surface of a geometry often results in undesired geometric decompositions. Because medial axis algorithms are utilized as decomposition algorithms for hex meshing, there is also a potential for generating a decomposition whose mesh is not consistent across multiple volumes.

Medial axis algorithms are still an active area of research, and the interest in these algorithms tends to ebb and flow with respect to hexahedral mesh generation. Efforts to produce the medial axis utilizing a Voronoi decomposition coupled with medial cleanup operations have shown some additional promise, although robust and highly utilized hexahedral decomposition algorithms are available on a very limited basis.

3.2.3.2 Plastering. The plastering [11] algorithms start with a quadrilaterally meshed boundary and place hexahedra individually onto the quadrilaterals in an advancing-front manner. The original algorithm was entirely heuristic, with little knowledge of the sheet structure associated with hexahedral meshes. The resulting algorithm often finalized with void regions where the algorithmic heuristics were exhausted and no additional hexahedra could be placed to close the void region.

More recent approaches that take into account a better understanding of hexahedral constraints offer a more realistic chance of success. The method proposed by Staten, et al. [99, 98] inserts entire sheets (rather than individual hexahedra) directly into the volume

based on geometric boundary definitions while still building on some of the paradigms advanced in the plastering and H-morph [76] algorithms. By advancing entire sheets, rather than individual elements, it is hoped that void closure problems can be avoided.

In the unconstrained plastering algorithm, a solid is first meshed with a tetrahedral mesh. This tetrahedral mesh is utilized to guide placement of a full-layer, or sheet, of hexes. By placing full sheets, rather than individual hexes, the topology constraints of the hexahedral mesh are consistently satisfied as each front is advanced (see Table 3.8). As long as the front can be advanced such that a void region remains that is either sweepable, or meshable using a known hexahedral primitive, the algorithm can generate a hexahedral mesh. While research into this algorithm continues, it is unclear whether, or not, void regions which remain unmeshable will be encountered.

3.2.3.3 Whisker weaving. Whisker Weaving is the most unique algorithm available in hexahedral mesh generation, and there are several variants of this algorithm available (see [101, 29, 69, 17]). Given a closed set of surfaces and a quadrilateral mesh on those surfaces, the dual of the quadrilateral mesh is a closed set of curves which will be the boundaries of the interior surfaces of an enclosed hexahedral mesh. The basic paradigm for the remainder of the Whisker Weaving algorithm was originally described by Thurston [105], as follows:

Table 3.8. Satisfaction of the hexahedral mesh generation constraints by the plastering algorithm.

Constraint Class	Description
Topologic	**Weak**-Topologic constraints are handled in a local manner by advancing elements or sheets directly into the volume being meshed. The algorithm has difficulty when the remaining fronts cannot be advanced without interference with other front regions and no known hexahedral topology can be inserted to fill the remaining void.
Boundary	**Moderate to Strong**-Boundary constraints are typically satisfied first since the front starts at the boundary and works inward.
Quality	**Neutral to Strong**-Quality should remain high with possible exception of the elements utilized to fill the final void region where regions of high curvature and/or skew might be encountered.

> This gives a way to construct the surface with boundary the given curves. It may not be dual to a hexahedral decomposition, because for instance there may be no interior vertices (corresponding to hexahedra) at all. So, add a bunch of spheres with enough intersection points with the existing surface to make lots of interior vertices. This turns it into the dual of a hexahedral decomposition.

Each of the weaving-based algorithms starts by utilizing a quadrilateral mesh on the surface to define the exterior boundaries of the sheets, which describe the interior hexahedral mesh. Then, by utilizing the topological constraints and several heuristic rules, the interior manifold of each of the sheets is developed until all sheets have been completed. The topology rules dictate how each sheet is allowed to intersect adjacent sheets. On a single sheet manifold, the intersection with another sheet is known as a chord, and it is from the 'weaving' of these chords that whisker weaving derived its name. Assuming that there are no self-intersecting sheets, each sheet is homotopic to a disk and all of the sheets can be drawn with chords on each sheet showing the progress of weaving the chords together on a 'sheet diagram'. The algorithm is mostly complete when all of the chords have been joined, and is completed by inserting spherical sheets as necessary to satisfy all of the remaining topological constraints, as described by Thurston above.

In [29], Mitchell and Folwell utilized Mitchell's hexahedral mesh generation existence proof [64] to guarantee algorithmic completion and topologic consistency of the resulting mesh. However, it was quickly discovered that the resulting meshes did not always have sufficient quality needed for use in subsequent analyses.

Because the quadrilateral mesh on the geometric boundary defines the boundaries of the sheets that will be placed on the interior of the volume, the premeshed boundaries are also the cause of the poor quality, or subsequent failure of these algorithms. As shown in Figure 3.6, it is quite common for quadrilateral meshes on the boundary to define sheets with high geometric complexity. Assuming the algorithm is sophisticated enough to solve the topologic requirements to define the sheets, the interactions of these high curvature sheets often prevent reasonable quality hexes from being realized.

Some success has been found by removing the problematic sheets with a sheet extraction algorithm [13], or placing additional constraints on the quadrilateral mesh placed on the boundary. In particular, Hannemann [69] was able to generate some relatively complex hexahedral meshes by utilizing quadrangular patches and reasonably structured surface meshes as the input to a dual creation algorithm.

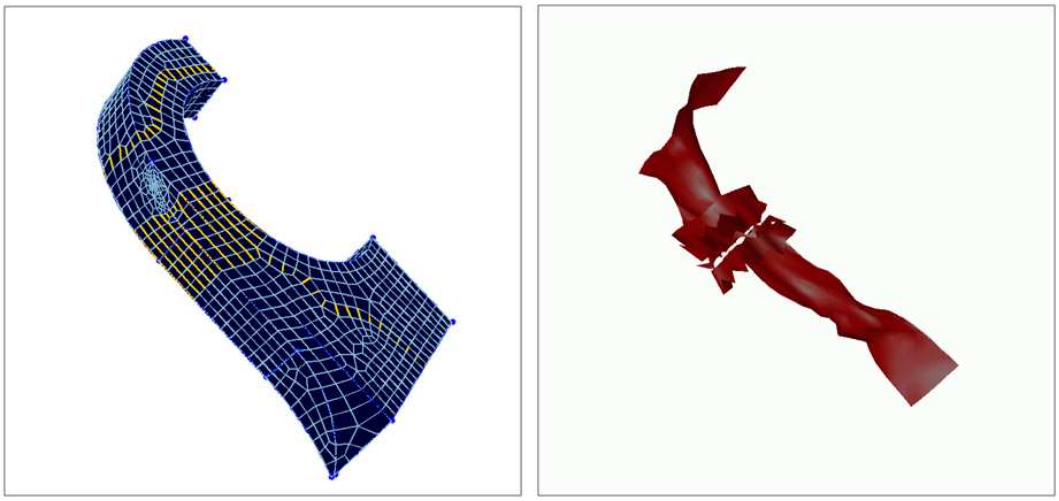

Figure 3.6. The surface mesh (shown on the left) has a dual cycle that spirals up to the top of the image. The resulting hex mesh, generated with whisker weaving, cannot be untangled. The sheet generated by whisker weaving is shown on the right.

Additional efforts have been conducted to try and improve the boundary mesh presented to a whisker weaving algorithm [36, 119], as well as to remove all self-intersections on the quadrilateral boundary mesh using templates and quadrilateral collapse operations [69, 29, 17]. Recent work has also been done to generate the sheets topologically and geometrically and incorporate the sheet self-intersections into the algorithm [100]. Also of note, Carbonera and Shepherd [18] introduced a constructive algorithm which can directly generate a hexahedral mesh topology in a volume with an even number of quadrilaterals as originally indicated by Mitchell's hexahedral existence proof [64]. These methods tend to be successful in satisfying hexahedral topology and boundary constraints, but have difficulty guaranteeing feasible quality (see Table 3.9).

3.3 Modification Methods

Over the years, several methods have been developed that make it possible to improve the flexibility of existing hexahedral meshing algorithms and aid in capturing constraints overlooked by the paradigm employed by a specific algorithm. In this section, we highlight some of these methods and show how they help to satisfy some of the constraints for hexahedral meshing of solid models. In particular, we discuss methods for inserting and extracting sheets, including pillowing [67], dicing [61, 60], generalized hexahedral refinement [104, 38, 88], mesh-cutting [14], grafting [40], and sheet extraction [13].

Table 3.9. Satisfaction of the hexahedral mesh generation constraints by a whisker weaving algorithm.

Constraint Class	Description
Topologic	**Moderate to Strong**-Topologic constraints are handled directly using an advancing front method to 'weave' the chords of the mesh together consistently.
Boundary	**Strong**-Boundary constraints are typically satisfied first since the front starts at the boundary and works inward.
Quality	**Weak**-Quality is sacrificed in an effort to satisfy the topology and boundary constraints. It is common for the sheets defined by the boundary mesh to be highly complex resulting in sheets that will have high curvature and skew.

3.3.1 Inserting Sheets

Sheet insertion is typically used as a method for hexahedral refinement or mesh improvement. The most common methods for sheet insertion employ a pillowing operation [67] or a sheet copy operation [61, 60]. These methods, along with some additional applications, will be described in detail in this section.

3.3.1.1 Pillowing. During the development of the whisker weaving algorithm [101, 29], Mitchell et al. consistently encountered meshes where two neighboring hexes shared two faces (or more basically, where two adjacent faces shared two edges). Called a 'doublet' (see Figure 3.7), this situation is undesirable because of the impossibility of moving the nodal locations in these elements such that both elements have positive Jacobian determinant values. A simple, but powerful, technique called 'pillowing' [67] was developed to locate and place a mesh refinement that effectively removed the doublets from the mesh.

In terms of the dual of the mesh, pillowing is essentially a sheet insertion operation, where a new sheet is inserted that effectively splits the doublet hexes into multiple hexes, and eliminating the problematic mesh topology.

The pillowing method turns out to be powerful not only for the ability to remove doublets, but also for providing a fairly straight-forward approach to insert sheets into existing meshes. The sheets can be inserted utilizing the primal elements of an existing mesh, and without explicitly creating a geometric definition for the sheet and calculating

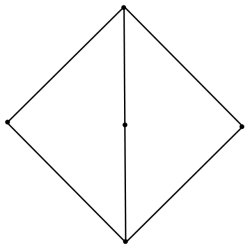

Figure 3.7. A quadrilateral doublet, where two adjacent quadrilaterals share two edges. Similar types of doublets occur in 3D with adjacent hexes sharing two or more quadrilaterals. The scaled Jacobian for both elements, as shown, is zero, and while node movement strategies can improve the Jacobian value for one of the two quadrilaterals, simultaneous improvement of the Jacobian value for both quadrilaterals is not possible.

the intersections with the other local sheets in the space.

The basic pillowing algorithm is as follows:

1. *Define a shrink set* - For our purposes, this step involves dividing the existing mesh into two sets of elements: one set for each of the half-spaces defined by the sheet to be inserted. One of these two sets of hexahedral elements comprises the shrink set. The choice of which one should be the shrink set is arbitrary, although the best algorithmic choice will be the set with the fewest number of elements.

2. *Shrink the shrink set* - This step essentially creates a gap region between the two previous element sets (see Figure 3.8). The difficulty in this step involves splitting the shared nodes, edges, and quads in the existing mesh, while maintaining the appropriate correspondence of the mesh entities with the geometric topology.

3. *Connect with a layer of elements* - This step results in a fully-conformal mesh with the new sheet inserted between the original two element sets. To complete this step, an edge is inserted between each node that was separated during the 'shrinking' operation. Utilizing the quadrilaterals on the boundary between the two sets of hexes, along with these new edges, it is fairly straight-forward to determine the connectivity of all of the hexes in this new layer.

It is often desirable to perform a smoothing operation on the resulting mesh after the new sheet has been inserted to obtain better nodal placement and higher quality elements. The speed of the pillowing algorithm is largely dependent on the time needed

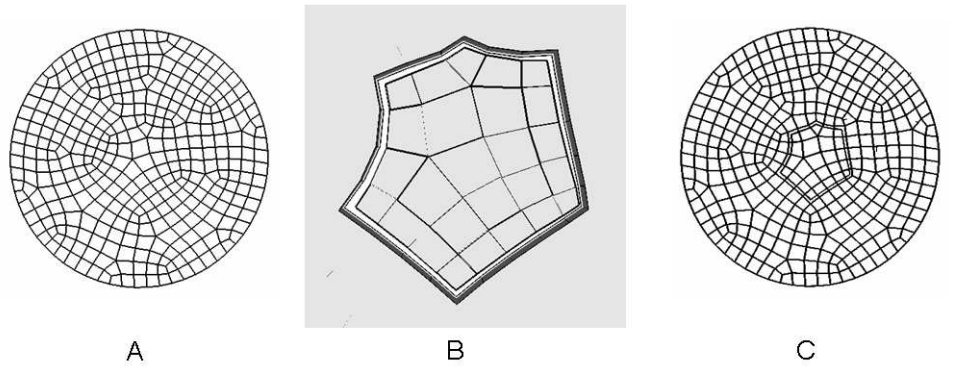

Figure 3.8. A basic pillowing operation starts with an initial mesh (A) from which a subset of elements is defined to create a shrink set. The shrink set is separated from the original mesh and 'shrunk' (B), and a new layer of elements (i.e., a dual sheet) is inserted (C) to fill the void left by the shrinking process.

to find the shrink set. The number of new hexahedra created will be equal to the number of quadrilaterals on the boundary of the shrink set.

A pillowing operation is nothing more than a sheet insertion operation for an existing mesh. It is therefore a useful multipurpose, foundational tool for operations on hexahedral meshes, including doublet removal, refinement, grafting, and mesh-cutting (grafting and mesh-cutting will be discussed in upcoming sections).

3.3.1.2 Dicing. The dicing algorithm [61, 60] was created to very efficiently generate very large, refined meshes from existing coarse meshes. The dicing method is an efficient tool for duplicating sheets multiple times within an existing mesh. The generation of these very large, refined meshes is accomplished by copying each of the existing sheets and placing them in a parallel configuration to the sheet being copied. The basic method for dicing is as follows:

1. *Define the sheet to be diced* - An edge in a hexahedral mesh can only correspond to a single sheet in the dual. Utilizing one edge, the opposite edges of the hexahedron can be deduced as belonging to the same sheet via the definition of the dual of the hexahedral mesh. It is then possible to iterate until all of the edges associated with a single sheet in the dual are found.

2. *Dice the edges* - With the list of edges found in the previous step, dicing then splits (dices) all of these edges the specified number of times. If for instance, we wish to

copy the sheet one time, then each of the edges is split once resulting in two new edges.

3. *Form the new sheets* - With each of the edges associated with the hexahedral sheet split, we can again utilize the idea that an edge can be associated with a single sheet in the mesh and form a new layer of hexahedra for each split in the original set of edges, where the hexahedral connectivity is similar to the original hexahedral layer before the edges were split.

Utilizing the dicing method, the number of elements increases as the cube of the dicing value. For instance, if an existing mesh is diced four times (i.e., each of the sheets in the existing mesh is copied four times), the resulting mesh would have 64X as many elements as the original mesh. Because all search operations can be performed directly, the dicing algorithm can produce large meshes at very efficient speeds (see Figure 3.9).

3.3.1.3 Generalized hexahedral refinement. Sizing in a hexahedral mesh is relative to the sheet density in a region of the hexahedral mesh. Reduced sizing and/or increased element counts within a region of the mesh can be realized by increasing the density of the sheets in the desired region. Because individual pillowing operations can be slow, refinement algorithms utilize precreated templates to increase the element density. These templates and the algorithms that insert them ensure manifold sheet structures within the mesh to maintain mesh validity. A basic refinement algorithm is constructed as follows:

1. *Define the region or elements to be refined* - The hexahedra in the refinement region are replaced with a predefined template of smaller elements. For example, a single

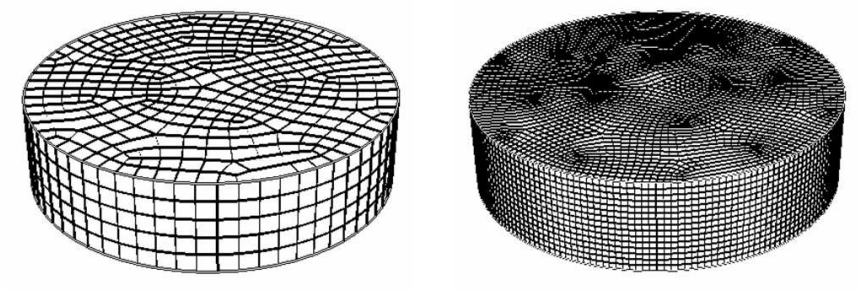

Figure 3.9. The original mesh (left) contains 1,805 hex elements before dicing. Each sheet in the original mesh is copied three times resulting in a mesh that is 3^3 larger, with 48,735 hex elements.

hexahedra is replaced by nine hexahedra occupying the same space as the original hexahedron. If all elements in the region are replaced with the same template, then the mesh will remain conformal everywhere except the boundary of the refinement region.

2. *Cap the boundary of the refinement region to ensure conformity to the rest of the mesh.* - A second set of templated elements which transition from the refinement template back to the original mesh size is inserted at the boundary of the refinement region. For example, a 3-to-1 transition template is inserted to cap a refined element and transition back to the original mesh size. There may be a variety of these templates (refer to [38] for more detail) for transitioning around the corners and within concavities of the refinement region (notice the number of templates utilized around the boundary of the refinement region in Figure 3.10).

3.3.2 Geometric Capture

In addition to mesh refinement, sheet insertion techniques can be used to capture features of the geometric boundary. We will highlight some procedures used to capture geometric features in a mesh in this section.

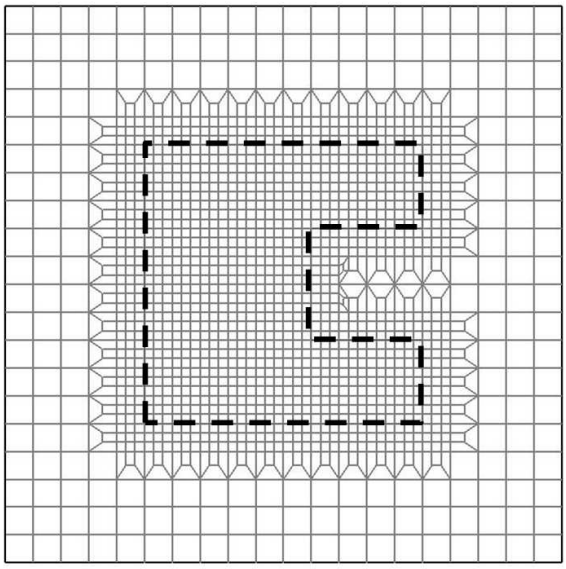

Figure 3.10. Refinement of a hex mesh using refinement templates. Refinement region is shown as a dashed line.

3.3.2.1 Mesh cutting. The mesh-cutting [14] method is an effective approach for capturing geometric surfaces within an existing mesh topology. The mesh-cutting method utilizes the pillowing and dicing methods mentioned previously to insert two sheets which are geometrically similar to the surface to be captured. By utilizing two sheets, the result is a layer of quadrilaterals, shared by the hexes in the two sheets, which can be viewed as a set of facets geometrically approximating the surface. The mesh-cutting method entails the following steps:

1. *Define the pillowing shrink set* - Utilizing the surface that is to be captured in the mesh, we divide the existing mesh into two sets of elements. One of these element sets will be the shrink set, and a sheet (pillow) is inserted between the two sets of elements.

2. *Dice the new sheet* - We split the newly inserted sheet into two sheets utilizing an approach similar to dicing.

3. *Move the shared quadrilaterals to the surface* - With two new sheets defined in the mesh topology, we can find all of the quadrilaterals that are shared by the hexes between the two sheets. These quadrilaterals become the mesh on the surface we are attempting to capture (see Figure 3.11).

A caveat with this method is that the existing mesh topology must be fine enough to capture the detail of the surface to be inserted. Because the resulting quadrilaterals

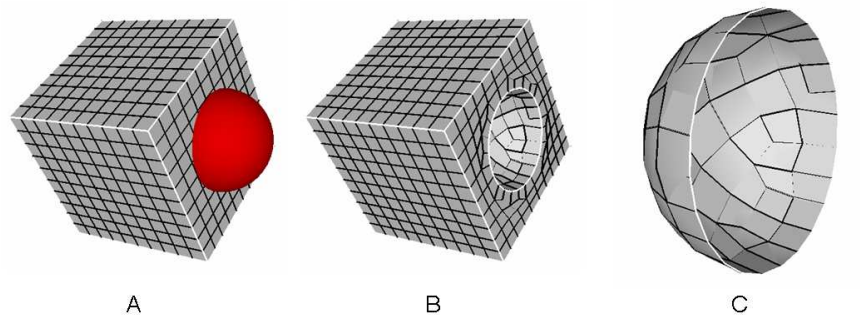

Figure 3.11. Mesh cutting utilizes an existing mesh and inserts new sheets to capture a geometric surface (the existing mesh is shown in (A) where the spherical surface is the surface to be captured.) The resulting mesh after mesh cutting is shown in (B), with a close-up of the quadrilaterals on the captured surface being shown in (C).

only approximate the inserted surface, if the resulting quadrilateral mesh is too coarse, the surface may not be approximated adequately.

One other item to remember with this method is that since a geometric surface is being utilized to define a sheet within the mesh space it may be necessary to have the ability to extend the geometric surface in some fashion such that it meets the requirements on a sheet that it divide the mesh space. If the geometric surface is trimmed, for instance, the trimmed surface may not adequately divide the space being meshed making it necessary to provide a continuation to the surface definition to the boundary of the mesh space.

3.3.2.2 Grafting. The term 'grafting' is derived from the process of grafting a branch from one tree into the stem, or trunk, of another tree. In meshing, the grafting method was initially to be utilized for allowing a branch mesh to be inserted into the linking surface of a previously hexahedrally swept volume [40]. The grafting method would then offer a reasonably generalized approach to multiaxis sweeping.

Grafting is a method that essentially captures geometric curves on previously meshed surfaces. Because the curve(s) already reside on a surface and the existing mesh already captures that surface, it is possible to introduce a second sheet to satisfy the curve constraint listed earlier. The introduction of the new sheet will produce the necessary mesh topology to enable the mesh to be compatible with the boundary curve(s). The method for creating a graft (i.e., capturing the geometric curve) can be outlined as follows (see also Figure 3.12):

1. *Create a pillowing shrink set* - In the case of grafting, the shrink set is typically defined as the set of hexes which have one quadrilateral owned by the surface and which are interior to the closed set of curves (with respect to the surface).

2. *Insert the pillow (sheet)* - By inserting the second sheet, we have essentially satisfied the hexahedral constraint for capturing a geometric curve. That is, we now have two sheets which generate a chord in the mesh that is offset from the set of curves which were the input to the grafting algorithm.

At this point, there are often some database adjustments also necessary to ensure that the new mesh entities are associated with the correct geometric entities, but the curve is essentially captured when the second sheet is inserted in conjunction with the initial set of sheets that captured the geometric surface. A similar method can be used to capture a single curve, rather than a set of curves, but it is still necessary that the sheet that is

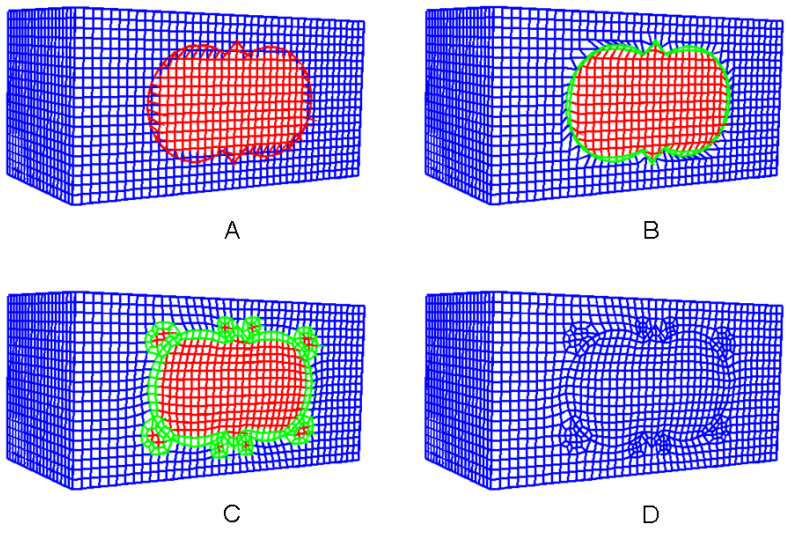

Figure 3.12. In grafting, a shrink set on a existing meshed volume is defined (A) and a new sheet is inserted via a pillowing operation (B). Once the new sheet has been inserted, the nodes are positioned along the curve to be captured via a smoothing operation (C). Additional pillows can also be inserted to remove any doublet hexes that may have been created (C). The resulting mesh topology captures the geometric curve (D).

inserted to capture the curve satisfy the definition of a sheet. That is, the sheet must completely divide the mesh space into two regions.

There is one caveat with this method: Because there are no explicit steps taken to capture the geometric vertices associated with each of the curves being grafted, there is a requirement that the resolution of the trunk mesh be fine enough to be able to capture all of the curve's endpoints by moving existing nodes in the final mesh to the vertex locations. The movement of the nodes to the vertex locations must be done intelligently to avoid destroying the required mesh topology necessary to correctly capture the curve. While other sheets may be added to avoid this problem, the addition of more sheets to capture the vertices may have the negative effect of locally refining the mesh sizes and/or mesh topologies that are not as aesthetically pleasing as may be desired.

3.3.3 Removing Sheets

In addition to inserting sheets into a mesh, it is often desireable to coarsen an existing mesh. In this section, we will outline a method for removing sheets of hexahedra from an existing mesh.

3.3.3.1 Sheet extraction. One of the positive practical benefits about working with the sheets in hexahedral meshing is that all of the processes are easily reversible. Sheet extraction is the inverse operation to a sheet insertion operation (i.e., pillowing or dicing). A method for extracting a sheet is detailed in [13], where the basic steps can be outlined as follows:

1. *Define the sheet to be extracted* - Because an edge in a hexahedral mesh can only correspond to a single sheet in the dual, this step can be easily accomplished by specifying a single edge in the mesh. From this single edge, the primal mesh can be iteratively traversed to determine all of the edges which correspond to the sheet to be extracted.

2. *Collapse the edges* - With the list of edges found in the previous step, the nodes of each of the edges can be merged, effectively removing the sheet from the mesh.

There are some special circumstances that must be avoided when extracting sheets in order to avoid either degenerating the mesh or producing a mesh which is no longer conformal with the geometric topology. These situations can be avoided by checking to ensure that one of the edges to be collapsed is not the only edge on a curve, or that the nodes in an edge are not owned by different curves, etc. An example mesh that has been modified by extracting several sheets to obtain a coarser mesh is shown in Figure 3.13.

3.4 Mesh Smoothing Methods

In the process of generating, modifying, cleaning up, or adapting meshes, it is possible to create distorted or poorly shaped elements. It is desirable, therefore, to have algorithms

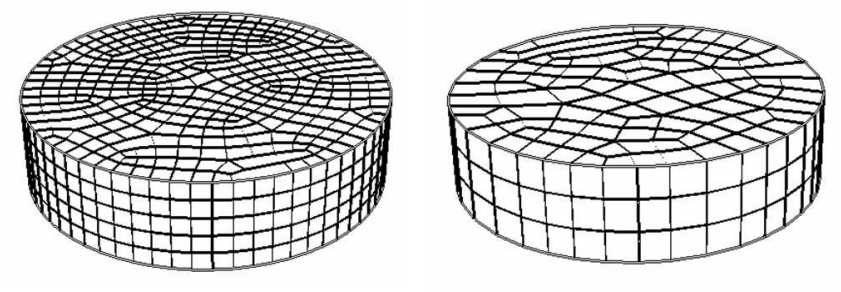

Figure 3.13. Original mesh, shown on left, with 1805 hex elements. After removing approximately half of the sheets in the original mesh, the resulting mesh (right) has 342 hex elements.

available for improving existing meshes. Over the years, many mesh smoothing algorithms have been developed. We will outline a few that have been utilized in the course of this research.

3.4.1 Mesh Smoothing

While there are a wide variety of smoothing algorithms available, including Laplacian [37, 28], centroidal area [41], Winslow [45], angle-based [126], and many others, we will highlight two algorithms that have been utilized in this research.

3.4.1.1 Laplacian smoothing. Of all the available mesh smoothing algorithms that are currently available, the most common algorithm is known as Laplacian smoothing. The easiest way to visualize Laplacian smoothing is to consider each edge attached to a node in the mesh as a spring. When the lengths of each of the edges attached to the node are identical, the spring force is balanced. Because this process iterates over each of the nodes, the spring forces may be constantly changing after each iteration, while the total force in the system should be dimished with each iteration.

The advantages of Laplacian smoothing are that the algorithm is easy to implement and is computationally efficient. However, it also has several disadvantages, including that it may generate inverted elements, element shapes are not necessarily optimized, and features may not be preserved if too many iterations are performed. Despite these disadvantages, Laplacian smoothing is a very powerful mesh optimization tool, especially when coupled with other optimization-based smoothing techniques [30].

3.4.1.2 Centroidal area smoothing. Another popular mesh smoothing algorithm is centroidal area smoothing. This algorithm is somewhat similar to Laplacian smoothing, but instead of pulling the node to equal edge lengths, the node is moved to the weighted average of the centroid of neighboring elements. The nice thing about this smoothing algorithm is that the features in the mesh tend to be largely preserved after smoothing [41].

3.4.2 Mesh Untangling

A mesh untangling algorithm uses node movement to remove nonconvexities of hexahedra within a mesh. We introduce the idea of mesh untangling as developed by Knupp, et al. [44]. As described earlier, the Jacobian matrix is calculated for each node with respect to a hexahedral element. For each node in a hex, there are exactly three 'neighbor' nodes

connected via an edge in the hexahedra. For a single node of a hexahedra, the Jacobian matrix is defined as:

$$A_0 = \begin{vmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{vmatrix}$$

For a single hex, there will be eight such matrices (for additional discussion on elements with multiple Jacobian matrices see [47]. The minimal determinant of these eight matrices is known as the 'Jacobian' of the hexahedral element. By allowing nodal movement for each of these elements, an optimization problem can be formulated to maximize the following objective function (other similar functions have also been utilized):

$$f(A) = 0.5 * \sum(|\alpha_m| - \alpha_m)$$

where α_m is the determinant a single Jacobian for a mesh with m elements.

If the mesh is untangled, then the summation reaches a maximum value of zero. For a hexahedral mesh, this can be a difficult optimization. It is common to use local optimization algorithms, such as conjugate gradient methods, to obtain a solution to the untangling optimization problem. However, because the untangling problem is nonconvex, it is possible to reach a local maxima without obtaining an optimal solution. This is an ongoing and challenging research area [44, 106, 49, 31]. It is undetermined whether a mesh that satisfies the topological requirements cited earlier can always be untangled, although it is believed that there are meshes that do not have an untangled solution [44].

3.4.3 Mesh Optimization

In addition to traditional mesh smoothing techniques, there has been a tremendous amount of research conducted in optimization-based smoothing. Traditional smoothing methods work heuristically and can create invalid meshes or elements containing worse quality than the original mesh. In contrast, optimization-based methods work to optimize a metric value associated with each of the mesh elements. Common metrics for optimizing include shape [48], condition number [44], Jacobian [46], and mean ratio [26, 47]. While these methods can be extremely effective at maximizing metric values, they can also be very computationally expensive. To reduce the computational expense, hybrid algorithms have been proposed which combine some of the speed advantages of the traditional methods while maintaining the quality improvement guarantees of the optimization-based techniques [30].

CHAPTER 4

FUNDAMENTAL HEXAHEDRAL MESHES

In this chapter we explore more fully how to capture geometric boundary features using elements in the dual. We will first make several observations about how geometric boundaries interact with dual elements, and then elaborate on the concept of a 'fundamental' mesh. We will demonstrate methods for converting an existing mesh to a fundamental mesh, as well as converting one fundamental mesh to another fundamental mesh. The concept of a minimal hexahedral mesh will be presented with a description of how the set of fundamental meshes are related to the minimal mesh.

4.1 Observations

A couple of key observations can be made from Table 2.3. Chords are drawn between centers of hexahedra, the chord between two centroids is dual to the quadrilateral shared by these two hexahedra. In some sense, the chord can be viewed as an approximation to the normal of the quadrilateral between two hexes.

4.1.1 Geometric Surfaces

Observation 1: The boundary of any hexahedral mesh is a quadrilateral mesh.

To improve the quality of the mesh at the boundary of our geometry, it is desirable to align the chords with the local surface normals. Doing this promotes higher quality hexahedral elements on the boundary of the mesh.

Given a hexahedral mesh and utilizing Observation 1, we can select a quadrilateral on the boundary. This quadrilateral is dual to a chord that is found at the intersection of two sheets whose boundary are the two chords intersecting the quadrilateral on the boundary. Since the boundary quadrilateral is contained in only one hexahedral element of the mesh, there must be a third sheet that intersects the other two creating the centroid that is dual to the hex. On this third sheet, there is an area, or patch, on the sheet that can be said to correspond directly with the quadrilateral on the boundary. This area of the sheet is also

geometrically similar in shape to the quadrilateral, although offset a distance based on the size of the hexahedron to which the sheet and quadrilateral correspond. A collection of contiguous areas on a single sheet corresponding to a set of quadrilaterals on the boundary will be defined as a ***'sheet patch'***. The collection of 'sheet patches' corresponding to all quadrilaterals on the surface boundary can be considered to be geometrically similar to the collection of quadrilaterals with which they are associated. (see Figure 4.1). For a single surface, the minimal number of sheets to which this collection of patches might belong is a single sheet.

This poses an immediate question: Is it better to capture a boundary surface with a single sheet, or with multiple sheet patches? For continuous surfaces, the answer to this question is that single sheets for capturing a surface are better than multiple sheet patches. The reasoning for this is due to the irregularities introduced whenever a sheet is diverted away from a surface. When partial sheets are utilized to capture a surface, the area of the sheet where the transition occurs results in increased nodal valence in the mesh, skewing of the elements (caused by the sheet curvature), and in some cases the formation of adjacent hexahedron that contain faces sharing two or more edges (also known as a 'doublet'). Figure 4.2 shows several examples of the transition elements resulting from partial sheets capturing boundary surfaces. Therefore, in most cases, the use of a single sheet to capture boundary surfaces is preferred.

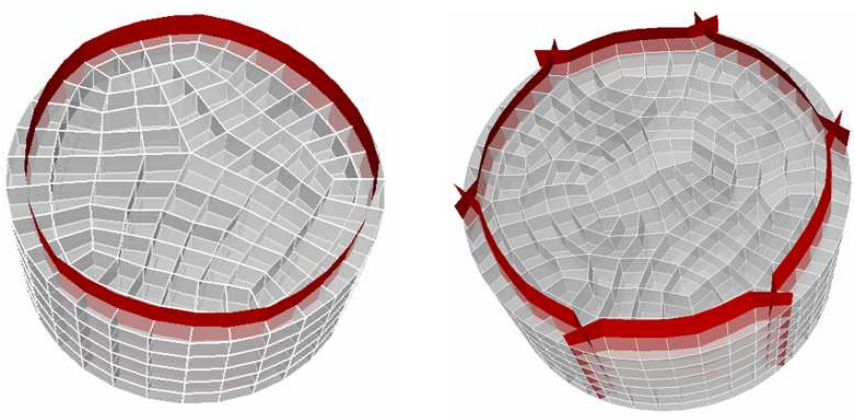

Figure 4.1. Image showing how a sheet captures the geometric boundary. The picture on the right shows a single sheet capturing the cylindrical surface, while the picture on the left (of a different mesh) shows the same surface being captured with multiple sheets.

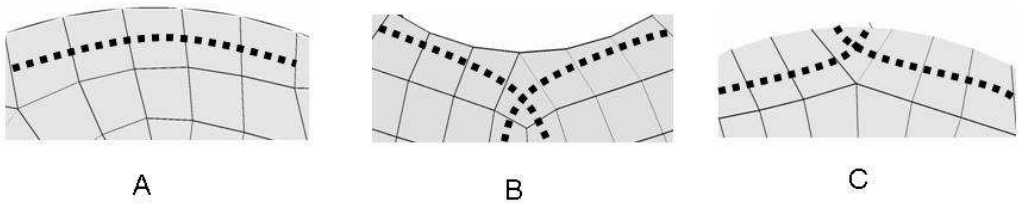

Figure 4.2. When partial sheets capture boundaries, the quality and regularity of the mesh is affected. Image A shows elements from a single sheet capturing the upper boundary of the solid. Image B and C use patches from two sheets to capture the upper boundary of the solid. In Image B and C, note how the regularity of the mesh is affected, and the resulting skew in the transition element due to the sheet curvature away from the boundary. In Image C, a near doublet element is formed due to the low curvature of the boundary being captured.

4.1.2 Geometric Curves

From the boundary constraints outlined in Chapter 2, every curve in the solid geometry must have a set of simply connected edges that approximate the curve. A set of hexes stacked end-to-end, which corresponds to a chord in the dual, will have four sets of simply-connected edges. The chord is formed by the intersection of two sheets. Chordal segments, then, can be utilized to ensure that a set of simply connected edges will exist to capture the geometric curve on the boundary. The lines of intersection of two sets of sheet patches are a set of chord segments that are geometrically similar to the curve at the boundary between the two surfaces, and that are offset a distance that is a function of the mesh size in the boundary curve's locale (see Figure 4.3).

Figure 4.4 shows a case of a curve (shown in black) being captured by two chordal segments. The first chordal segment (the intersection of the green and red sheets) produces a string of mesh edges capturing the upper portion of the curve, while the second chordal segment (the intersection of the yellow and red sheets) result in the string of mesh edges on the lower portion of the curve.

4.1.3 Geometric Vertices

For every geometric vertex in the geometry, there must be at least one node in the mesh that is associated with that vertex. There are eight nodes per hexahedron, and one of those nodes can be associated with the vertex. This implies that there is also at least one hex associated with each vertex.

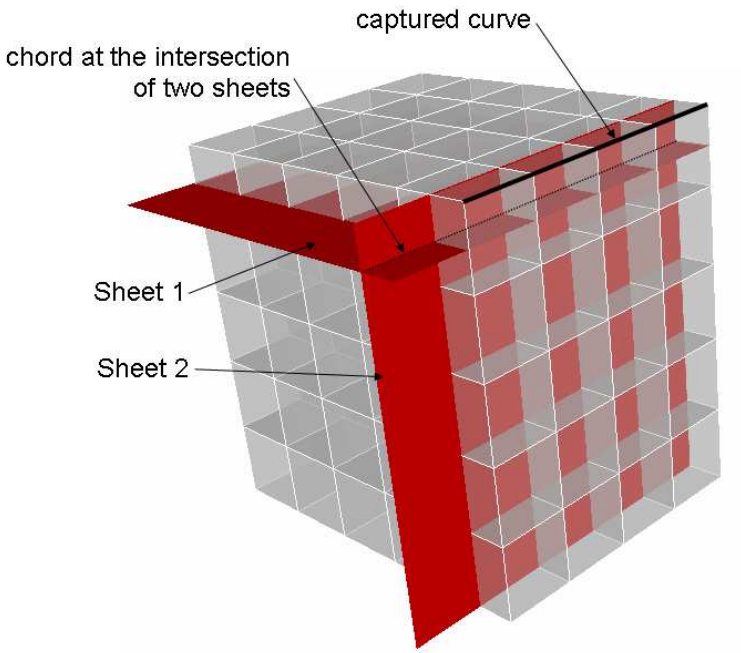

Figure 4.3. Capturing a curve utilizing an offset chord (from the intersection of two sheets).

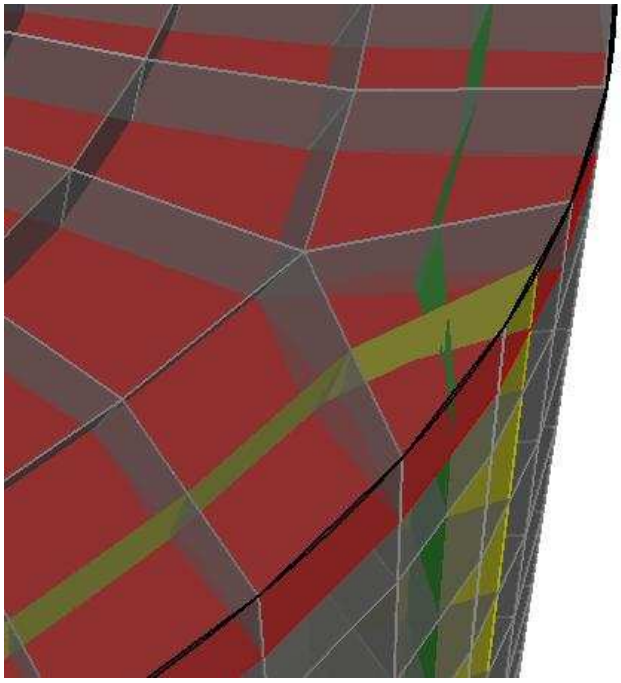

Figure 4.4. Capturing a curve utilizing an multiple chordal segments.

Because a node on a hex is always trivalent (i.e., it always has only three edges connected to the node on that hexahedral element), we must increase the number of hexes associated with the geometric vertex whenever the vertex valence is increased to values greater than three. The minimal number of hexes associated with a geometric vertex then must be:

$$max(v - 2, 1)$$

where v is the vertex valence (i.e., the number of geometric curves emanating from the vertex).

We can rewrite this observation in terms of sheets, such that, for each vertex in the geometry there exists at least one local, triple-set of sheets whose intersection form a centroid that corresponds to the vertex. The number of centroids associated with the vertex will increase as the vertex valence increases. A few examples are shown in Figure 4.5).

In this case, utilizing the sheets in meeting this constraint is helpful instead of the centroids because there are usually geometric curves emanating from each of the vertices.

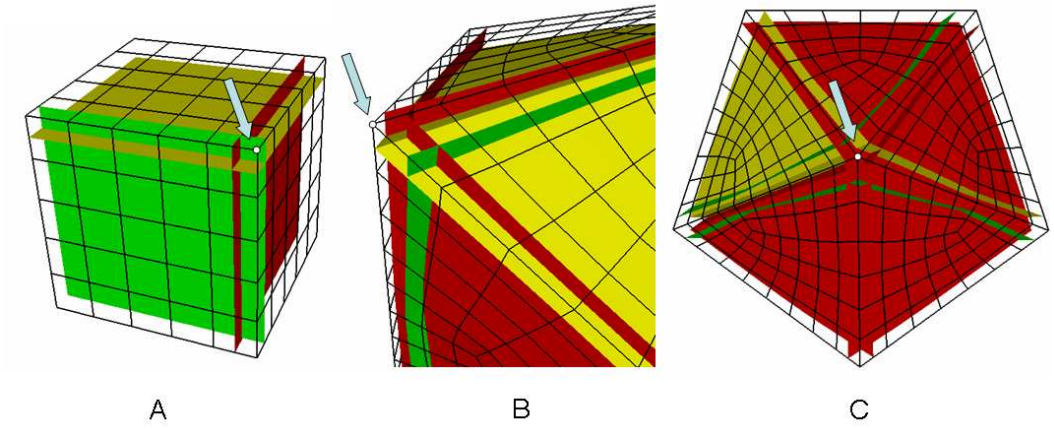

Figure 4.5. At least three sheets are necessary to capture a geometric vertex in a given mesh topology (A). However, more than one triple pair of sheets is not exclusive for each vertex. For geometric vertices whose valence is higher than three, more than one triple pair is necessary to capture all of the geometric features related to the vertex. A four-sided pyramid (B) requires four sheets (creating two distinct centroids, or two triple-pairs) and a five-sided pyramid (C) requires five sheets (creating three distinct centroids, or three triple-pairs) to succinctly capture the geometric features associated with the vertex. In (B), there are two red sheets, one green, and one yellow shown. In (C), there are two red, two green and one yellow sheet shown.

The sheets utilized to capture the geometric surfaces and curves will likely be the same sheets that capture the vertices. However, the convergence of many sheets around some vertices must be handled with care to maintain the topological constraints local to a vertex (i.e., only three sheets can intersect at a single point, etc.).

4.2 Fundamental Meshes

In wave theory, an object that vibrates has a collection of natural frequencies that form standing wave patterns within the vibrating object. A commonly referred to example is a guitar string, which is a string with the two endpoints fixed at opposite ends of the guitar. The lowest frequency wave that can be formed on the guitar string has a wavelength that is twice the length of the string, and is known as the first harmonic, or the fundamental frequency, of the guitar string. Other harmonics exist within the string and can be identified by the integer number of anti-nodes within the wave (refer to Figure 4.6).

4.2.1 Fundamental Meshes

Definition: A ***fundamental mesh*** is a hexahedral mesh that contains one sheet for every surface, at least one continuous two-sheet intersection (chord) for every curve, and *(vertex valence - 2)* triple-point intersections (centroids) for every geometric vertex.

Definition: As defined earlier, a ***'sheet patch'*** is a collection of contiguous areas on a single sheet corresponding to a set of quadrilaterals on the boundary.

Definition: A ***boundary sheet*** is a sheet that contains at least one *sheet patch* directly associated with a quadrilateral on a geometric surface boundary.

Definition: A ***fundamental sheet*** is a sheet that contains all *sheet patches* associated with the quadrilaterals of one or more surfaces, or is a sheet that is required to satisfy a topologic constraint, as defined in Chapter 2, with other fundamental sheets.

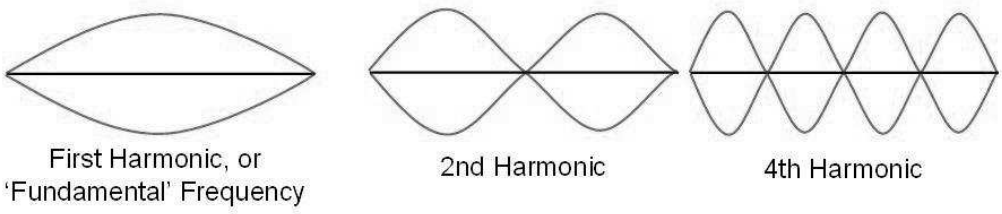

Figure 4.6. Harmonic waves on a string.

At the boundary of a mesh, the arrangement of sheets also form patterns (see Figure 4.7). Any sheet that has a sheet patch associated with a quadrilateral on a geometric surface boundary will be termed a 'boundary' sheet. When a single sheet is associated with all of the quadrilaterals on a surface, we will refer to this as a 'fundamental' sheet for that surface.

Similarly, chords segments associated with curves on the geometric boundary will be termed 'boundary' chords. If a single chord is associated with all of the mesh edges on a curve, this chord will be a 'fundamental' chord in the mesh. Additionally, there will be boundary centroids and fundamental centroids associated with each vertex in the geometry.

4.2.2 Hexahedral 'Flipping'

In tetrahedral and triangle mesh generation, there is a technique known as 'edge-flipping' [33, 39], that modifies the topology of the tetrahedral/triangle mesh. The basic idea is shown in Figure 4.8, where an edge shared between two elements in the mesh is 'flipped' to an alternate, but equally valid, topological state and the validity of the mesh is retained. The benefits of edge flipping include removal of sliver elements, improvement in overall mesh quality (including small angles), and mesh topology that may be more suited to a given geometry.

Bern et al. [6, 7] have expanded this concept for hexahedral meshes where a set of cells whose boundary matches a known flippable primitive is replaced with one of a set of known new topologies. Tautges [102] has further expanded on this idea and attempts to demonstrate that all flip operations in hexahedral meshes are derived around a set of atomic (i.e., simple, foundational) operations in the dual of the hexahedral mesh. While none of these algorithms have been presented as a practical method for improving existing

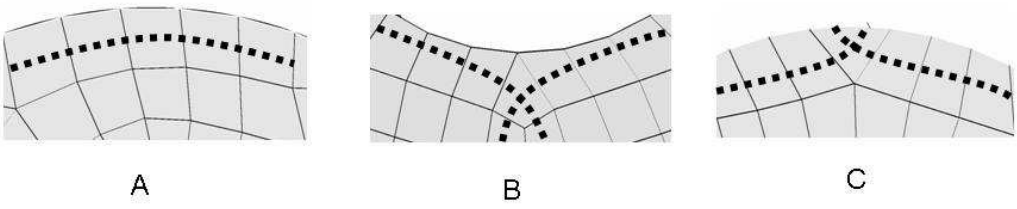

Figure 4.7. Image A shows a 'fundamental' sheet for the surface; images B and C show boundary sheets that are not fundamental with the associated with the surface.

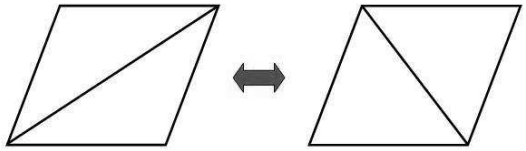

Figure 4.8. Flipping edges in a triangle mesh.

hexahedral meshes, many similar flip-like operations have been used to modify or change meshes (for instance, as a preliminary step to the whisker weaving algorithms [29, 69].

While Bern et al. [6, 7] and Tautges [102] provide a more complete list of possible flip operations in hexahedral meshing, we will highlight two operations that demonstrate a 'flip' in the topology of the hexahedral mesh. These operations coupled with sheet insertion (pillowing) and extraction operations will enable us to modify boundary sheets to obtain fundamental sheets in a hexahedral mesh.

4.2.2.1 Face collapse. Face collapsing involves collapsing a quadrilateral in a mesh by merging two opposite nodes on the quadrilateral. For surface meshes, this procedure is outlined by Folwell [29]. In a hexahedral mesh, all of the faces in the chord need to be collapsed to avoid degenerating the elements in the hexahedral mesh. An example of a face collapse operation is shown in Figure 4.9.

The powerful thing about this operation is the change in the sheet structure of the mesh (see Figure 4.10). One way to view this operation is as a rotation in the connectivity of the sheets. Instead of connecting to the opposite edge across the original quadrilateral, this operation rotates the connectivity and removes the crossing. It is also be possible to

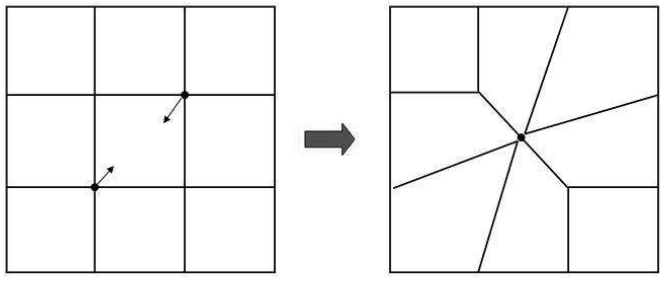

Figure 4.9. Face collapse operation.

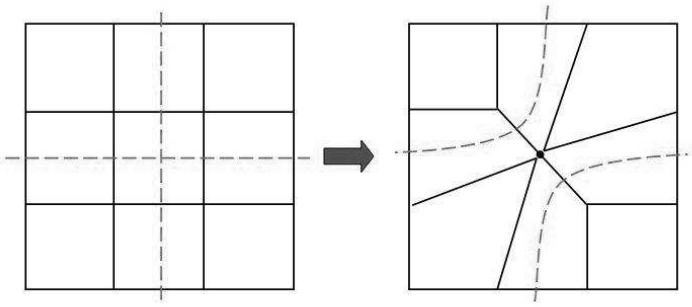

Figure 4.10. Sheet 'flips' due to face collapse operation.

make this process reversible with a face open operator.

4.2.2.2 Boundary face collapse. The boundary face collapse operation is similar to the face collapse operation, except that instead of merging just two nodes, three nodes are merged into a single node. This is possible when there are nodes are on the boundary of the mesh. Figure 4.11 shows two examples at different boundary locations in a mesh.

The resulting sheet structure changes are similar to the face collapse operation, however because a third node is involved in the merge, there is not a second sheet affected by the rotation. Instead the two original sheets become a single sheet (see Figure 4.12).

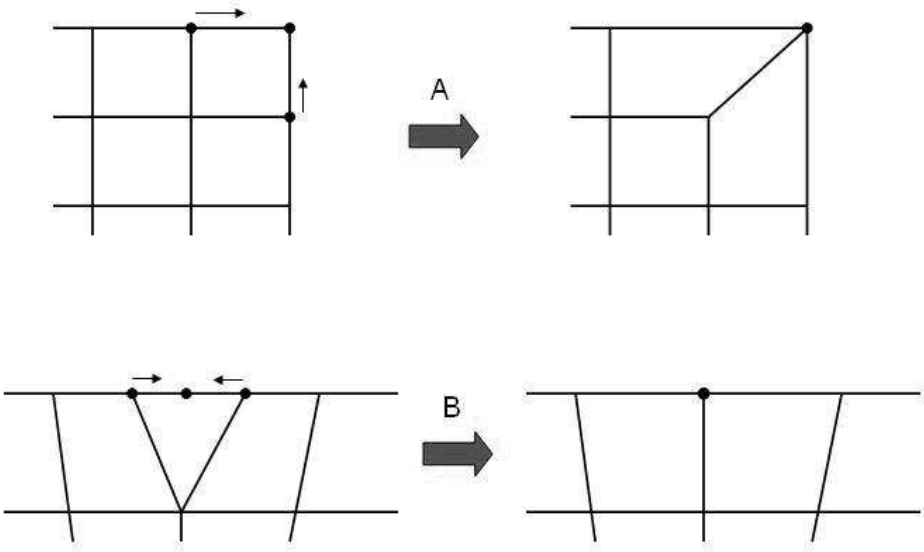

Figure 4.11. Two examples of a boundary face collapse operation.

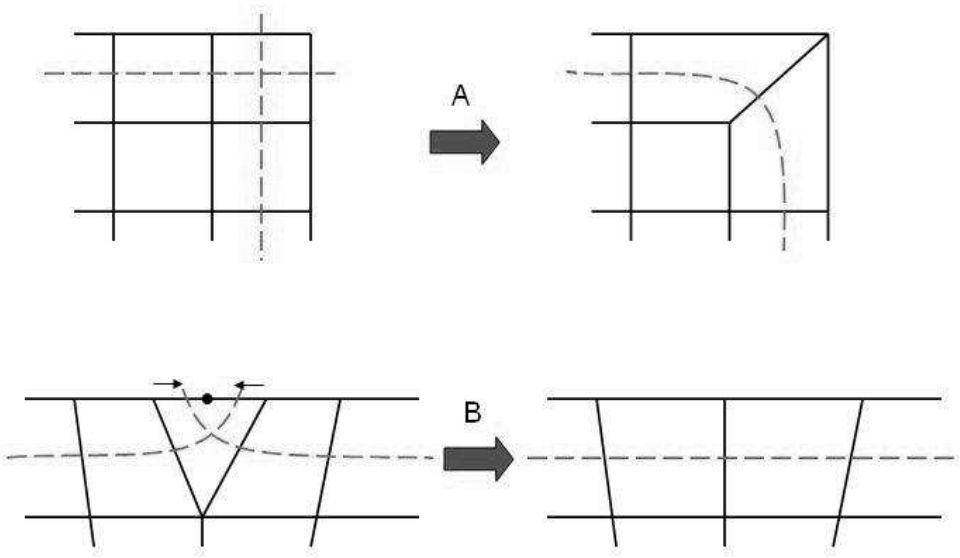

Figure 4.12. Sheet 'flips' due to the boundary face collapse operation.

4.2.3 Converting to a Fundamental Mesh

The operators described above, along with some possible others, can be utilized to convert nonfundamental sheets to fundamental sheets. We make the following assertion:

Assertion 1: For any given hexahedral mesh of a geometric object, there exists a set of transformations that converts the set of boundary sheets into a set of fundamental sheets for the geometry.

Rationale: Utilizing the flip operations described above, we can show that this conversion is possible for 2D quadrilateral meshes. If the collapse operations are performed on an entire chord, the 2D operations are easily extended to include all structured meshes (e.g., mappings, blockings, and sweeps). The assertion should also be true for unstructured meshes, and the existence of such transformations is easily shown using a sheet insertion operation for each surface on the boundary. Proof of this assertion is an area of future work.

An example of converting a volume with a set of boundary sheets into a set of fundamental sheets is shown in Figure 4.13. An additional example will be shown later in the chapter. These examples are shown for quadrilateral meshes, but are easily extended to hexahedral meshes when the collapse operations are performed on the entire chord. Because the collapse operations remove edges in the mesh, there may be some cases where additional refinement is needed to complete the operation. Sheet insertion

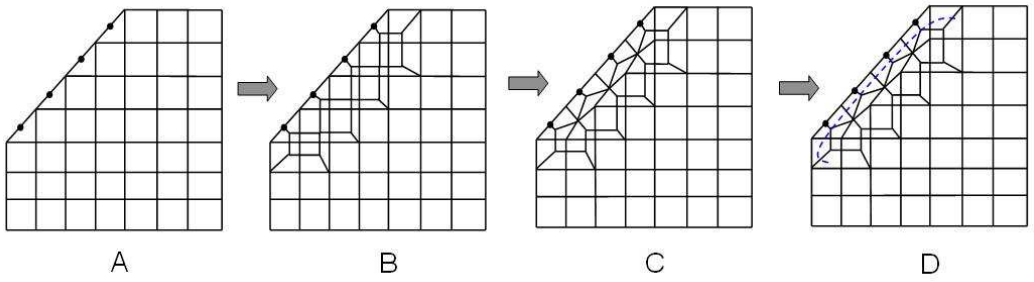

Figure 4.13. Transforming boundary sheets into fundamental sheets. Apply pillowing to nodes in A to get B, apply face collapses in B to get C. Image D shows the new fundamental sheet for the sloping surface.

(pillowing) is utilized in the example in Figure 4.13 to maintain mesh validity throughout the transformation process. Some additional face collapse operations on the final mesh, along with sheet extraction, can be performed afterwards to further clean up the mesh.

4.2.3.1 Fundamental mesh conversion example. Figure 4.14 demonstrates an example of a mesh of a thin region that was created with a paving algorithm [8] and extruded into a hexahedral mesh with a sweeping algorithm [43]. In thin regions, it is typically desirable to obtain a mesh with high regularity and low element skew. Due to the seaming heuristics employed by the paving algorithm, it is typical in thin regions to create elements like those shown in Image A of Figure 4.14. We can improve this mesh to obtain the desired characteristics by converting the boundary sheets into fundamental sheets throughout the thin region. An example of this process is shown in Images B and C of Figure 4.14.

4.2.4 The Set of Fundamental Meshes

With very few exceptions, a given geometric object will have multiple sets of sheets that satisfy the definition of a fundamental mesh (see Figure 4.15).

Assertion 2: There exists a transformation that converts one set of fundamental sheets into an alternative set of fundamental sheets.

Rationale: Refer to Figure 4.16 to explain this assertion. The definition of a fundamental mesh requires a single sheet for each surface on the boundary, a two-sheet intersection (chord) for each curve, and a variable number of triple-sheet intersections (centroids) for the vertices in the geometry. We can draw this situation schematically as shown in Image A of Figure 4.16. Depending on how these schematic pieces connect with

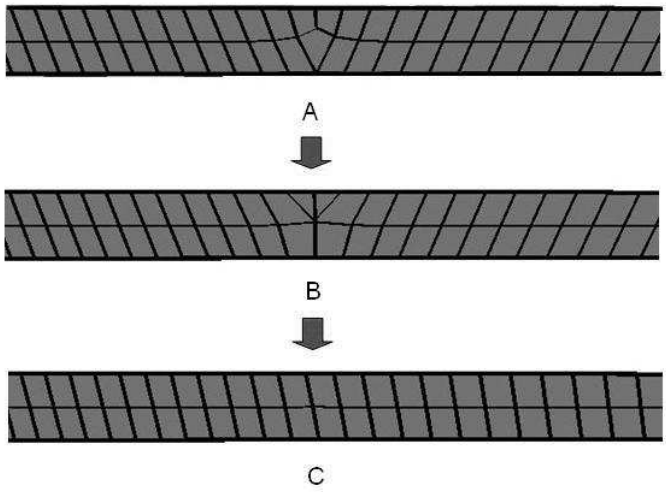

Figure 4.14. Cleaning up nonfundamental sheets in thin regions. Image A is the original mesh, image B is the mesh after a face collapse operation, image C shows the fundamental mesh following two additional face collapse operations.

one another gives us the various sets of fundamental sheets for a given geometric solid (as shown in Image B, C, and D in Figure 4.16). Defining an operation that changes the connectivity enables one fundamental mesh into an alternate fundamental mesh for the given geometry is needed to prove Assertion 2. The collapse operations defined earlier perform the desired rotations to the topology, but have the detrimental affect of coarsening the mesh, and may destroy needed topology elsewhere in the mesh.

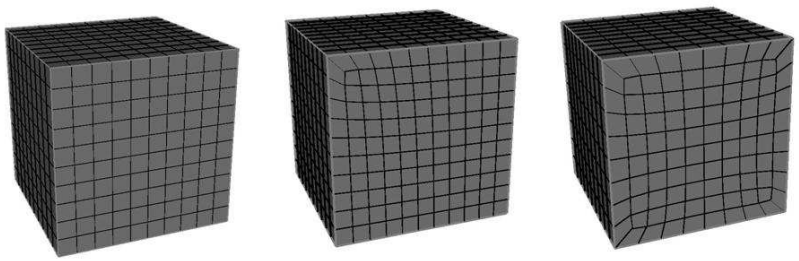

Figure 4.15. Differing sheets sets may satisfy definition of a fundamental mesh. The image on the left has six fundamental sheets capturing surfaces (one for each surface of the cube). The image in the middle has five fundamental sheets capturing six surfaces, and the image on the right has three fundamental sheets capturing six surfaces. All meshes shown satisfy the definition of a fundamental mesh.

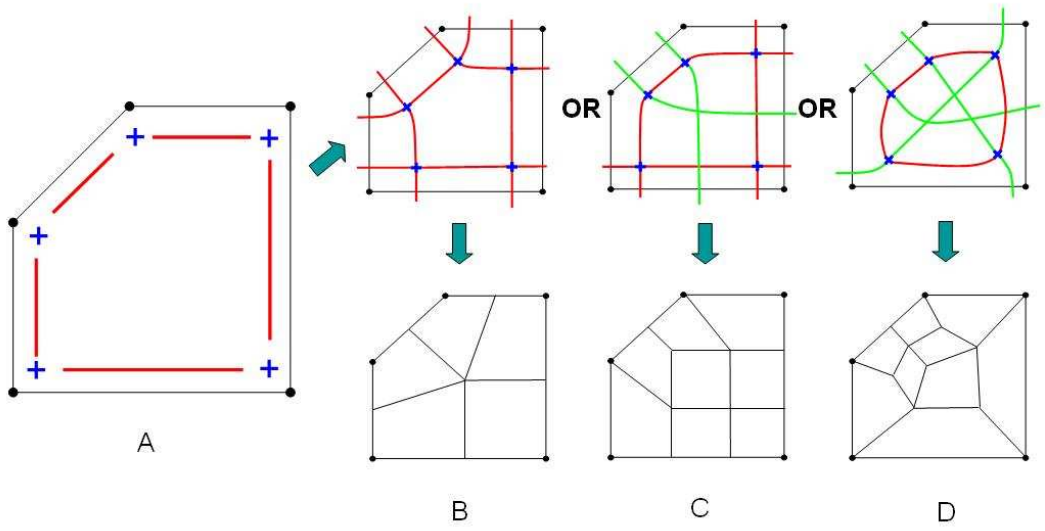

Figure 4.16. Fundamental transformations. Image A shows necessary elements for a fundamental mesh of the geometry (sheets for each surface (red) and sheet intersections (blue) for each curve). Depending on how these fundamental pieces are connected results in differing fundamental meshes for the geometric object (shown in image B, C, and D). The green sheets are additional sheets needed to satisfy topologic constraints for the hexahedral mesh.

Again, utilizing the flip operations described above, we can show that conversions from one fundamental form to another are possible for 2D quadrilateral meshes. If the collapse operations are performed on an entire chord, these 2D operations are easily extended to include all conversions for all structured meshes (e.g., mappings, blockings, and sweeps). The assertion should also be true for unstructured meshes, and the existence of such transformations is easily shown using a sheet insertion operations to produce the desired fundamental set of sheets for the mesh.

Assertion 2 states that there is a transform operation that will convert one fundamental mesh into another fundamental mesh for the geometric object. As shown in Figure 4.16 creating one fundamental mesh from another fundamental mesh involves 'flip' operations for reconnecting each of the sheets to create the desired fundamental mesh.

4.2.4.1 Multivolume meshes. When considering fundamental meshes for collections of volumes, it is important to remember that sheets are manifold within the space occupying the mesh. As stated earlier, a volume defines a subspace into which the mesh is placed. If more than one volume exists, then the collection of volumes defines the entire space being meshed. The sheets must be manifold in the space (not just in

each subspace). Therefore, when considering the fundamental mesh for a single volume within a collection of volumes, there may be additional sheets present within the volume that result from fundamental requirements in adjacent volumes.

This can be seen in Figure 4.17 where two adjacent volumes are shown. A set of fundamental sheets is shown (in blue) for the volume on the left, while a separate set of fundamental sheets is shown (in red) for the volume on the right. Two of the fundamental sheets in the volumes can be merged at the boundary of the two volumes (i.e., the top- and bottom-most sheets in both volumes can be joined to become a single manifold in the space defined by both volumes). However, there is a third sheet which is necessary in the volume on the right to satisfy fundamental mesh requirements, but is unnecessary

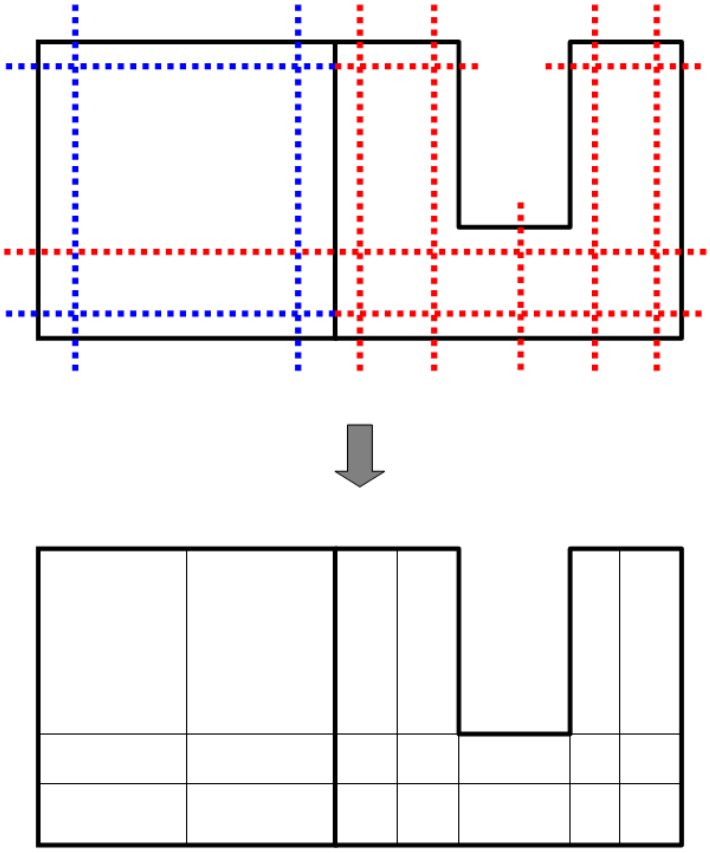

Figure 4.17. Fundamental sheets in collections of volumes. Because sheets must be manifold in the space occupied by the mesh, one of the sheets from the volume on the right must be extended through the volume on the left to satisfy all hexahedral requirements for the mesh of both volumes (shown at bottom).

for the volume on the left. When meshing both volumes together, this sheet must be manifold in the space. In the figure, we show this sheet continuing through the volume on the left, producing the meshes shown in the bottom of Figure 4.17.

4.3 Minimal Meshes

The fundamental mesh of a geometric object creates the necessary mesh topology to match the geometric topology with hexahedral elements. We now demonstrate a method to minimize the hexahedral elements required to mesh a geometric object.

4.3.1 Secondary Sheets

In addition to fundamental sheets and boundary sheets, there are other sheets in a hexahedral mesh that are not associated with the boundary, or do not satisfy the requirements to be defined as fundamental sheets. We will call define these additional sheets to be 'secondary sheets'.

Definition: A ***secondary sheet*** is a sheet in a mesh that is not a boundary sheet or a fundamental sheet in that mesh. Secondary sheets are typically utilized to meet shape/size requirements within the final mesh.

Figure 4.18 shows an example cube with the fundamental sheets drawn in red (on the left) (with the exception of the fundamental sheet for the front surface), and the secondary sheets drawn in green (on the right). Utilizing a sheet extraction algorithm, we can remove the secondary sheets from a mesh without affecting the underlying mesh topology that is required to maintain conformity of the mesh with the geometric topology of the solid object. The resulting mesh with the secondary sheets removed is shown in Figure 4.19.

4.3.2 Minimal Hexahedral Meshes

The mesh shown in Figure 4.19 is not the minimal mesh for the geometry (i.e., the hexahedral mesh with the fewest number of elements that still captures all of the geometric entities in the solid geometry). However, the remaining sheets in the mesh are fundamental sheets. When the half-space defined by two sheets are overlapping, fundamental sheets can in some instances be merged into a single sheet to further simplify the mesh. The resulting mesh will still be fundamental and the mesh is minimized further.

We stated previously that there may be more than one fundamental set of sheets for a geometric object. Allowing for fundamental sheets to be merged, we make the following

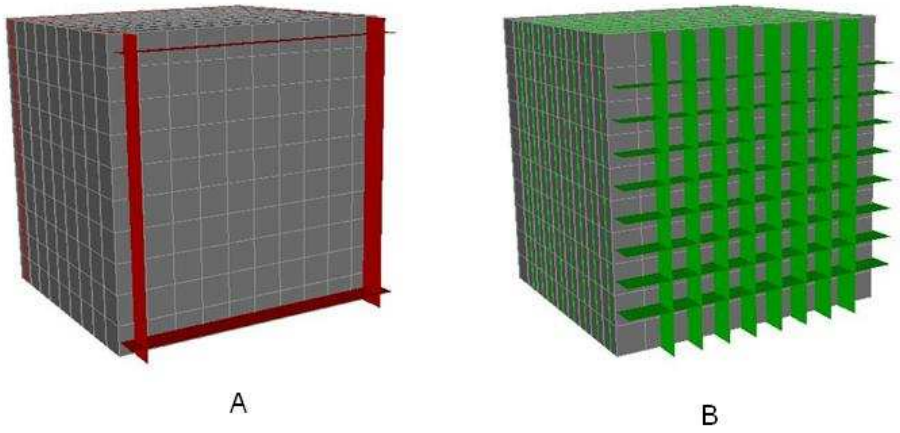

Figure 4.18. The image on the left shows the fundamental sheets (in red) for a mesh of a cube. The image on the right shows the secondary sheets (in green) for the same mesh.

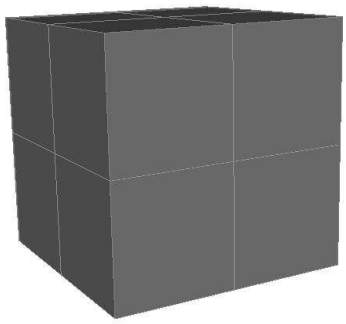

Figure 4.19. Mesh of the cube in Figure 4.18 after all secondary sheets have been removed.

assertion:

Assertion 3: The ***minimal hexahedral mesh*** for a geometric object is defined by one of the possible sets of fundamental sheets for a geometric object.

Rationale: By assertion 1, any mesh topology which captures the geometric topology of the mesh can also be converted to a fundamental mesh for that geometry. Additionally, by assertion 2, any fundamental mesh can be converted to any other fundamental mesh for that geometric topology. Because the fundamental sheets are the only sheets necessary to maintain the geometric topology, any sheets that are not fundamental can be removed from the mesh. This results in a mesh which only contains fundamental sheets. Therefore,

the minimal mesh is also a fundamental mesh for the geometry.

In the case of the cube, merging three sets of sheets to produce a final set of three fundamental sheets produces the minimal mesh for the cube (i.e., a single hexahedron). This process is also shown in Figure 4.20 for the meshes in Figure 4.16. We start with the mesh with the fewest elements (Mesh B in Figure 4.16) and note that the bottom sheet (shown in Figure 4.20) can be removed to produce a nonfundamental mesh (shown as the middle image of Figure 4.20). If we convert the nonfundamental mesh to a fundamental mesh using a boundary face collapse operation, the resulting mesh is both fundamental and minimal.

4.3.3 Examples

In this section, we will demonstrate a couple of examples of removing hexahedral sheets while preserving the geometric fidelity of the mesh to the original geometry by maintaining the fundamental sheets of the mesh.

4.3.3.1 Ellipse in cube. The image on the left of Figure 4.21 shows a mesh of two volumes (a cube with an embedded ellipsoid). The image on the right of the same figure shows the fundamental sheets for the ellipsoid in yellow and the fundamental sheets for the cube are shown in red. The remaining sheets are secondary sheets that are subsequently removed. Figure 4.22 shows the mesh after approximately half of the secondary sheets are removed and Figure 4.23 shows the mesh after all secondary sheets have been removed. There are 28 hexahedra shown in Figure 4.23.

If we perform a boundary face collapse operation to which joins the four sheets around the outer boundary into a single sheet, we can obtain the mesh shown in Figure 4.24. This mesh has 20 elements and is more minimal than the mesh shown in Figure 4.23.

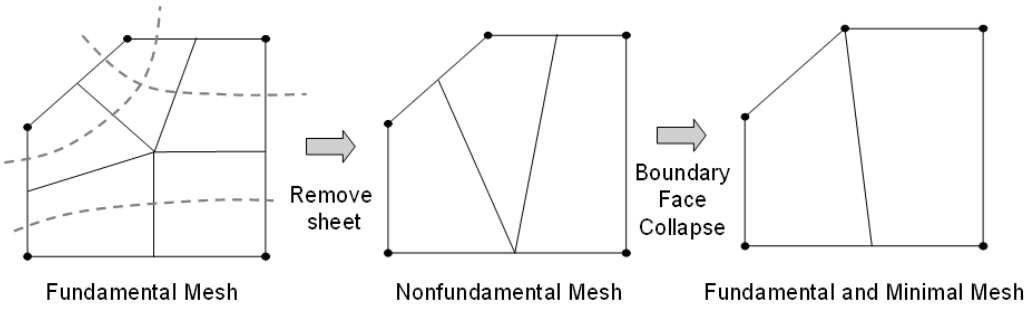

Figure 4.20. Conversion of a fundamental mesh (from Figure 4.16) to an alternative fundamental and minimal mesh.

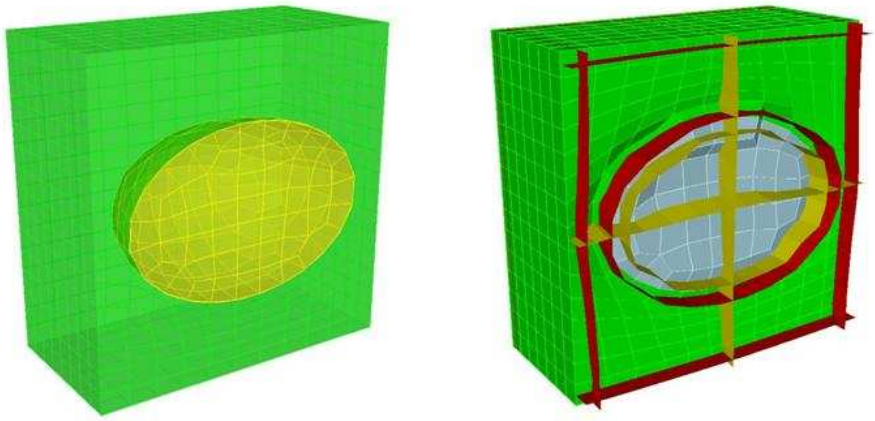

Figure 4.21. Hexahedral mesh of an ellipsoid embedded in a cube. This mesh contains 2,022 elements. The fundamental sheets of this mesh are shown on the right.

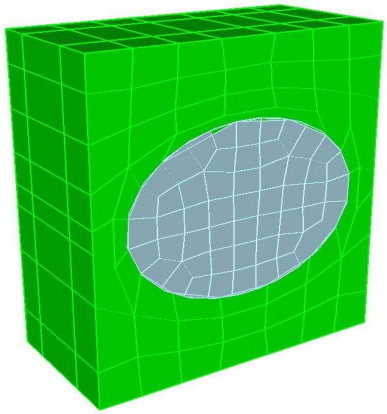

Figure 4.22. Coarsened mesh of ellipsoid embedded in a cube after removing approximately half of the secondary sheets

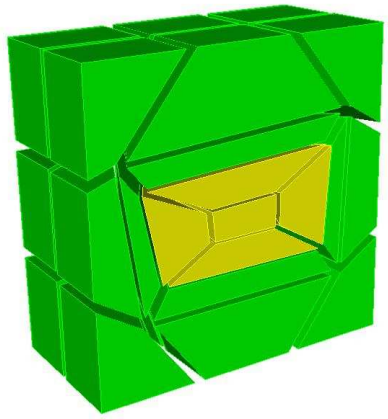

Figure 4.23. Coarsened mesh of ellipsoid embedded in a cube after removing the secondary sheets. This mesh has 28 elements.

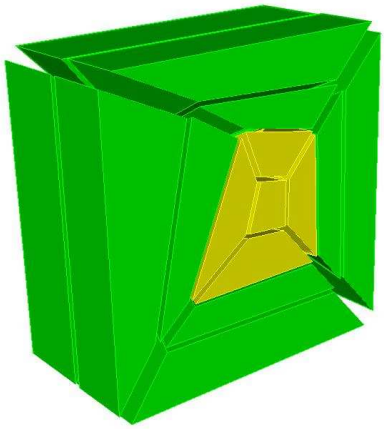

Figure 4.24. Hexahedral mesh of ellipsoid embedded in a cube after boundary face collapse operations were performed on the four corners to further reduce the number of elements. This mesh has 20 elements.

4.3.3.2 Artificial knee joint. Figure 4.25 shows a hexahedral mesh of an artificial knee joint meshed in CUBIT [23]. The fundamental sheets associated with the geometric surfaces is shown in the image on the right (the fundamental sheets associated with curves in the geometric model are not shown in this image).

Figure 4.26 shows the hexahedral mesh after the secondary sheets have been removed from the mesh. This mesh contains 77 elements. The image on the right shows the remaining fundamental sheets associated with this model. Removal of any additional sheets would result in a mesh topology that would not conform to the geometric topology. However, it may be possible to further minimize the number of elements in the mesh utilizing additional face collapse and subsequent sheet extraction operations.

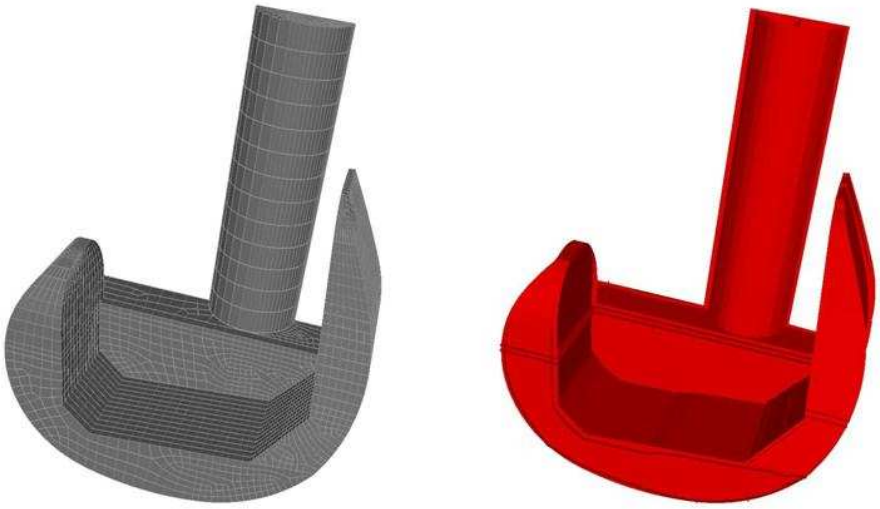

Figure 4.25. Hexahedral mesh of an artificial knee joint. The mesh contains 10,534 elements. The image on the right displays the fundamental sheets associated with surfaces only (the fundamental sheets associated with the curves in the model are not shown.)

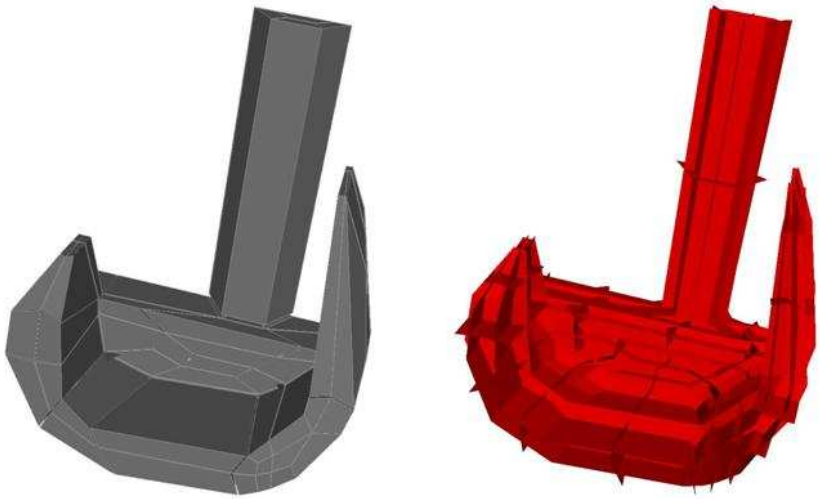

Figure 4.26. Hexahedral mesh of an artificial knee joint after secondary sheets have been removed. This mesh contains 77 elements. The image on the right displays all of the sheets in the coarsened model.

CHAPTER 5

GENERATING HEXAHEDRAL MESHES FROM ISOSURFACES

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, and there has been substantial effort placed in generating computer models that can be visualized and manipulated. Traditionally, the Marching Cubes algorithm [55] 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 are not of sufficient quality for use with numerical techniques including finite element, finite volume, or finite differences methods.

In this chapter, we demonstrate the creation of hexahedral meshes given an isosurface via modification to an existing algorithm (LBIE Mesher) and the creation of a new algorithm in SCIRun [92]. Since the boundary of a hexahedral mesh is a quadrilateral mesh, this method can be utilized to created quadrilateral isosurfaces in a manner similar to the marching cubes method [55] for creating triangular isosurfaces. We will present two similar solutions to this problem. The first method is an enhancement to the LBIE mesher created by Zhang, et al. [124, 122] using volumetric image data as input and producing the hexahedral mesh as output. The second method was implemented as a new module in SCIRun [92] and 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 remainder of this chapter will be as outlined. First, we will provide a background for generating isosurfaces, along with some additional theory for capturing these isosurfaces in hexahedral meshes. Then we will provide some results showing the improvements to the quality of the hexahedral meshes generated by the LBIE mesh generation tool. Finally, we will describe the method implemented in SCIRun and give some additional results utilizing this tool.

5.1 Introduction to Hexahedral Isosurfacing

A hexahedral isosurfacing problem can be described as follows: Given a 3D volume V, and an isosurface S defined inside this volume, compute a hexahedral mesh H that has S as its external boundary. Furthermore, we require H to be composed completely of high quality elements suitable for use in computational simulations, in particular, H should not contain any inverted or void elements.

One of the most general and powerful techniques for volumetric boundary reconstructions is isosurface extraction. An *isosurface* $\mathcal{S}_a$ is defined as the *preimage* of a function $f : \Re^n \rightarrow \Re$ and value a; it is the set of values in the domain that map to a, *i.e.,*$\mathcal{S}_a = \{x \in \Re^n : f(x) = a\}$. We then say f is the *implicit function* and a is the *isovalue*. Most often, f is given as a sampled 3D volume.

One of the most widely used techniques for isosurface generation is the Marching Cubes (MC) algorithm [55, 72]. Marching Cubes samples the function f at a grid of fixed resolution, and then uses a table of possible configurations of range signs to create a triangulated surface out of those samples. The main strengths of MC are its generality, simplicity and robustness. Following the discovery of the Marching Cubes algorithm, the last two decades have seen a constant stream of work on effective techniques for the computation and visualization of isosurfaces (e.g., see [72, 54, 4, 118, 20]).

Surfaces generated by MC have inherent bias caused by placing vertices on all intersections between grid edges and the surface. Because of this, meshes generated by MC (and variants) are typically over-tessellated, and contain many bad elements. Dual contouring methods [42, 35] reduce many of these problems. They generate surfaces that are the topological dual of the MC surfaces, and can also be used to reproduce sharp features by extrapolating the sampled normals.

5.1.1 Background on Octree Methods

In the 1980s, an octree technique was developed to generate tetrahedral meshes [121, 94]. Cubes containing the geometric model are recursively subdivided until the desired resolution is reached. Tetrahedral meshes are constructed from the irregular cells on the boundary and the internal regular cells. Quality is improved by restricting the octree construction; in particular, the octree subdivision is constrained to make the subdivision level of neighboring cells to only differ by one.

A grid-based approach was also proposed to generate a fitted 3D grid of hexahedral elements on the interior of the volume [87]. The regular grid of hexes is generated for

the interior volume and hexahedral elements are added at the boundaries to fill gaps. The resulting mesh does not preserve a predefined input surface mesh, and will change as the orientation of the cubes in the octree structure is changed. This method is robust, but it tends to generate poor quality elements along the boundary. Therefore, quality improvement is required. Modifications were introduced to allow for significant transitions in element sizes utilizing an octree decomposition of the domain [110, 89].

5.1.2 LBIE-Mesher

Isocontouring methods, primal contouring (or Marching Cubes) and dual contouring, have been extended to generate finite element meshes from volumetric data. The Marching Cubes algorithm (MC) [55] was extended to extract tetrahedral meshes between two isosurfaces directly from volume data [32]. A different algorithm, Marching Tetrahedra (MT), was proposed for interval volume tetrahedralization [73].

Dual Contouring [42] was also extended to finite element mesh generation. A software package, namely LBIE-Mesher (Level Set Boundary Interior and Exterior Mesher), has been developed to construct adaptive and quality 2D (triangular/quadrilateral) and 3D (tetrahedral/hexahedral) finite element meshes [124, 122].

We base our work on LBIE-Mesher, which is used to generate the initial hexahedral meshes from the volumetric data [124, 122]. The meshing algorithm in LBIE-Mesher works as follows. First, a bottom-up surface topology preserving octree-based algorithm is applied to select a starting octree level. Then the dual contouring method is used to extract a preliminary uniform hexahedral mesh by analyzing each interior grid point. The uniform mesh is decomposed into finer hexes adaptively using predefined templates without introducing any hanging nodes. The positions of all boundary vertices are recalculated to approximate the boundary surface more accurately.

To improve mesh quality, LBIE-Mesher performs a final step, where geometric flows are used to smooth the boundary surface, and improve the quality of the extracted hexahedral meshes [94]. There are three main steps in the LBIE-Mesher quality improvement scheme: (1) Denoising the surface mesh, which consists of vertex adjustment in the normal direction with volume preservation. Here the surface diffusion flow is chosen to smooth the surface, and the discretized Laplacian-Beltrami operator is used to solve the geometric partial differential equation (GPDE). (2) Improving the aspect ratio of the surface mesh, which consists of vertex adjustment in the tangent direction with feature preservation. Surface features are preserved since the movement in the tangent plane does not change

the surface shape. (3) Improving the aspect ratio of the hexahedral mesh, which consists of adjustment inside the volume.

5.2 Enhancements to the LBIE Algorithm

The resulting mesh produced by the LBIE mesher provides a very nice quadrilateral isosurface; however, many of the resulting hexahedral elements, especially near the isosurface boundary, have unusable quality with many of the scaled Jacobians measuring well below zero. As described in Chapter 4, we conjecture that the reason behind the low element quality is a result of the nonfundamental nature of the mesh at the isosurface boundary. We hypothesize that insertion of a single fundamental sheet at the boundary of the hexahedral mesh that describes the original isosurface will result in dramatic improvements in the overall quality of the resulting hexahedral mesh.

Insertion of this new fundamental sheet has the effect of aligning the boundary hexes with the isosurface, and of providing extra flexibility for the optimization of the hexahedra resulting in meshes with sufficient quality for computer-based simulations.

5.2.1 Adding a Boundary Sheet

In order to add a new sheet near the boundary of the geometry, we use a modified version of a pillowing algorithm [67] as described in Chapter 3.

When inserting a single sheet next to the boundary, we define our shrink set as all of the elements within the solid. We desire the original surface mesh to be undisturbed by any of our modification operations, so a copy of the surface mesh is made prior to shrinking the layer of elements near the boundary. Reconnecting the boundary of the shrink-set with the correct nodes in the previously copied surface mesh completes the inserted sheet. For each face on the boundary, one new hexahedral element is created as the boundary layer defining the newly inserted sheet in the improved mesh.

Figure 5.1 demonstrates a mesh with a newly inserted sheet for both the interior and exterior boundary. The mesh on the left is the mesh resulting from the LBIE Mesher where the octree mesh is refined to a level to capture major details of the isosurface geometry of the embedded head and a dual contouring technique is used to project the nodes to the isosurface [124, 122]. The resulting mesh then has the nodes on the boundaries moved to better capture the geometric features of the data using a geometric flow operator to determine optimal placement of the nodes for the resulting quadrilateral isosurface [125]. A new sheet is inserted for each surface on the boundary of the mesh

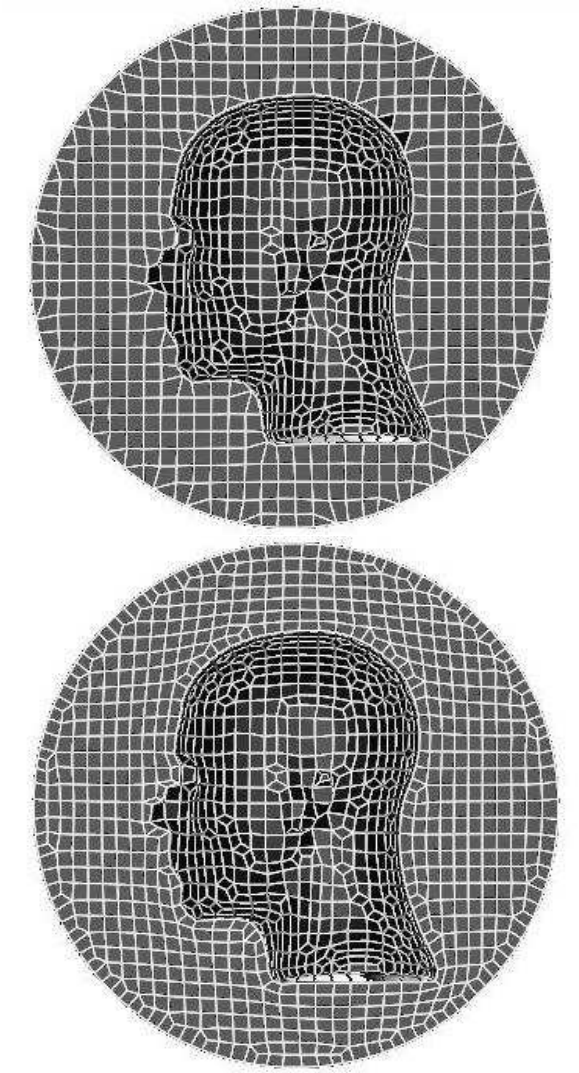

Figure 5.1. A hexahedral mesh of a sphere with a human head embedded in the center.

(one for the spherical boundary and a second for the boundary of the head). The mesh topology in the upper image is identical to the mesh topology in the lower image with the exception of the two additional sheets of hexahedral elements for the interior and exterior boundary surfaces. The resulting mesh topology has a higher quality potential and flexibility due to the improved mesh topology capturing the geometric boundary. (Note the quadrilaterals in the left figure, which look more triangular in shape, and how the addition of the inserted sheet allows some additional flexibility to improve the shape of these elements.)

5.2.2 Mesh Optimization and Verification

Because correct and optimal placement of the new sheet can be difficult, it is often desirable to perform a smoothing, or mesh optimization, operation on the resulting mesh after the new sheet has been inserted to obtain better nodal placement and higher quality elements. We have utilized the TSTT Mesh Quality and Improvement Toolkit (MESQUITE) library of smoothing algorithms to accomplish the optimization of the meshes [62, 15]. Within the MESQUITE class of mesh smoothers we have access to both *untangling* smoothers (a *tangled* mesh contains elements that are inverted, or have a negative Jacobian value) and a wide range of optimization algorithms for untangled meshes. We have utilized an *inverse mean ratio* smoother (as described by Knupp [48]), that incorporates an L2-norm template with guarantees that (1) the mesh will remain untangled if the initial mesh is untangled, and (2) the average value of the inverse mean ratio will either stay the same, or be decreased.

Utilizing the guarantees of the L2-norm template, we can assume that if the nodes on the boundary of the mesh are fixed in place while the interior nodes are free to move during optimization, then the smoothed mesh will have either the same or better quality upon completion of the optimization. We, therefore, can run the smoother until the optimization converges with the given geometry and mesh topology. All results reported in the next section have all been optimized using this inverse mean-ratio smoother (including the original LBIE meshes). While it may be possible to subsequently improve the quality of some of the individual elements, it would be done at the expense of the quality of the adjacent elements. Therefore, we have some confidence that the average element quality for the given mesh topology and geometry is optimal, and only modifications to the mesh topology (i.e., sheet insertions or extractions, refinements, etc.) can be utilized to gain additional average quality improvements in the reported meshes.

5.3 Results with LBIE Mesher

In this section we display results detailing the difference in element quality after improving the method of capturing the original boundary. We will utilize the scaled Jacobian metric, described earlier, as the definitive measure for hexahedral quality.

In order to make appropriate comparisons, the boundary (surface) mesh is assumed to be fixed. Thus only nodes on the interior of the mesh are free to move in an effort to optimize the quality of the reported mesh. The boundary meshes were originally

optimized using the *geometric flow* smoothing algorithm as reported by Zhang, et al. [125]. No additional optimization of the boundary mesh has been incorporated, although in many cases improved quality of the interior elements would result with improved boundary optimization. The results shown in the figures in this section are the difference in the quality of the meshes before insertion of the hexahedral sheets and after insertion of the hexahedral sheets.

Additionally, all reported hexahedral meshes have been smoothed with the MESQUITE smoother in an attempt to report maximal quality results for the given geometry and mesh topology. The distributions of element quality are reported using the *scaled Jacobian* metric as implemented in the VERDICT library [108].

There are four examples given: a human knee model, a human head, a meshed sphere around the human head, and a model of an mAChe biomolecule. The distribution of element quality is given for each of the meshes both before sheet insertion and after inserting a sheet to more appropriately capture the geometric features of the boundary surfaces. As will be shown in each of the quality distribution plots, after sheet insertion and mesh smoothing the meshes have generally higher quality than the original meshes without sheet insertion. Additionally, the introduction of the boundary sheet enables the removal of all negative or questionable scaled Jacobian values from the displayed meshes.

5.3.1 Knee Model

The hexahedral mesh of the knee model, shown in the middle image of Figure 5.2 was generated by the LBIE mesher and contains 1,338 total hexes whose surface boundary was optimized utilizing 'geometric flow' optimization [125]. A cut-away view of the original mesh is shown in the center. An additional sheet of hexahedral elements was added to the boundary resulting in a new mesh consisting of 2,682 total elements. The new sheet was added to improve the potential quality of the mesh, and the original surface mesh was not changed by the sheet insertion process (shown on the far right).

To ensure an appropriate and fair comparison, both the original mesh and the improved mesh were optimized using an inverse mean-ratio smoother, as discussed previously. After optimization, the original mesh contains 367 elements of questionable or unacceptable quality, while the mesh with the sheet inserted has all elements with scaled Jacobians greater than 0.2 (see Figure 5.3).

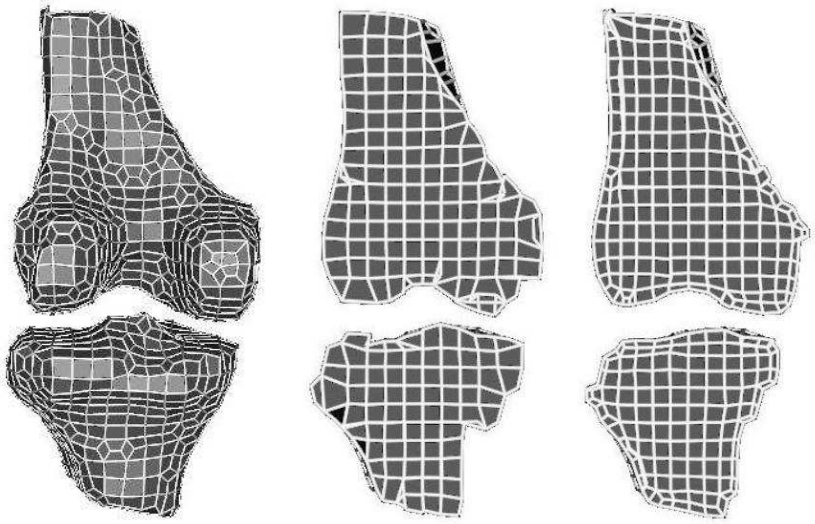

Figure 5.2. Hexahedral mesh of the knee model. The image in the middle was generated using the LBIE mesher, and a boundary sheet was added to improve the mesh quality (shown on the right.)

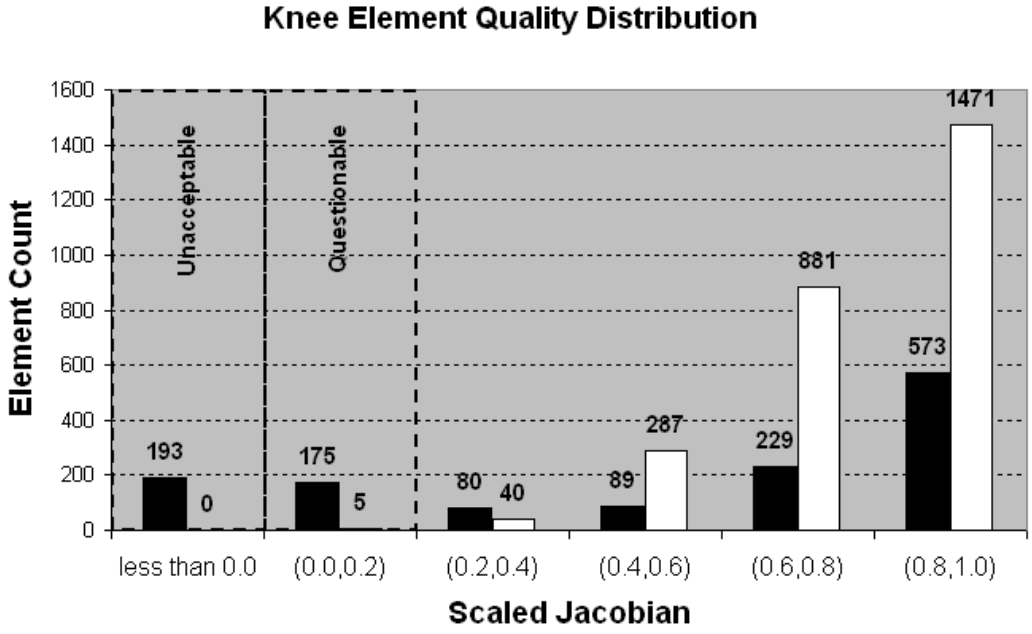

Figure 5.3. Distribution of element quality based on the scaled Jacobian measure of each element for the knee model. Element quality distribution before sheet insertion is shown in black, and after sheet insertion is shown in white. The mesh before sheet insertion contains 1,339 hexahedra, and after insertion of the sheet the knee contains 2,684 hexahedra.

To ensure that the quality improvement displayed in these meshes is not due to the increased number of elements produced by the sheet insertion process, we have conducted two additional experiments to verify that sheet insertion at the boundary results in the quality improvements noted. The first experiment is to increase the number of elements in the original mesh by 'dicing' random sheets in the original mesh (the dicing algorithm is described more fully in Chapter 3). The mesh resulting from this procedure is shown in Figure 5.4. This mesh contains 3,005 hexahedral elements with the distribution of element quality shown in Figure 5.5. There is no substantial change in the quality distribution of the hexahedra in this mesh resulting from random addition of new sheets, and we can draw the conclusion that the improved quality distribution demonstrated after sheet insertion is not due to the increased number of elements in the mesh.

The second experiment conducted was to randomly remove sheets (with the possible exception of the one fundamental sheet initially added) from the mesh with the sheet inserted at the boundary until the element count of this mesh was approximately equivalent with the original mesh. The sheets were removed using a sheet extraction algorithm as described in Chapter 3. The mesh resulting from randomly removing sheets is shown in Figure 5.6. This mesh contains 1,362 elements with the distribution of element quality shown in Figure 5.7. This mesh continues to show substantial improvement in quality

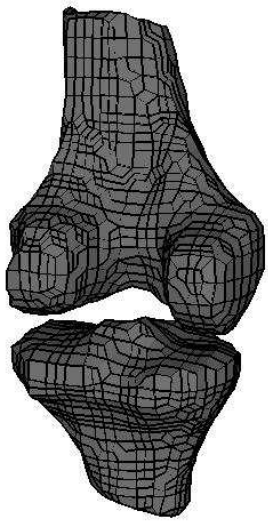

Figure 5.4. Hexahedral mesh of the knee model after randomly adding new hexahedra using a dicing algorithm.

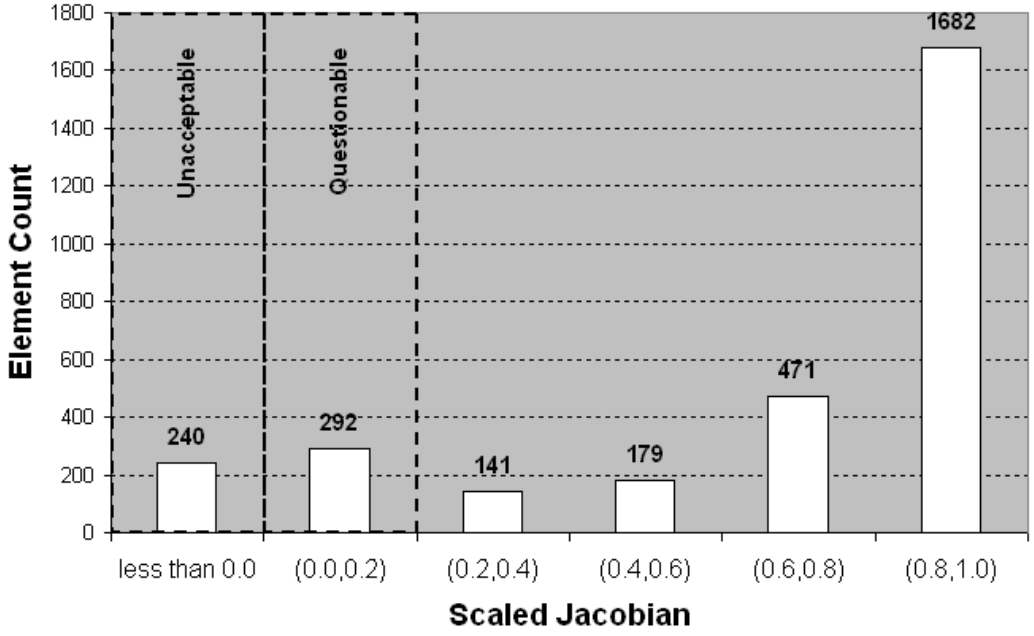

Figure 5.5. Distribution of element quality based on the scaled Jacobian measure of each element for the original knee model after randomly adding sheets using a dicing algorithm. The mesh contains 3,005 hexahedra.

despite the removal of a significant number of elements.

From the results of these two experiments, we conclude that the insertion of the fundamental sheet near the isosurface boundary is responsible for the quality improvement noted in the hexahedral mesh of the knee. The addition of a hexahedral sheet near the boundary adds flexibility to the hexahedra near the boundary which is recognized in the improvements in mesh quality after smoothing the mesh. By using a fundamental sheet, as opposed to many nonfundamental sheets, the hexahedral elements in areas of boundary curvature are now properly aligned with the boundary surface and the adjustment to hexahedral shape can be performed a layer into the mesh where additional nodal movement is possible.

This new layer can also be duplicated with a dicing algorithm [61, 60] to align multiple layers of hexahedra for improved quality near the boundary. Using a dicing algorithm to duplicate the new fundamental sheet will result in multiple layers of hexahedra with regular topology near the boundary of the geometry which may be useful in specific types of analysis.

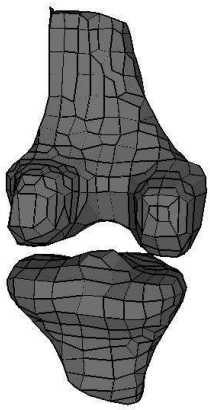

Figure 5.6. Hexahedral mesh of the knee model after a fundamental sheet has been inserted at the boundary and then randomly removing sheets using a sheet extraction algorithm.

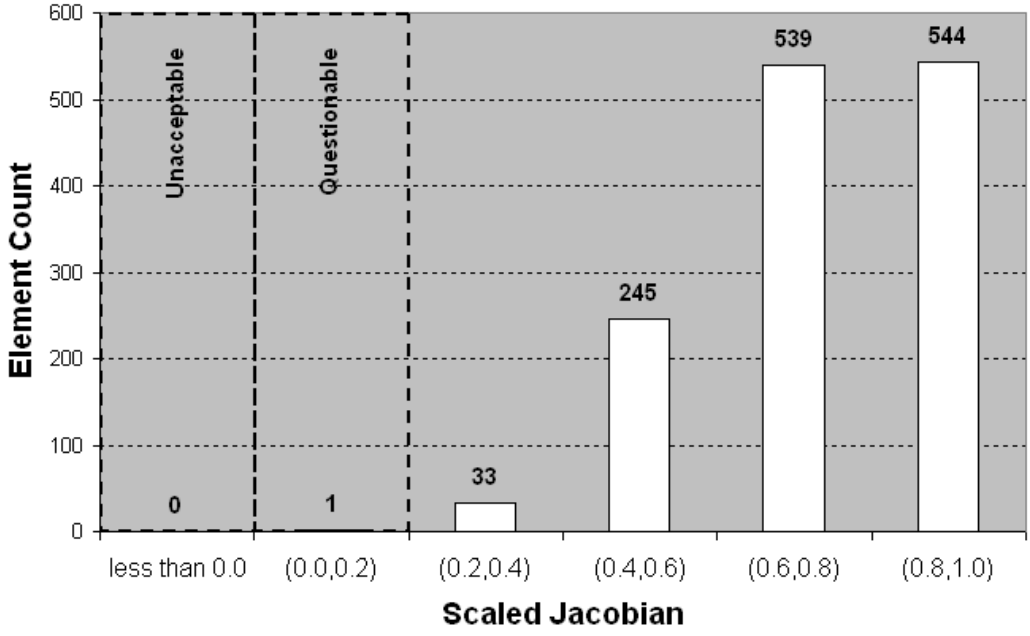

Figure 5.7. Distribution of element quality based on the scaled Jacobian measure of each element for the original knee model after a fundamental sheet was inserted at the boundary and then randomly removing sheets using a sheet extraction algorithm. The mesh contains 1,362 hexahedra.

5.3.2 Head Model

The hexahedral mesh of the human head contains 6,583 total hexahedra whose surface boundary was optimized utilizing 'geometric flow' optimization as generated by the LBIE mesher (a cut-away view of the original mesh is shown in the middle image of Figure 5.8. An additional sheet of hexahedral elements was added to the boundary to improve quality resulting in a new mesh consisting of 9,487 total elements, leaving the original surface mesh unchanged (shown on the far right).

Both the original mesh and the improved mesh were subsequently optimized with using an inverse mean-ratio smoother. The boundary of the original mesh was optimized with 'geometric flow' smoothing [125] and contains 1,025 elements of questionable or unacceptable quality, while the mesh with improved boundary capture has only 1 element of questionable quality (due to poor quality of one of the boundary quadrilaterals). The distribution of element quality based on a scaled Jacobian metric is shown in Figure 5.9.

Because the meshes generated by the LBIE mesher do not add additional elements to the original octree, but rather use a dual contouring algorithm to control placement and to locate a quadrilateral boundary mesh interior to the octree, the elements on the exterior of the isosurface remain a valid hexahedral mesh as well as the elements on the interior of the isosurface. We can take advantage of this mesh validity, and improve the mesh on the exterior of the head in addition to the interior, and place additional boundary sheets to improve the quality of the two pieces. The result will be a conformal mesh with improved quality for both the interior and exterior elements.

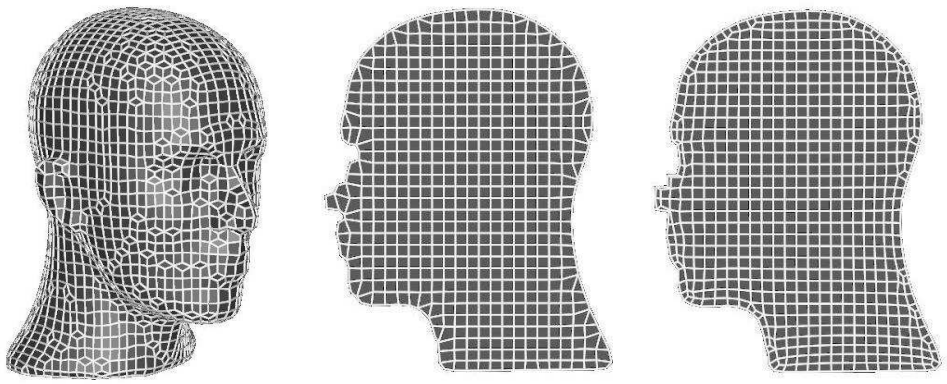

Figure 5.8. Hexahedral mesh of a human head. The mesh in the middle image was generated using the LBIE mesher. A boundary sheet was added to improve the mesh quality to generate the mesh in the image on the far right.

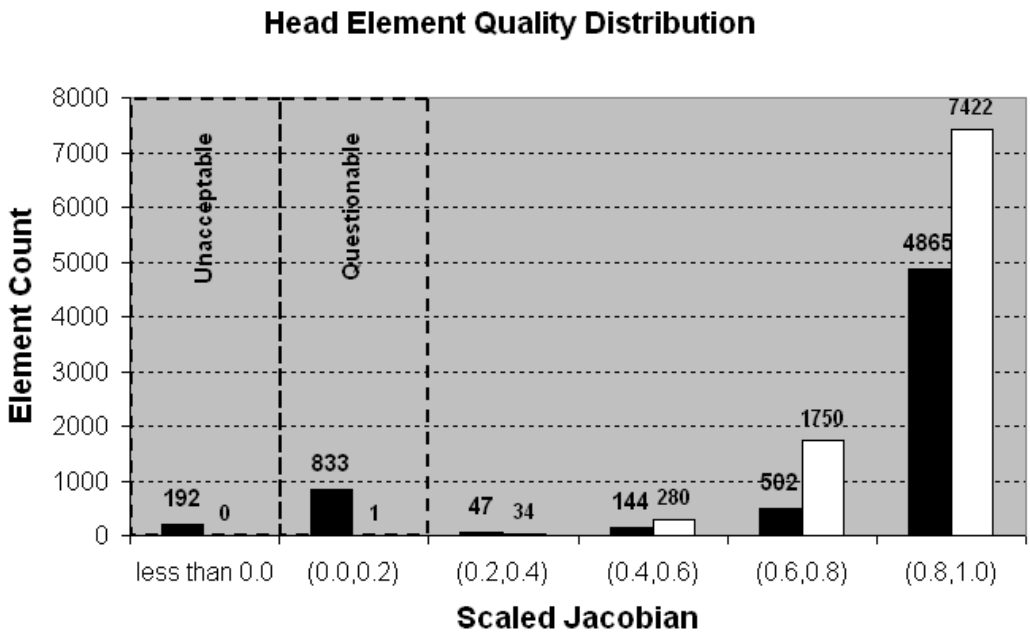

Figure 5.9. Distribution of element quality based on the scaled Jacobian measure of each element for the human head model. Element quality distribution before sheet insertion is shown in black, and after sheet insertion is shown in white. Before sheet insertion there are 6,583 hexahedra in the mesh of the head, while after sheet insertion there are 9,487 total hexahedra.

Figure 5.10 shows the human head model embedded in a spherical boundary mesh. The original exterior hexahedral mesh contains 13,352 total hexes whose surface boundary was optimized utilizing 'geometric flow' optimization. Two additional sheets of hexahedral elements were added to the boundary to improve potential quality resulting in a new mesh consisting of 19,412 total elements, leaving the original surface mesh unchanged (refer to Figure 5.1 for cut-away views of the mesh).

The quadrilateral boundary mesh was optimized with 'geometric flow' smoothing [125]. After sheet insertion, both the original mesh and the mesh with the newly inserted sheets were optimized with an inverse mean-ratio smoother. The original mesh contains 2,485 elements of questionable or unacceptable quality, while the mesh with improved boundary capture has all elements with scaled Jacobians greater than 0.2 (see Figure 5.11).

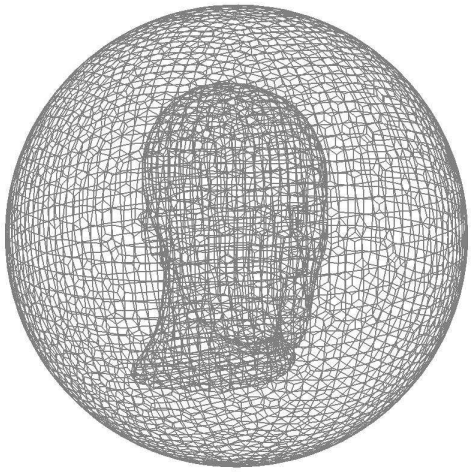

Figure 5.10. Human head model embedded in a spherical boundary mesh. The hexahedral mesh in the middle image was generated by the LBIE mesher. Two boundary sheets are added (one near the spherical boundary and a second near the isosurface o the head) to improve the overall quality.

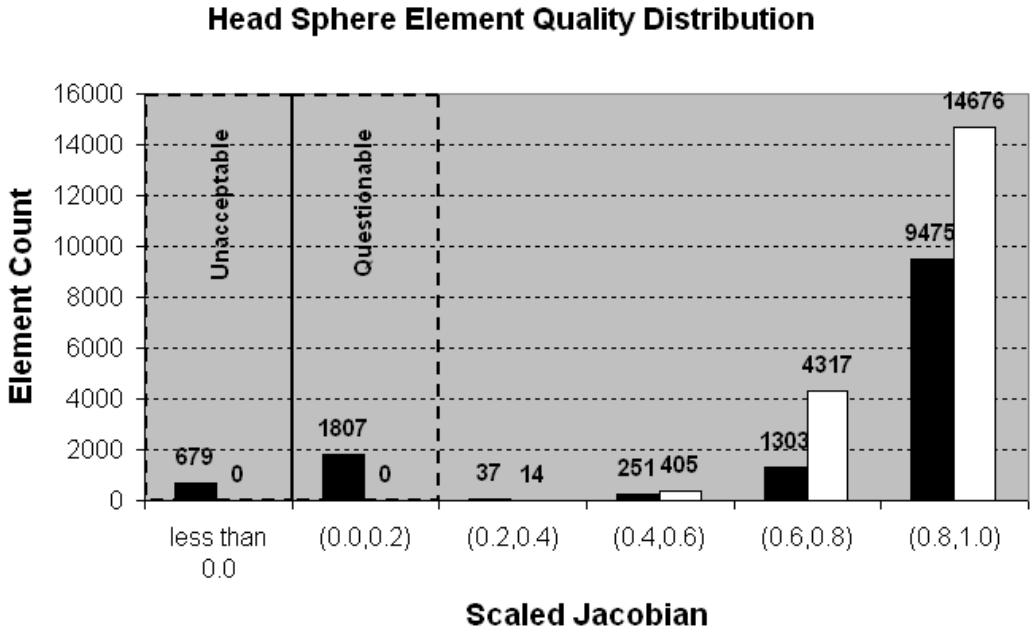

Figure 5.11. Distribution of element quality based on the scaled Jacobian measure of each element for the model of the head embedded in a sphere. Element quality distribution before sheet insertion is shown in black, and after sheet insertion is shown in white. Before sheet insertion, the mesh contains 13,552 hexahedra, while after sheet insertion the resulting mesh contains 19,412 hexahedra.

5.3.3 mAChE model

A hexahedral mesh of a mAChe biomolecule model is shown in Figure 5.12. The original mesh was generated by the LBIE mesher and contains 70,913 total hexes (a cut-away view of the original mesh is shown in the middle image). An additional sheet of hexahedral elements was added to the boundary to improve potential quality resulting in a new mesh consisting of 90,937 total elements, leaving the original surface mesh unchanged (shown on the far right).

Both meshes were smoothed using an inverse mean-ratio optimization routine (discussed earlier). After optimization, the original mesh has 7,298 questionable or unacceptable elements, while the mesh with improved boundary capture has 52 elements of questionable quality (due to poor quality of the boundary quadrilaterals) (see Figure 5.13).

In all of the previous examples, the process of inserting a fundamental sheet at the boundary of hexahedral mesh resulted in significant improvements to the quality of the hexahedra within the mesh. For all examples shown, the insertion of the new sheet provided additional flexibility for mesh optimization which resulted in elimination of all nonconvex or inverted elements from the mesh. The original meshes without the inserted fundamental sheet do not exhibit this same flexibility, and mesh optimization failed to remove the nonconvex and inverted elements from the original meshes. Additionally, after the sheet insertion technique, the hexahedra contained within the mesh maintained high quality even after element removal to equalize the number of elements within the meshes.

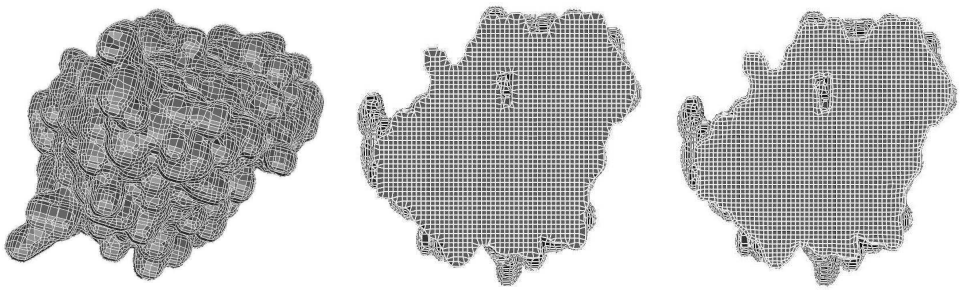

Figure 5.12. Biomolecule mAChE model.

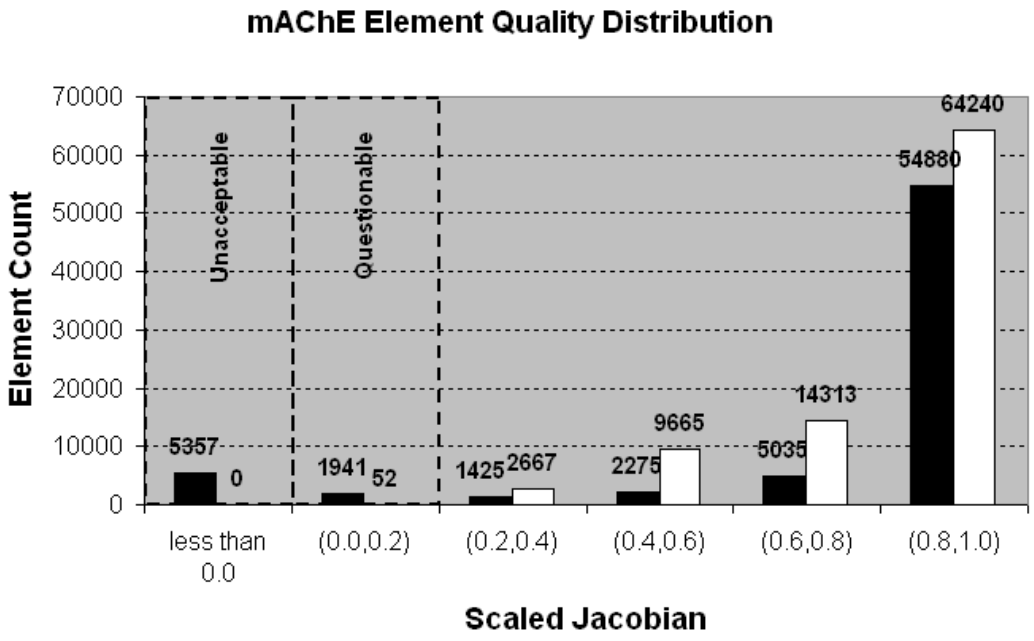

Figure 5.13. Distribution of element quality based on the scaled Jacobian measure of each element for the mAChe biomolecule model. Element quality distribution before sheet insertion is shown in black, and after sheet insertion is shown in white. The mAChE mesh contains 70,913 hexahedra before sheet insertion, and 90,937 hexahedra after sheet insertion.

5.4 Results with SCIRun

In this section we describe a new algorithm for converting triangle meshes on manifold surfaces into hexahedral meshes. 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 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 [92] 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 Chapter 6). 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 Figure 5.14 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). The algorithm for detecting intersecting triangles and separating the hexes into these three groups is explained in further detail in [91].

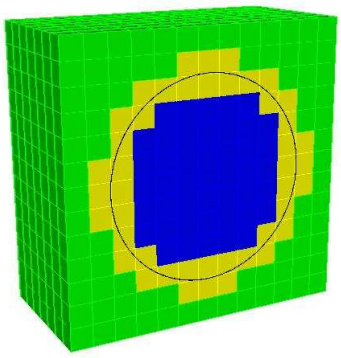

Figure 5.14. A hemispherically-shaped triangle mesh (the boundary of the triangle mesh is shown in black) is placed in a hexahedral grid. The hexahedra intersected by the triangle mesh are shown in yellow, while 'Side1' is drawn in green and 'Side2' is shown in blue.

3. *Collate the 'Intersected' hexahedra with either 'Side1' or 'Side2' and insert two hexahedral sheets between these two groups of hexahedra.* The 'Intersected' hexahedra are subsequently added to either 'Side1' or 'Side2', and two sheets of hexahedra, or pillows (see description of 'pillowing' in Chapter 3), around these two groups. For the example highlighted in Figure 5.14, depending on which side the intersected hexahedra are grouped, one of the meshes shown in Figure 5.15 will result.

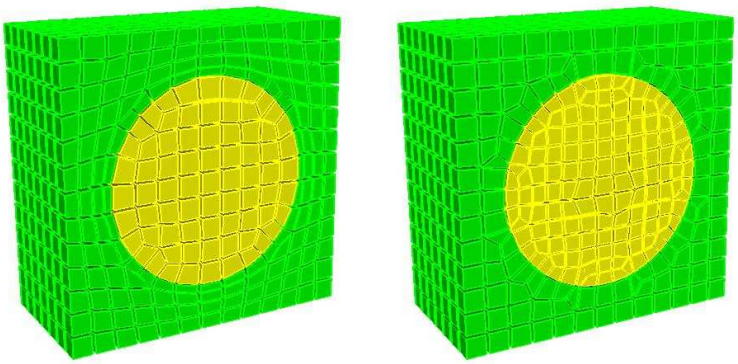

Figure 5.15. Slightly different meshes result depending on which side the intersected hexes are grouped. The image on the left shows the resulting mesh after sheet insertion if the intersected hexes are placed with Side1's hexes, while the image on the right has the intersected hexes being grouped with Side2.

The hexahedral sheets are inserted by (refer to Figure 5.16):

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 in pillowing 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 [62, 15] suite of smoothing algorithms available from a mesh smoothing module implemented in SCIRun [92]. These smoothers include Laplacian smoothing, a hybrid smoothing/optimization algorithm known as Smart Laplacian [30], and a mesh

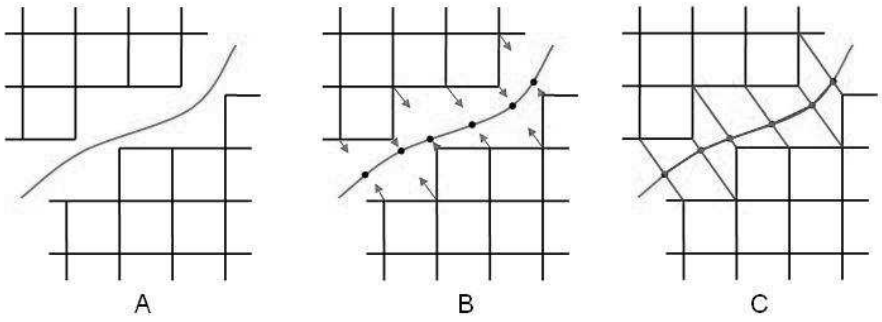

Figure 5.16. Image A shows the shrunken hexahedra with the triangle mesh shown in between the hexahedra. Image B shows a newly projected node to the triangle mesh for each node on the boundary of the shrunken mesh (note that a single node on the triangle mesh corresponds to one node on each of the shrunken 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.

optimization algorithm for improving the 'shape' metric, called Shape Improvement Optimization [48] (for a more in-depth description refer to Chapter 3). 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 [23]. CUBIT has the mesh smoothing and optimization algorithms listed above, along with some additional smoothing algorithms, including centroidal area smoothing, condition number optimization, and untangling. Additionally, CUBIT optionally allows smoothing to occur on a focused-set of elements that can dramatically reduce the amount of time needed for optimization of specific hexahedral elements.

5.4.1 Bumpy Sphere

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

The hexahedral mesh of the bumpy sphere model, shown in of Figure 5.17 was generated in SCIRun and optimized in CUBIT. The mesh contains 59,401 total hexes. The mesh was generated by first creating a regular grid of hexahedra that was 5% larger than a tight bounding box around the bumpy sphere geometry. Two hexahedral sheets were then placed in the grid using the original triangle mesh as a guide as described in the previous section. This mesh was then exported from SCIRun and translated into a file format readable by CUBIT.

In CUBIT, the quadrilateral mesh on the boundary was smoothed with a centroidal-area smoother. 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 Laplacian smoothing operation, an optimization algorithm to improve the condition number of each of the elements was performed to give the final results shown in Figure 5.18.

Figure 5.19 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)). The image of the combined facets gives an indication of how closely the hexahedral mesh conforms to the original geometry. Areas of solid red indicate a region with higher elevation where

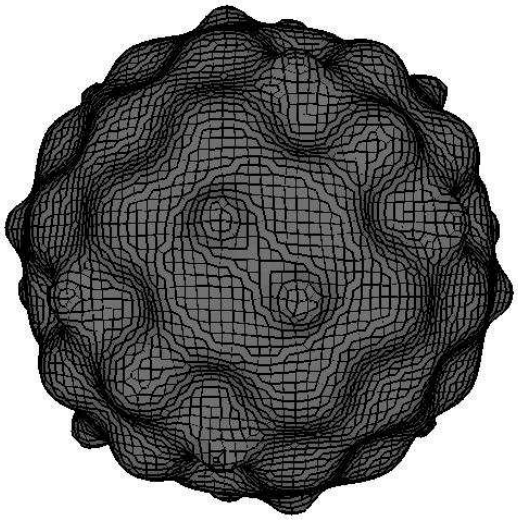

Figure 5.17. Hexahedral mesh of bumpy sphere model. The mesh contains 59,401 elements.

the hexahedral mesh does not attain the same height in elevation. Areas in solid green indicate areas of lower elevation in the original geometry where the hexahedral mesh has too much elevation. Areas where both red and green facets appear mottled with one another indicate satisfactory geometric fidelity to the original model. In the bumpy sphere model, the best way to improve the geometric fidelity would involve either refinement of the hexahedral mesh in areas of high curvature, or a generalized refinement to improve resolution throughout the model. Table 5.1 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 0.34% smaller than the volume enclosed by the original triangles with the bulk of the volume being lost under each of the bumps on the sphere.

Following the sheet insertion process, the hexahedral elements which were exterior to the bumpy sphere model were discarded. However, the discarded hexahedral elements also maintain conformity with the remaining elements and a contiguous mesh of both the interior and exterior of the bumpy sphere model could be maintained utilizing this method. Reduced element sizes will improved the geometric fidelity to the original model, and additional layers of elements could be inserted if a highly regular hexahedral topology is desired at the boundary of the bumpy sphere geometry.

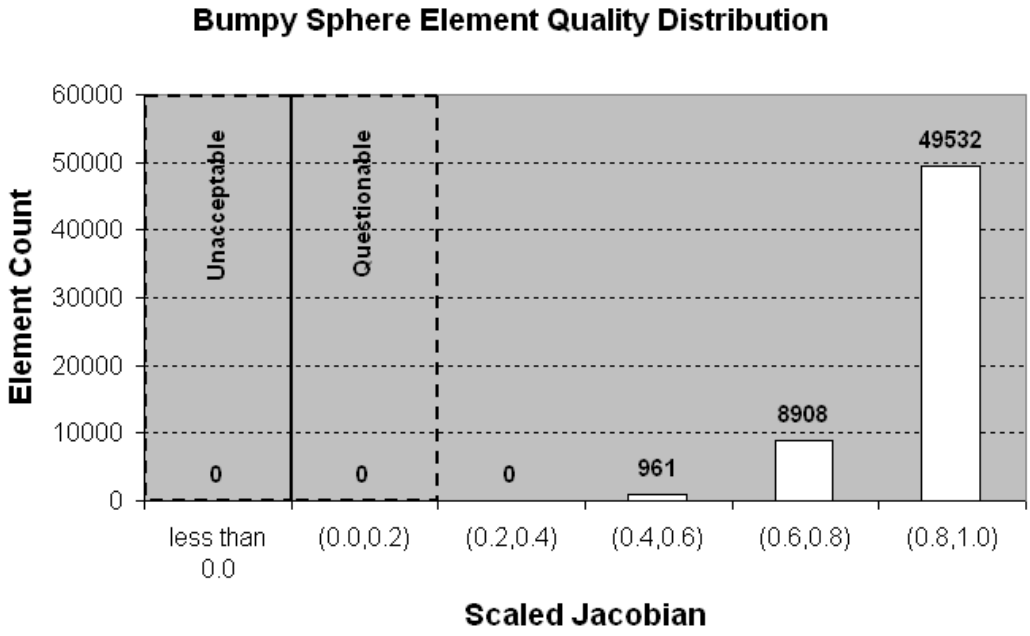

Figure 5.18. Distribution of element quality for bumpy sphere model.

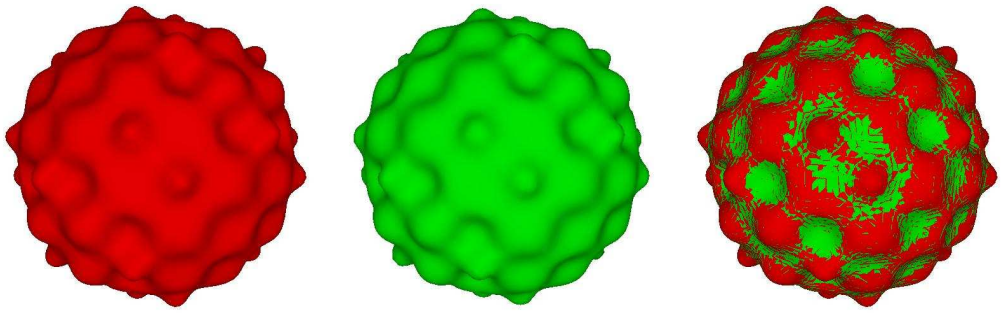

Figure 5.19. Geometry generated from original triangle facets shown in red (on the left), and the geometry generated from the hexahedral facets is shown in green (in the middle). An image where both sets of facets are overlapped is given on the right to give an indication of the overall geometric fidelity of the hexahedral mesh.

Table 5.1. Table indicating volume changes resulting from conversion of the original triangle mesh to a hexahedral mesh for the bumpy sphere model.

Volume (Triangle Mesh)	Volume(Hexahedral Mesh)	Difference	Percent Change
4186.41	4172.21	-14.20	-0.34%

5.4.2 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 Figure 5.20 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. This mesh was then exported from SCIRun and translated into a file format readable by CUBIT.

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 quadrilaterals, the boundary nodes were fixed and the hexahedral elements were smoothed

Figure 5.20. Front and back view of the hexahedral mesh of the hand. The mesh contains 202,974 elements.

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 Laplacian smoothing operation, an optimization algorithm to improve the condition number of each of the elements was performed to give the final results shown in Figure 5.21.

Figures 5.22 and 5.23 display 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 5.2 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 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.

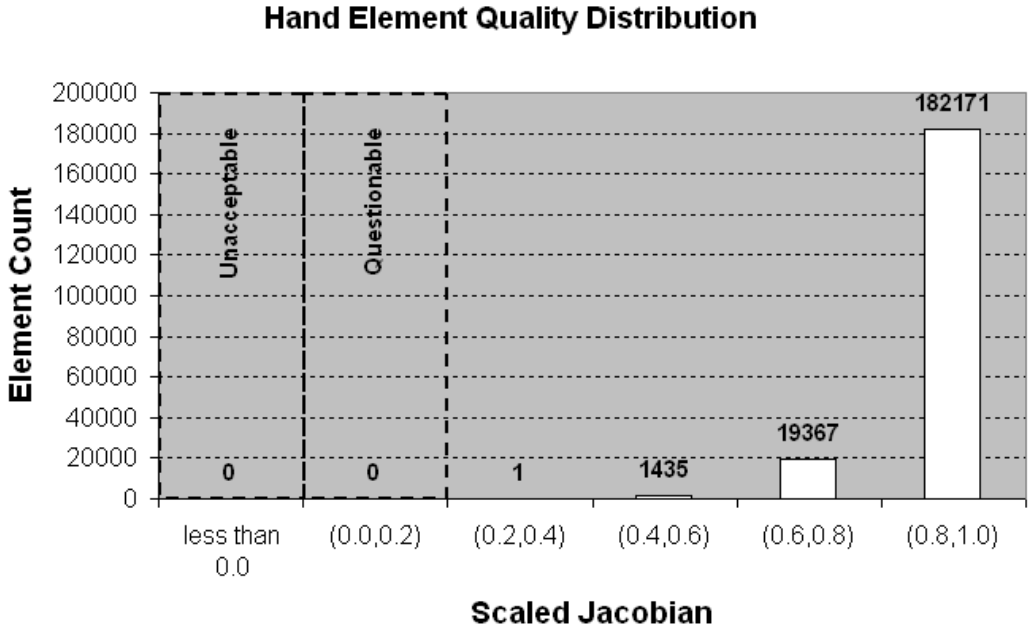

Figure 5.21. Distribution of element quality for the hand model.

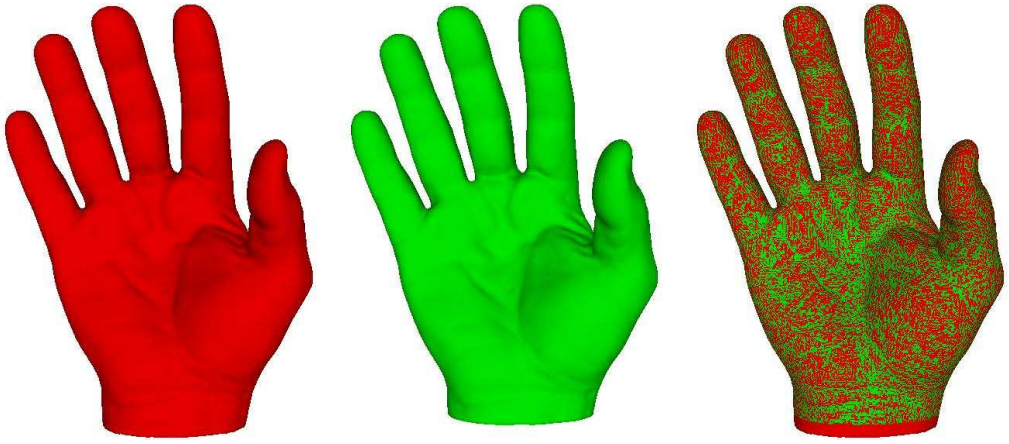

Figure 5.22. Geometry generated from original triangle facets shown in red (on the left), and the geometry generated from the hexahedral facets is shown in green (in the middle). An image where both sets of facets are overlapped is given on the right to give an indication of the overall geometric fidelity of the hexahedral mesh.

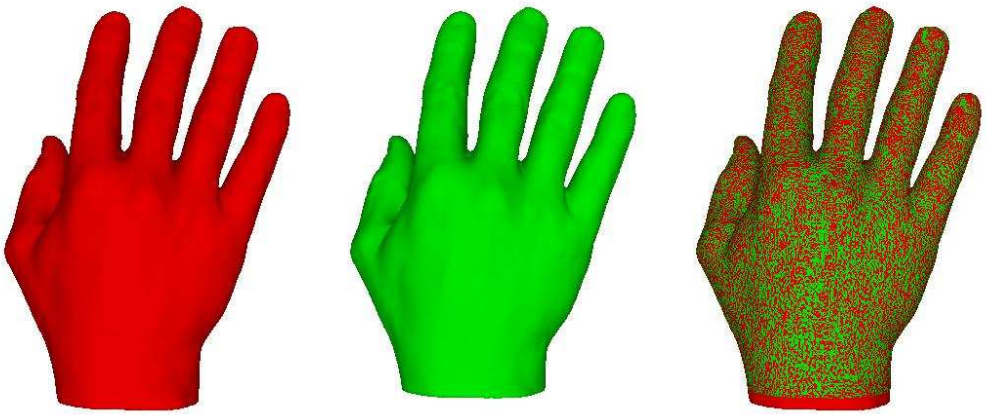

Figure 5.23. Geometry generated from original triangle facets shown in red (on the left), and the geometry generated from the hexahedral facets is shown in green (in the middle). An image where both sets of facets are overlapped is given on the right to give an indication of the overall geometric fidelity of the hexahedral mesh.

Table 5.2. 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%

5.4.3 Mouse Model

The 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 Figure 5.24 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 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, and the remaining hexahedral mesh was then exported from SCIRun and translated into a file format readable by CUBIT.

In CUBIT, 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 Figure 5.25), 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 nodes were fixed and the hexahedral elements were

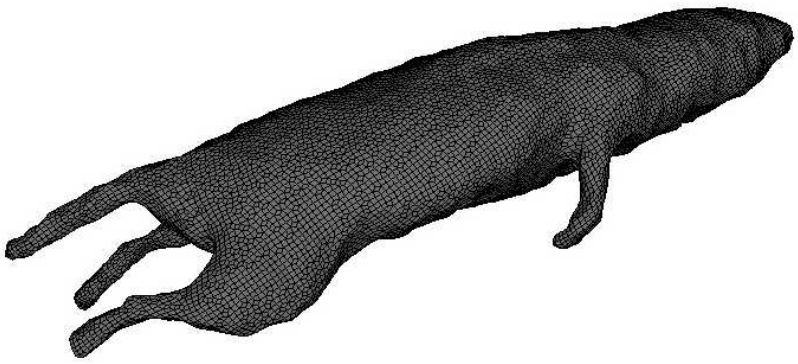

Figure 5.24. Hexahedral mesh of a mouse generated from CT data. The mesh contains 74,828 elements.

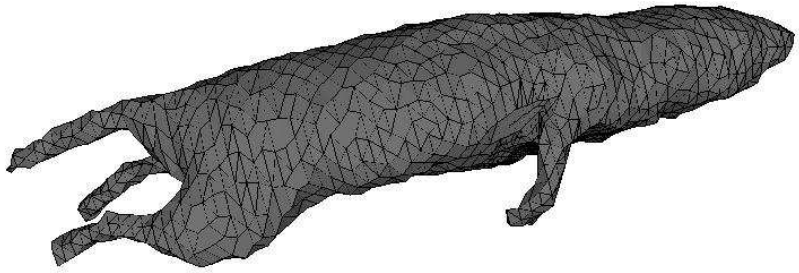

Figure 5.25. Original triangle mesh of the mouse model.

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 Laplacian smoothing operation, an optimization algorithm to improve the condition number of each of the elements was performed to give the final results shown in Figure 5.26.

Figure 5.27 displays the geometry for both the original geometry and the hexahedral mesh (the facets of the original triangle mesh are shown in red (upper left) and the facets

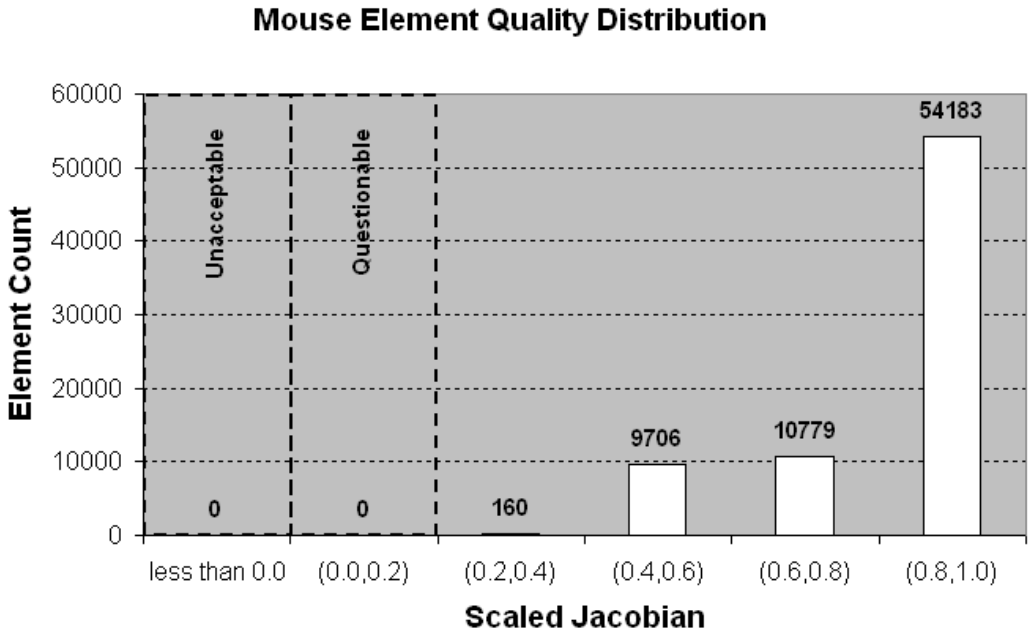

Figure 5.26. Distribution of element quality for the mouse model.

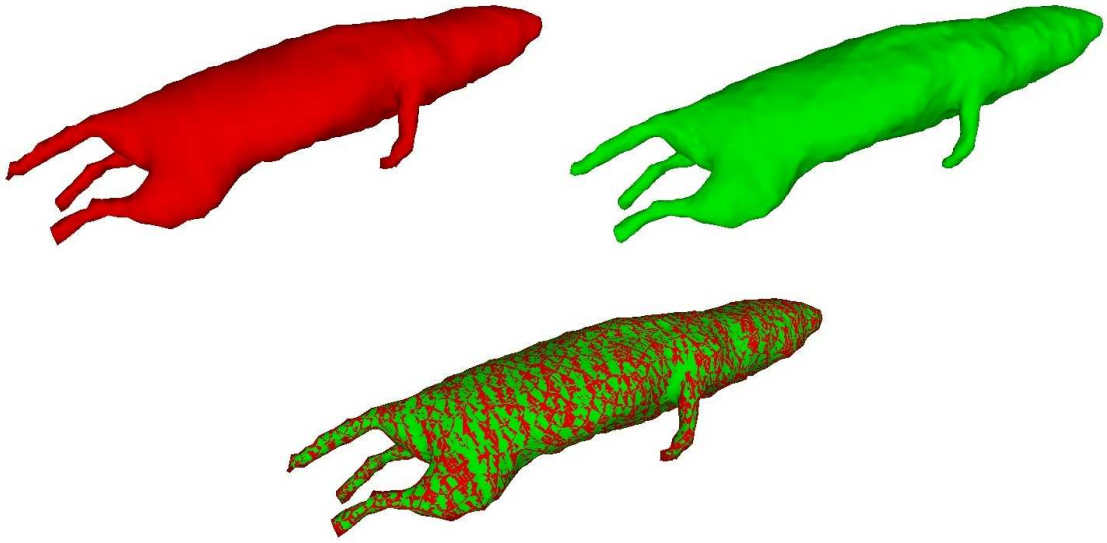

Figure 5.27. Geometry generated from original triangle facets shown in red (upper left), and the geometry generated from the hexahedral facets is shown in green (upper right). An image where both sets of facets are overlapped is given on the bottom to give an indication of the overall geometric fidelity of the hexahedral mesh.

from the hexahedral mesh are shown in green (upper right)). 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 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 5.3 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 0.17% larger than the volume enclosed by the original triangles with the additional volume gained mainly in the concave regions near the arms and legs.

Table 5.3. 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%

5.4.4 Bunny Model

The original triangle mesh for the bunny model was generated by John Schreiner using the 'afront' software [86].

The hexahedral mesh of the bunny model, shown in of Figure 5.28 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

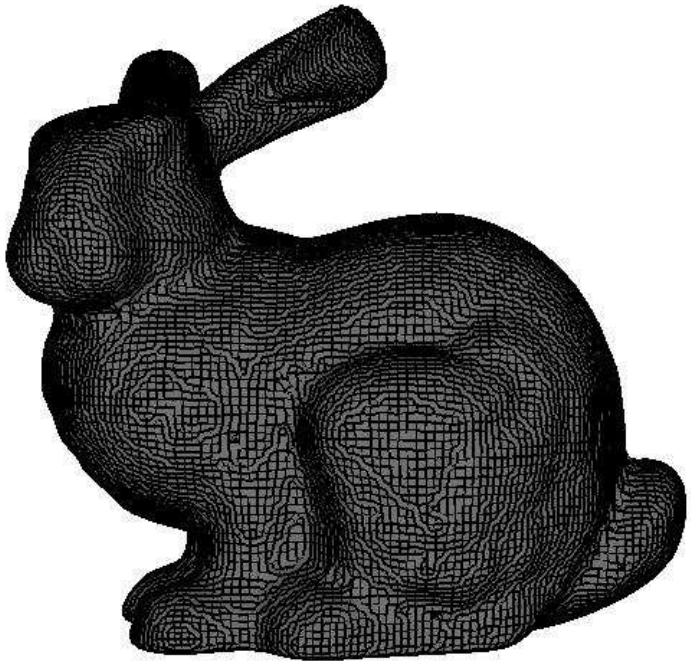

Figure 5.28. Hexahedral mesh of the bunny model, containing 125,183 elements

completion of the Laplacian smoothing operation, an optimization algorithm to improve the condition number of each of the elements was performed to give the final results shown in Figure 5.29.

Figure 5.30 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 (upper left) and the facets from the hexahedral mesh are shown in green (upper right)). 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 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 5.4 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 0.05% 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.

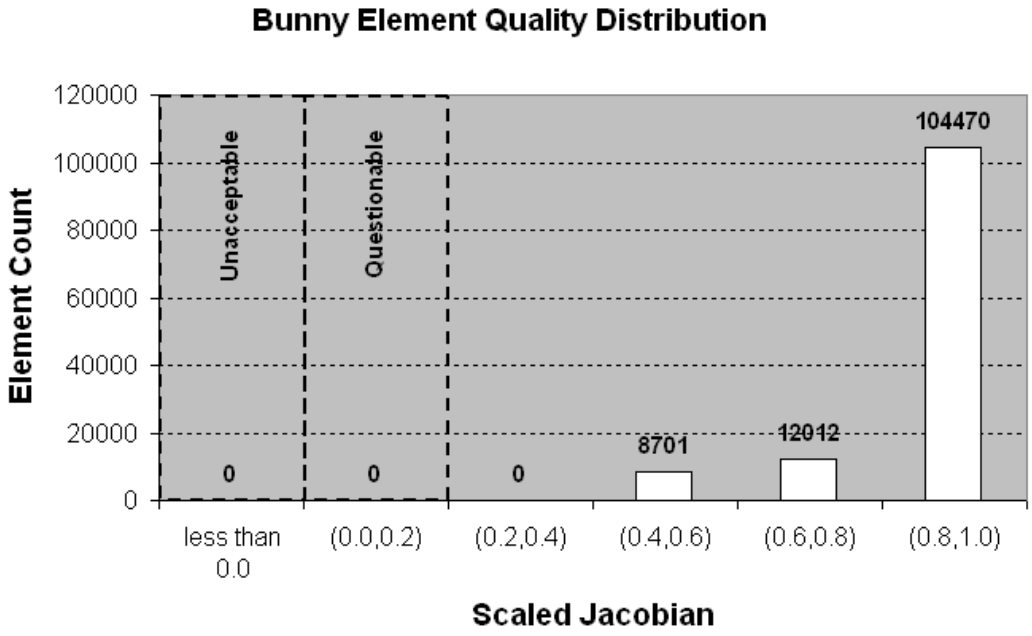

Figure 5.29. Distribution of element quality for the bunny model.

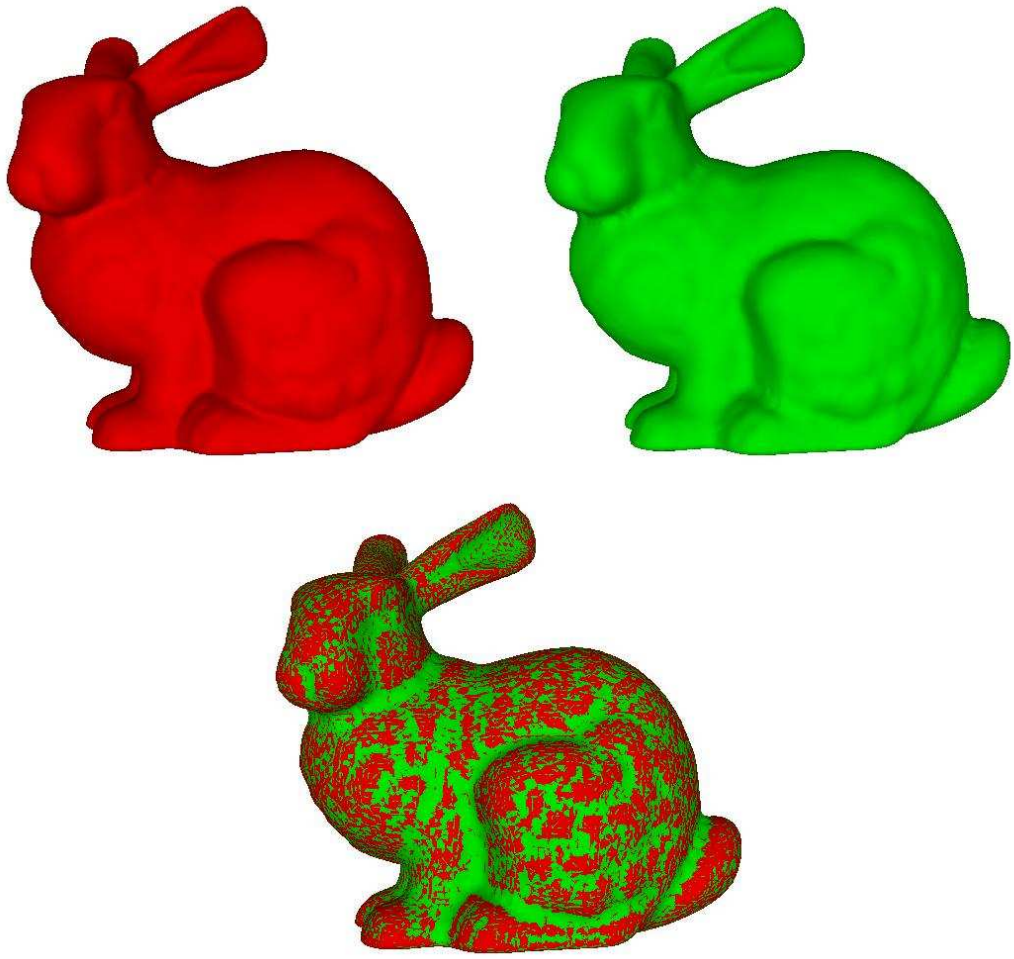

Figure 5.30. Geometry generated from original triangle facets shown in red (upper left), and the geometry generated from the hexahedral facets is shown in green (upper right). An image where both sets of facets are overlapped is given at the bottom to give an indication of the overall geometric fidelity of the hexahedral mesh.

Table 5.4. 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%

5.4.5 Triceratops Model

The hexahedral mesh of the triceratops model, shown in two separate views in Figure 5.31 contains 86,209 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 triceratops geometry. A uniform element size 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 (see Figure 5.32) as a guide, and the mesh was then exported from SCIRun and translated into a file format readable by CUBIT.

In CUBIT, a centroidal-area smoother was used on the boundary, followed by a Laplacian smoothing operation on the hexahedral elements. Mesh untangling was performed on any hexahedra that may have been inverted by the Laplacian smooth, and a single node in the upper right horn was moved by hand to remove an inverted quadrilateral. An optimization algorithm to improve the condition number of each of the elements was performed to give the final results shown in Figure 5.33. All of the questionable quality elements are located in the upper two horns. Hexahedral refinement in the region of these horns should produce a mesh that is free of elements of questionable quality with proper mesh optimization.

Table 5.5 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 0.03% smaller than the volume enclosed by the original triangles.

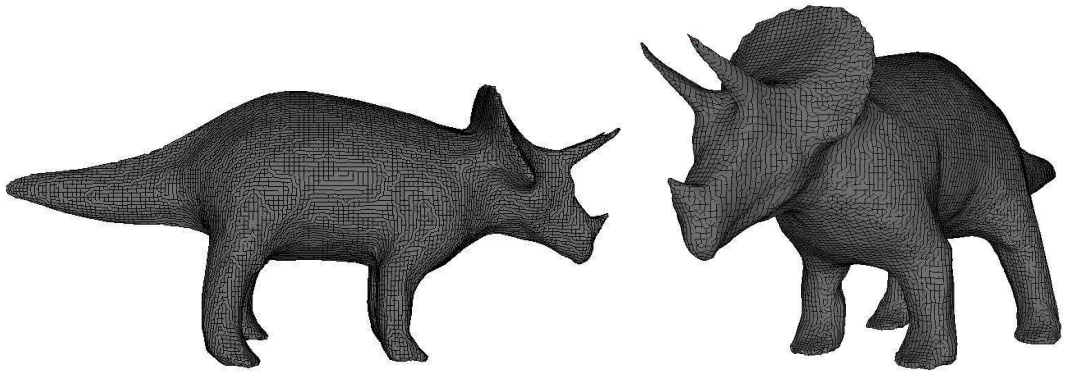

Figure 5.31. Two views of the hexahedral mesh of the triceratops model. The mesh contains 86,209 elements.

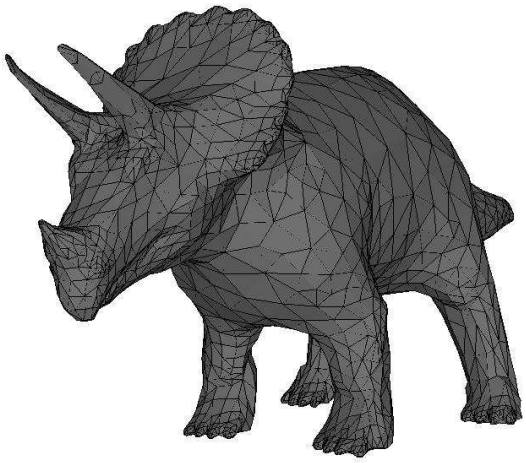

Figure 5.32. Original triangle mesh of the triceratops model.

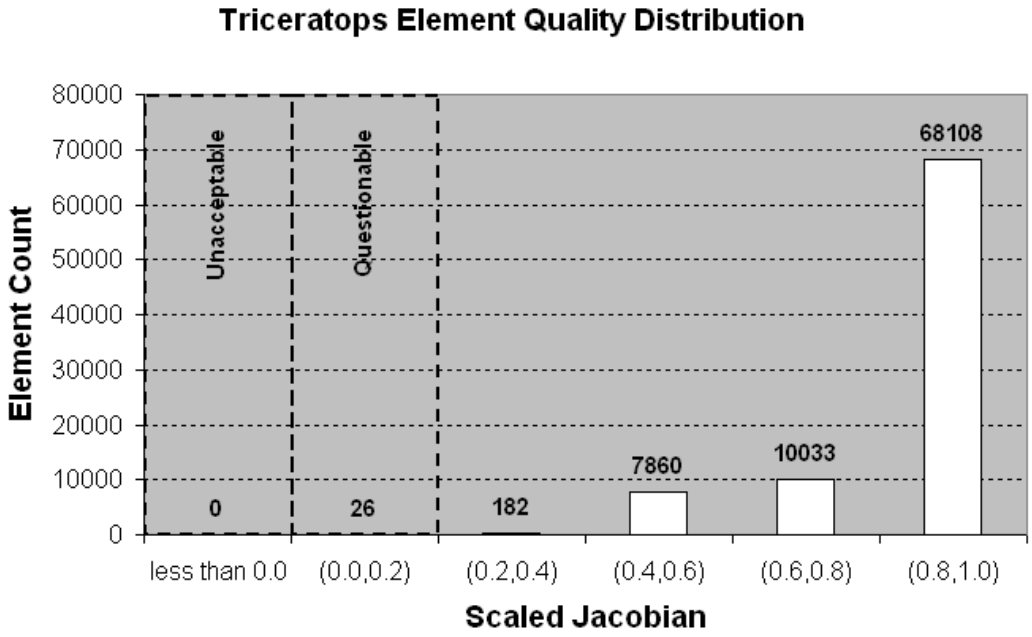

Figure 5.33. Distribution of element quality for the triceratops model.

Table 5.5. Table indicating volume changes resulting from conversion of the original triangle mesh to a hexahedral mesh for the triceratops model.

Volume (Triangle Mesh)	Volume(Hexahedral Mesh)	Difference	Percent Change
136.596	136.553	-0.0419	-0.03%

Figure 5.34 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 triceratops. Some artifacts are evident of the coarser nature of the original triangle mesh compared with the much finer hexahedral mesh. Additional refinement of the hexahedral mesh would improve the fidelity in the three horns of the triceratops.

Because the geometry in the horns is not ideal for hexahedral mesh generation, having a mesh which contains all positive and convex elements is a good demonstration of the potential flexibility and robustness of the sheet insertion algorithm. Obtaining positive and convex hexahedra in conical shapes is often difficult for existing hexahedral mesh generation algorithms. While the quality in the triceratop horns is not fully ideal, the overall quality in the triceratops model is noteworthy.

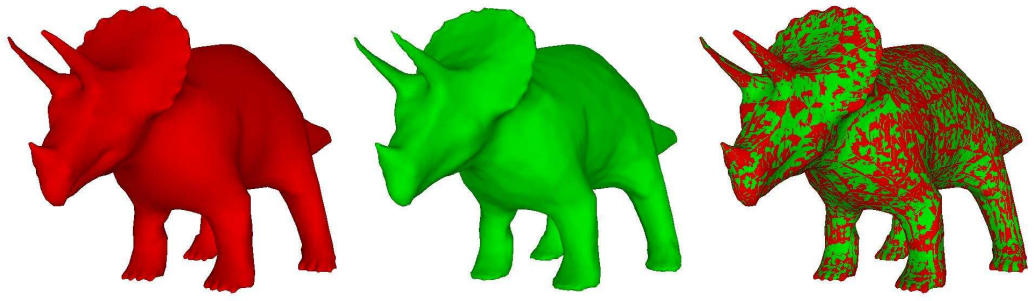

Figure 5.34. Geometry generated from original triangle facets shown in red (on the left), and the geometry generated from the hexahedral facets is shown in green (in the middle). An image where both sets of facets are overlapped is given on the right to give an indication of the overall geometric fidelity of the hexahedral mesh.

5.4.6 Dragon Model

The original triangle mesh for the dragon model was generated by John Schreiner using the 'afront' software [86].

The hexahedral mesh of the dragon model, shown in two separate images in Figure 5.35 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 Figure 5.36). 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, and the resulting mesh was then exported from SCIRun and translated into a file format readable by CUBIT.

In CUBIT, 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 Figure 5.37.

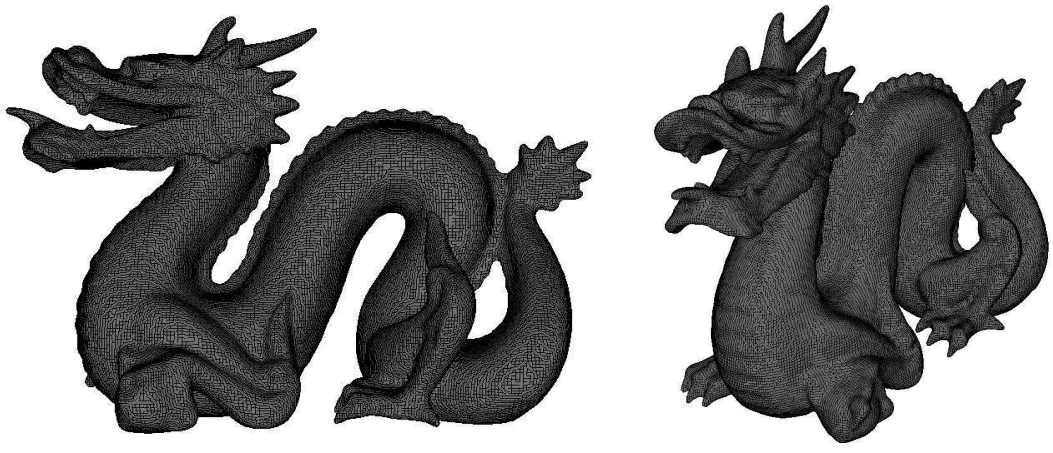

Figure 5.35. Two views of the hexahedral mesh of the dragon model. The mesh contains 465,527 elements.

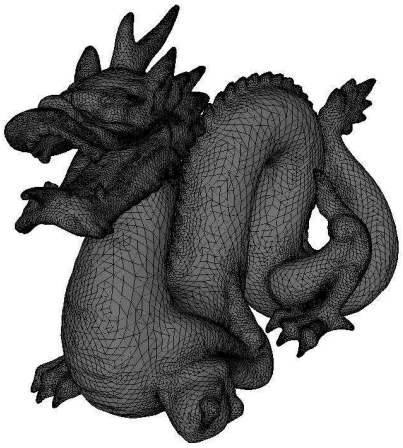

Figure 5.36. Original triangle mesh of the dragon model.

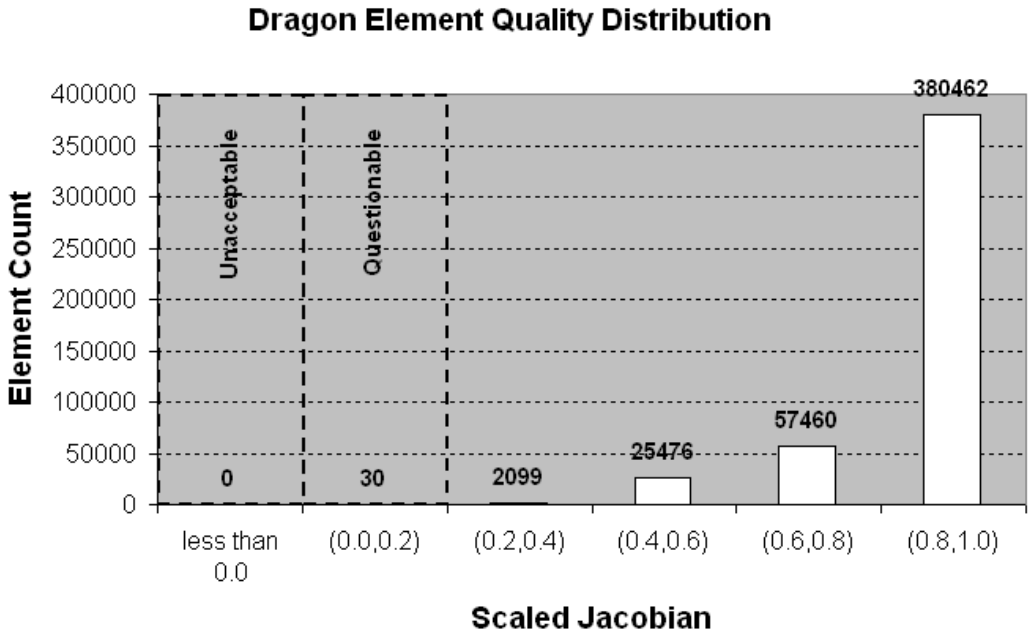

Figure 5.37. Distribution of element quality for the dragon model.

Figure 5.38 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 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. The possible improvements will be discussed in greater detail in Chapter 7.

Table 5.6 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%

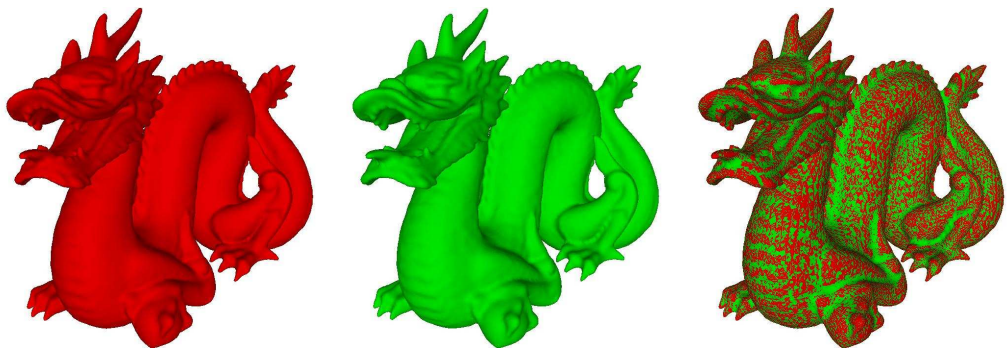

Figure 5.38. Geometry generated from original triangle facets shown in red (on the left), and the geometry generated from the hexahedral facets is shown in green (in the middle). An image where both sets of facets are overlapped is given on the right to give an indication of the overall geometric fidelity of the hexahedral mesh.

Table 5.6. 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%

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.

5.4.7 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 Figure 5.39 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 Figure 5.40) as a guide, and the mesh was then exported from SCIRun and translated into a file format readable by CUBIT.

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 quadri-

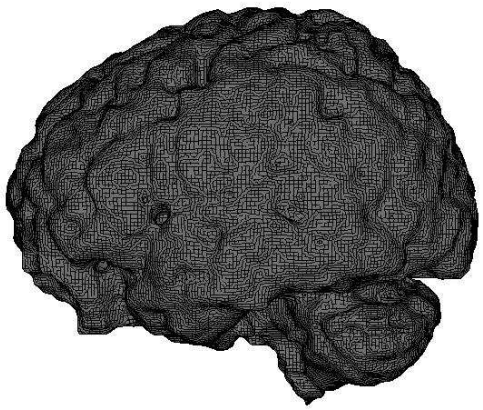

Figure 5.39. Hexahedral mesh of the brain model, containing 644,221 elements.

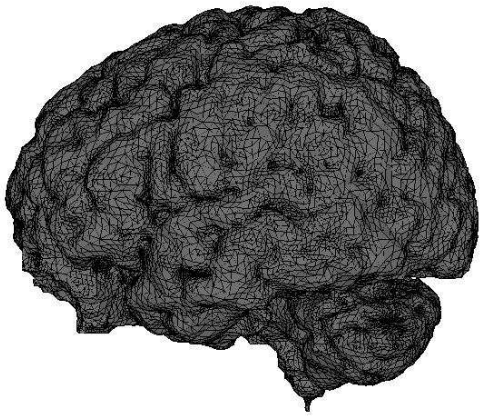

Figure 5.40. Original triangle mesh of the brain model.

lateral 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 Laplacian smoothing operation, an optimization algorithm to improve the condition number of each of the elements was performed to give the final results shown in Figure 5.41.

Table 5.7 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 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 5.42 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 structure of the brain as evident in Figure 5.43. The loss of the internal cavities accounts for the bulk of the volume gain shown in Table 5.7.

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

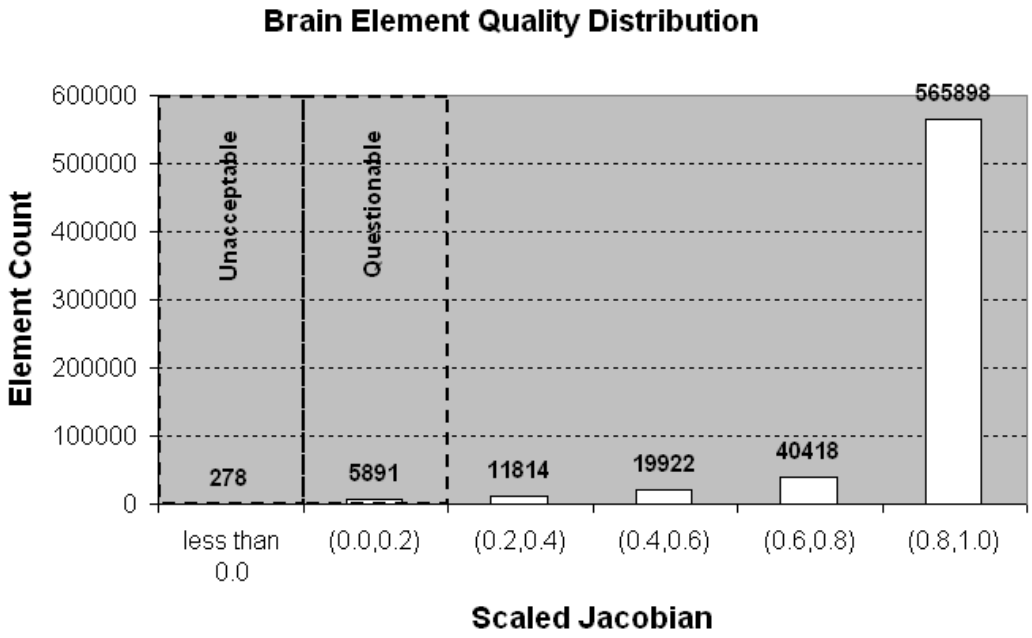

Figure 5.41. Distribution of element quality for the brain model.

Table 5.7. 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%

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 Figure 5.44). Excluding

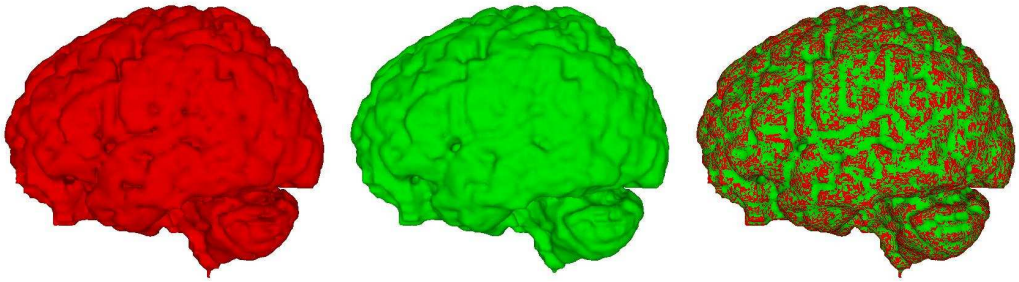

Figure 5.42. Geometry generated from original triangle facets shown in red (on the left), and the geometry generated from the hexahedral facets is shown in green (in the middle). An image where both sets of facets are overlapped is given on the right to give an indication of the overall geometric fidelity of the hexahedral mesh.

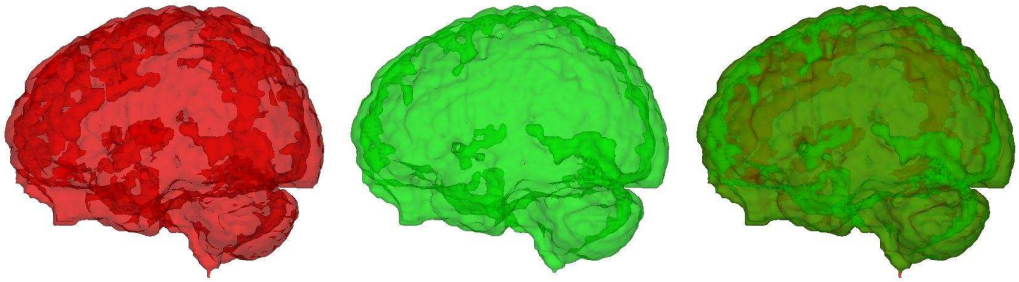

Figure 5.43. Transparent view to show internal structure of the geometry generated from original triangle facets shown in red (on the left), and the geometry generated from the hexahedral facets is shown in green (in the middle). An image where both sets of facets are overlapped is given on the right to give an indication of the overall geometric fidelity of the hexahedral mesh.

anisotropic mesh sizing, additional options for detection and removal of the nonmanifold elements will be discussed further in Chapter 7.

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 homeomorphism to the triangle mesh is only requirement, better 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. This problem will also be discussed in more detail in Chapter 7.

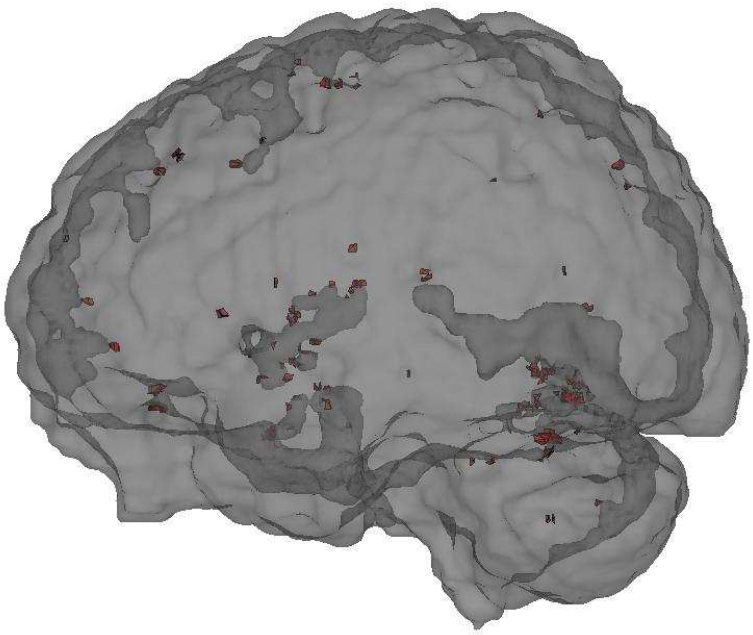

Figure 5.44. Locations of negative scaled Jacobian elements in the brain model.

CHAPTER 6

MULTISURFACE HEXAHEDRAL MESH GENERATION

In the previous chapter, we demonstrated hexahedral meshing of complex geometric solids that are defined by a single surface. In this chapter, we demonstrate hexahedral mesh creation on geometric solids that are bounded by more than one surface using a similar methodology as used earlier. In essence, we will be demonstrating insertion of multiple sheets into existing hexahedral meshes to obtain a fundamental mesh (refer to Chapter 4) for a given solid.

6.1 Introduction

As stated in Chapter 2, 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 surface with zero curves can be considered an isosurface as described in the previous chapter. In this chapter, 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 Figure 6.1):

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.

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 chapter will be general enough to account for this ambiguity; however, the examples presented in this chapter will, for the most part, ignore the cases of curves

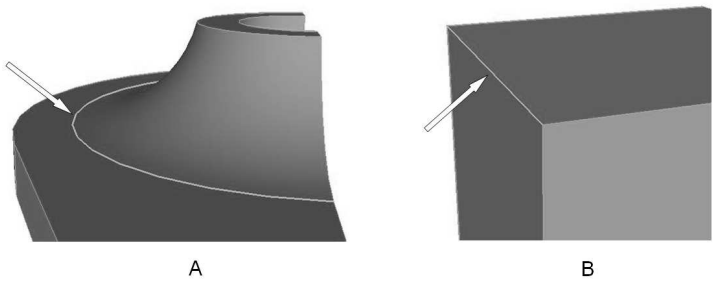

Figure 6.1. The highlighted curve in the image on the left could be considered a soft curve, while the highlighted curve in the image on the right would be considered a hard curve.

that can be defined nearly unambiguously as being 'soft curves' on the boundary of the geometry.

6.1.1 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 Figure 6.2 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 (see Chapter 5), 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.

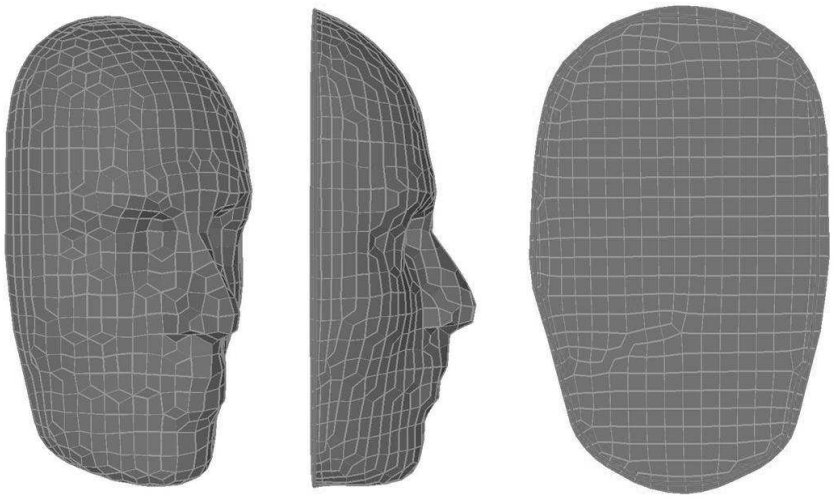

Figure 6.2. In locations where two sheets intersect, the resulting mesh topology contains a string of edges that can be aligned with sharp features, or hard curves. In this image, we 'cut' the face from the head model by inserting a planar sheet behind the face. The inserted boundary sheet capturing 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.

In Figure 6.3 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 6.4 lists the resulting quality of each of the elements demonstrating the overall high quality of the resulting mesh.

In SCIRun [92], we utilize the same algorithm that was developed for the hexahedral isosurfacing (as described in Chapter 5) 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. It should be noted, however, that SCIRun does not currently have a data structure for creating and storing solid-modeling geometric hierarchies beyond volumes created by isosurfaces. Specifically, the solid-model and curve representations are needed for boundary smoothing of the resulting mesh without destroying the geometric fidelity created by the multisheet insertion process.

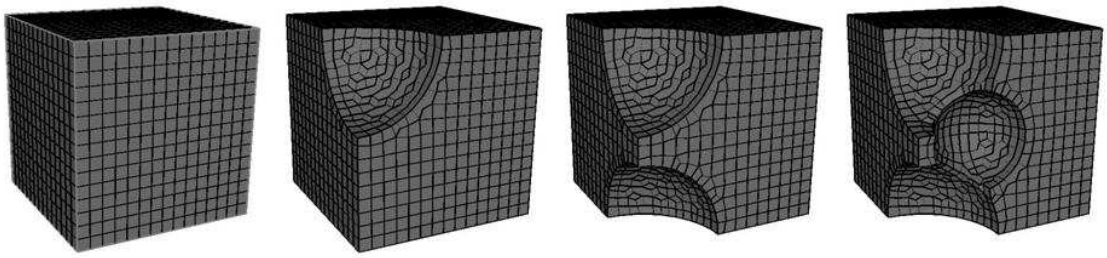

Figure 6.3. By inserting spherical sheets into the geometry, we can perform Boolean-like operations in the mesh, while maintaining the integrity of the hexahedral mesh. At each of the boundary surfaces, the intersection of the spherical sheet with the original planar sheets in the cuboid mesh is sufficient 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.

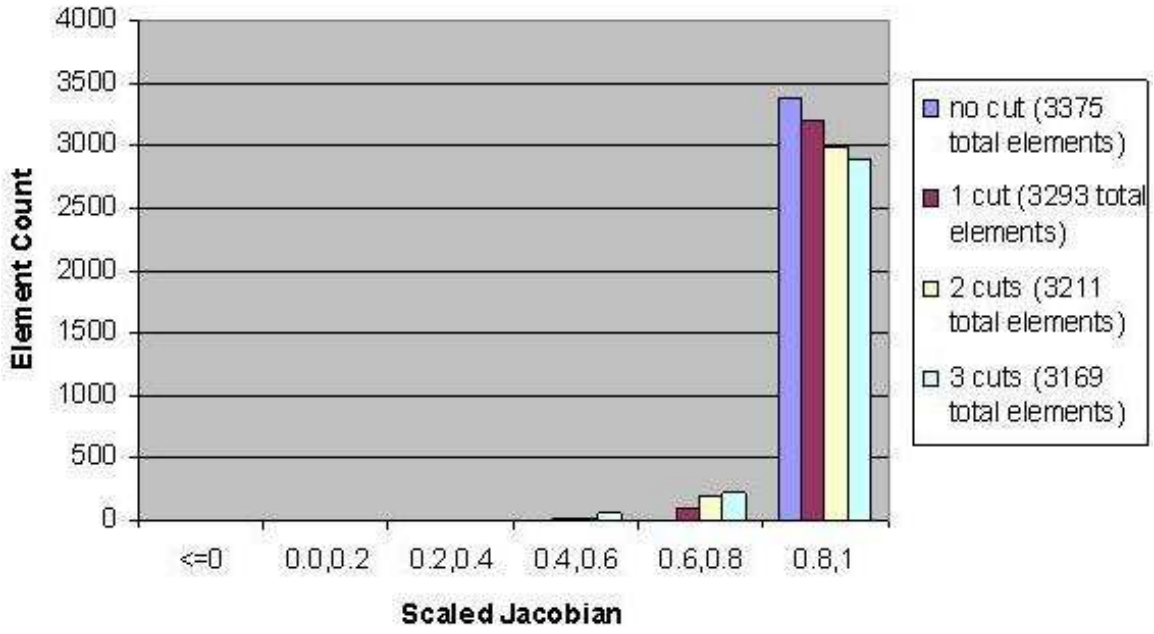

Figure 6.4. The distribution of scaled Jacobian values for the cuboid geometry with the spherical cutouts shown in Figure 6.3.

To overcome this problem, where needed, we have created the appropriate mesh topologies in SCIRun, and then exported these meshes to CUBIT. Within CUBIT, we utilized some feature recognition software [78] that creates the appropriate solid model data structures directly from the mesh topology.

Additionally, we have modified some software we developed previously in CUBIT [14] to perform these operations directly in CUBIT. The MeshCutting algorithm in CUBIT uses a similar methodology for placing the sheets but couples the sheet insertion with an actual solid-model Boolean operation to create the geometry and mesh simultaneously. Research on the MeshCutting algorithm is still on-going in CUBIT, and still results in many failures in creating a correct solid model. Common points of failure in the CUBIT algorithm include: 1) Lack of robustness in the algorithm for determining the how to separate hexahedra into separate sides for pillowing, 2) removal of nonmanifold elements in the separated sides, 3) errors in nodal projections back to the original cutting surface, 4) lack of robustness in the solid model Boolean/cutting operations, and 5) inconsistencies in creating the new sheet of elements. We have migrated some of the code generated in this research from SCIRun into the CUBIT MeshCutting algorithms to improve robustness and also to take advantage of the solid-modeling hierarchies that are native to the CUBIT framework. Specifically, the goal in moving the algorithm to CUBIT was to take advantage of the geometric hierarchies native to CUBIT to enable robust multisurface hexahedral meshes. Migrating some of the code from SCIRun into CUBIT has dramatically improved the robustness of CUBIT's MeshCutting algorithm; however, currently the nonmanifold element prevention algorithm, nodal projection algorithms, and pillowing algorithms from SCIRun have not been fully migrated into the CUBIT code base. Migrating these additional algorithms and improving the robustness of CUBIT's MeshCutting algorithm is still an on-going research effort.

The remainder of this chapter demonstrates hexahedral meshing on several multi-surface 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.

6.2 Results

6.2.1 Mechanical Part

The hexahedral mesh of the mechanical part model, shown in Figure 6.5, was generated in CUBIT and contains 27,486 hexahedra. The original geometry contains two soft curves that are shown in Figure 6.6. 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 (see Image A in Figure 6.7). 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 Figure 6.7. These triangle meshes were created directly on the original surfaces using an advancing front 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

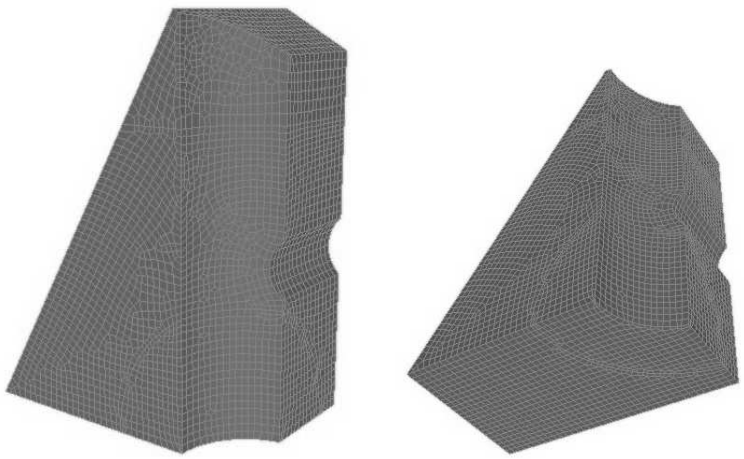

Figure 6.5. Hexahedral mesh of the mechanical part model showing images from the side and bottom of the mesh.

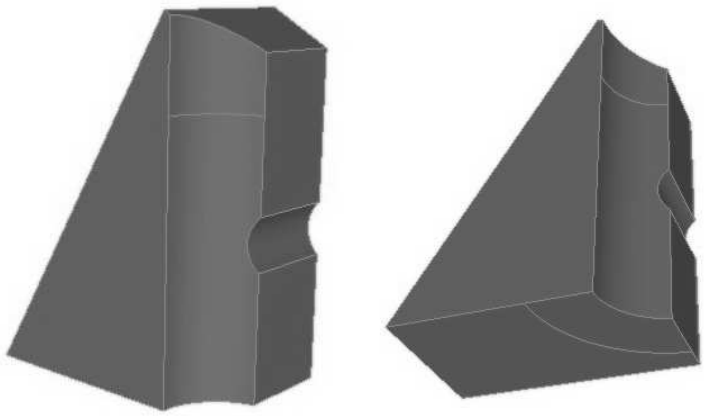

Figure 6.6. Geometry for the mechanical part showing two soft curves: one in the upper cylindrical section and a second on the base of the model.

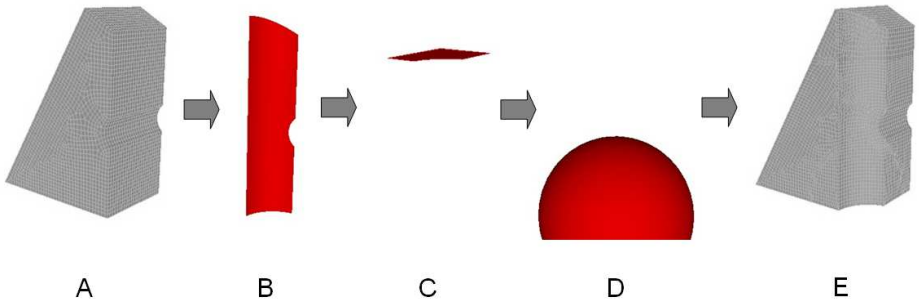

Figure 6.7. Flow chart showing the sheet insertion steps to create the mesh for the mechanical part. The red surfaces represent hexahedral sheets that were inserted into the simplified hexahedral mesh on the left to create the final hexahedral mesh on the right.

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 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 Figure 6.8

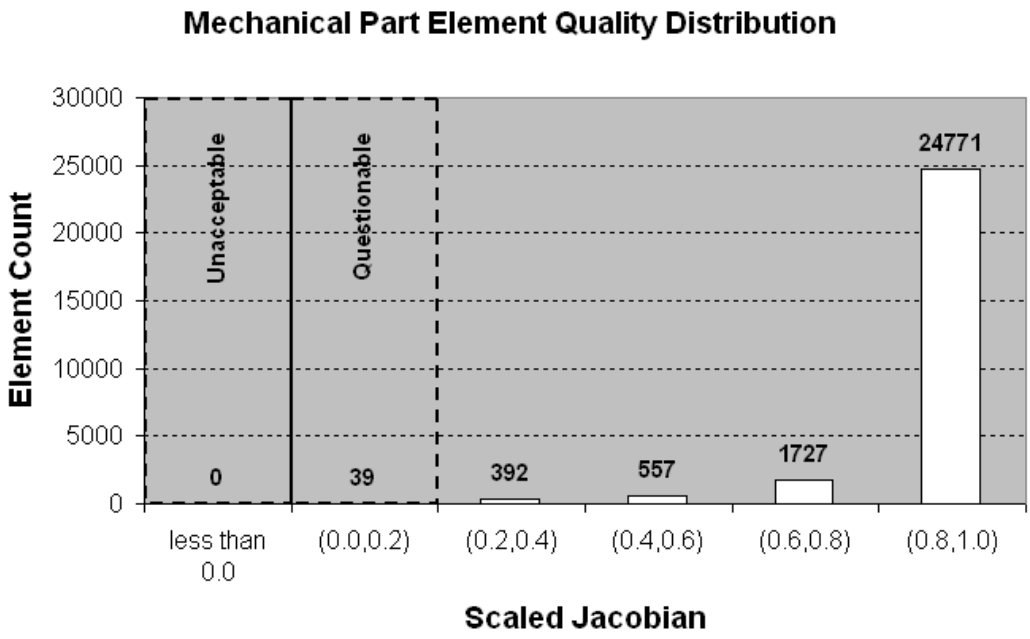

Figure 6.8. Distribution of element quality for the mechanical part model.

6.2.2 Spring Model

The hexahedral mesh of the spring model, shown in Figure 6.9, was generated in SCIRun and contains 6,908 hexahedra. This model was generated by placing a triangle mesh describing the geometry of the spring in a regular grid of hexahedra and inserting two hexahedral sheets using the triangle mesh to guide the placement of the newly formed hexahedra. Two additional sheet insertions were made at the two ends of the spring to obtain a flat surface to cap the ends of the spring.

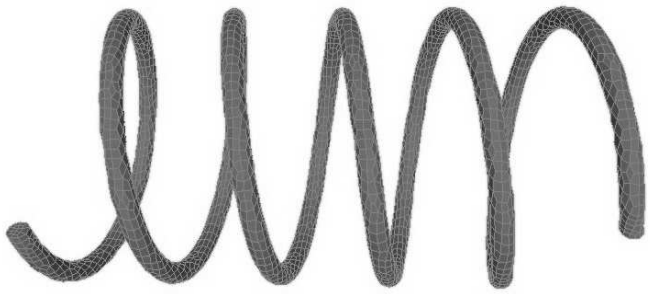

Figure 6.9. Hexahedral mesh of the spring model.

A hexahedral mesh of this model can also be generated with a sweeping algorithm; however, to ensure high quality hexahedra as the mesh is projected around the coils of the spring, it is common to decompose the model into several smaller pieces. We generated the mesh with sheet insertion to demonstrate the flexibility of the sheet description to capture the geometry. Additionally, the sheet insertion method enables generation of the inverse mesh of the region surrounding the spring which would be extremely difficult for traditional hexahedral methods to accomplish. In this example, the inverse mesh is discarded, however a contiguous mesh of the interior and exterior space defined by the spring and the initial bounding box does exist for this model.

After discarding the exterior hexahedral elements, the resulting mesh on the boundary of the spring was then smoothed using a Laplacian smoother on the quadrilaterals, followed by a Laplacian smoothing of the hexahedra. Following the smoothing of the hexahedral elements a mesh untangling algorithm was utilized on any remaining inverted elements followed by a mesh optimization using a condition number metric to obtain the scaled Jacobian distribution shown in Figure 6.10.

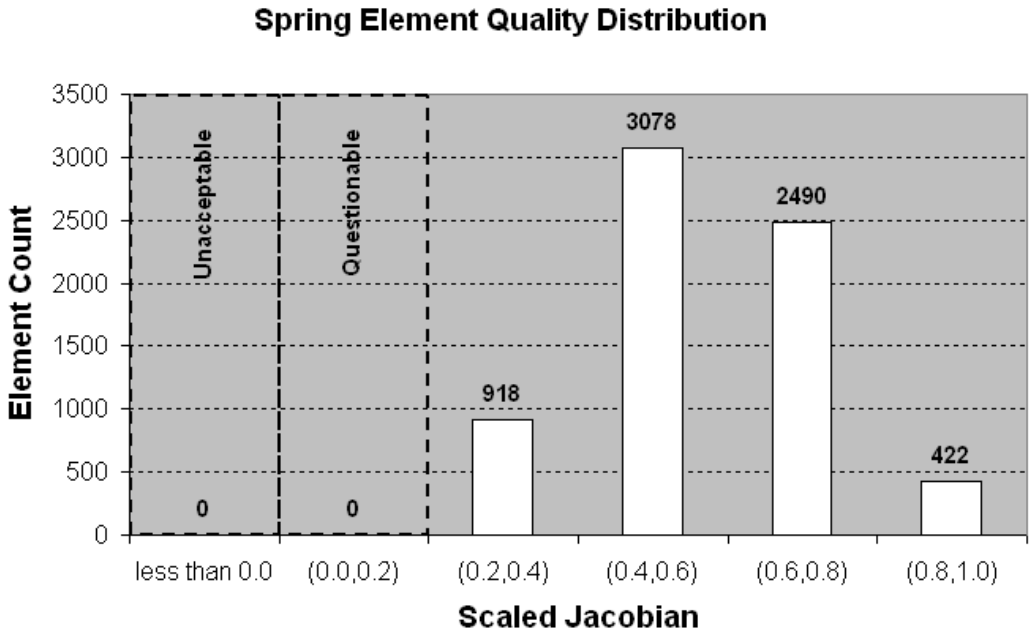

Figure 6.10. Distribution of element quality for the spring model.

6.2.3 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 sweeping or blocking methods are not present. With methods that are commonly available for performing hexahedral mesh generation, this model would be extremely difficult to produce.

The hexahedral mesh of the skull model, shown in Figure 6.11, 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 Figure 6.12.

This model was generated by placing a triangle mesh describing the geometry of the

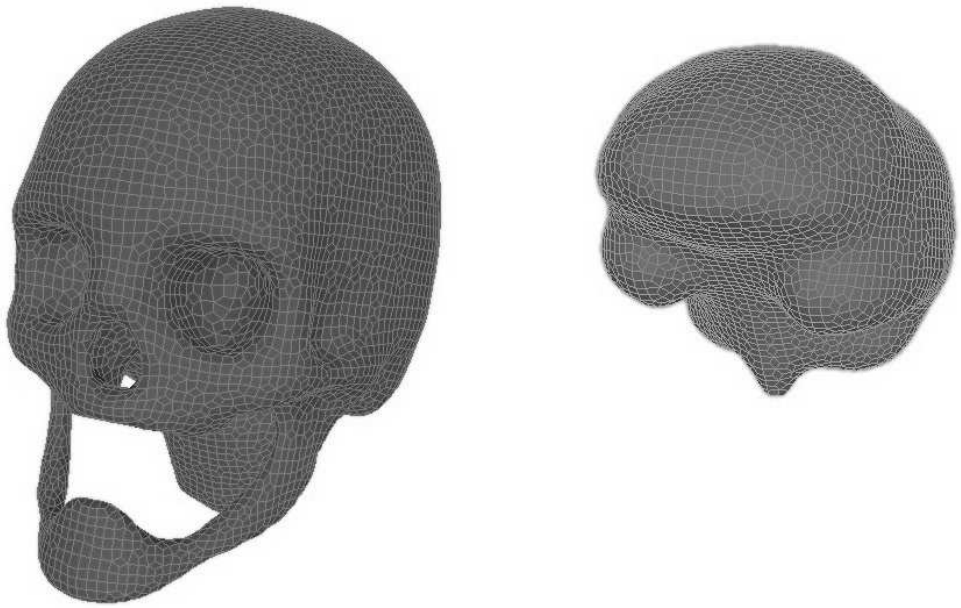

Figure 6.11. Hexahedral mesh of the skull model. Bone (left) and cranial cavity (right) meshes are shown separately.

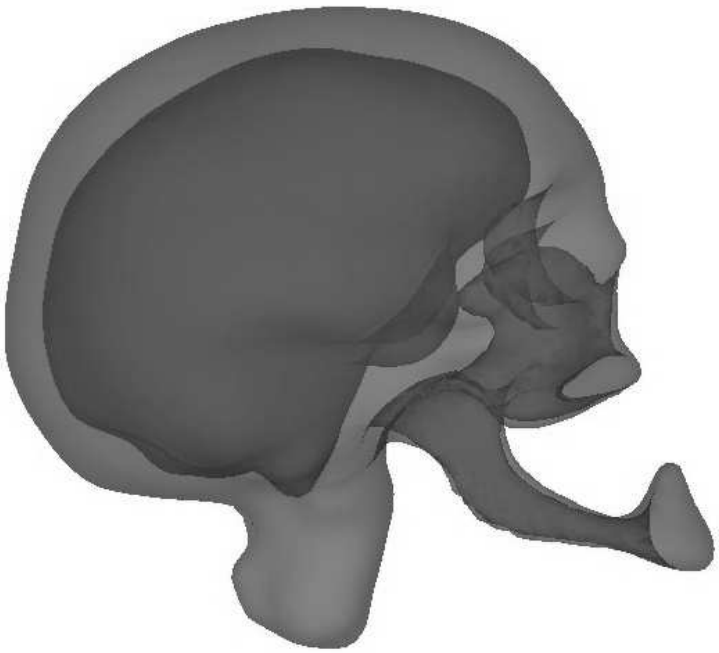

Figure 6.12. Transparent view of the combined geometry generated from the facets of the hexahedral mesh of the skull model.

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 Figure 6.13.

These two groups of hexahedral elements were then exported from SCIRun and imported into CUBIT. In CUBIT, smoothing and optimization of the mesh was performed. First, centroidal area smoothing was performed 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 Figure 6.14.

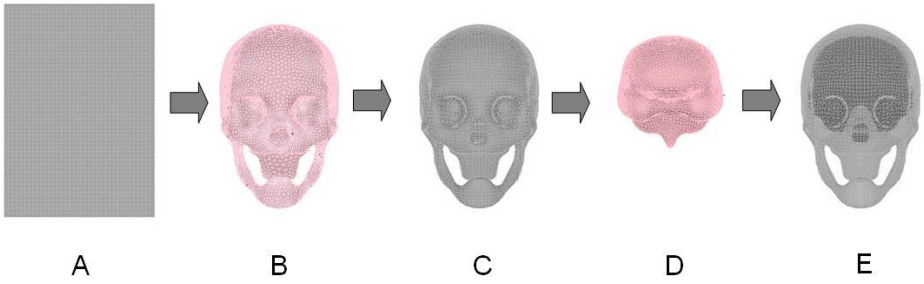

Figure 6.13. Pictorial flow chart demonstrating the mesh generation process for creating the hexahedral mesh of the skull. Triangle meshes (pink) are utilized to guide placement of hexahedral sheets into existing hexahedral meshes to achieve new meshes that are conformal with the original solid geometry.

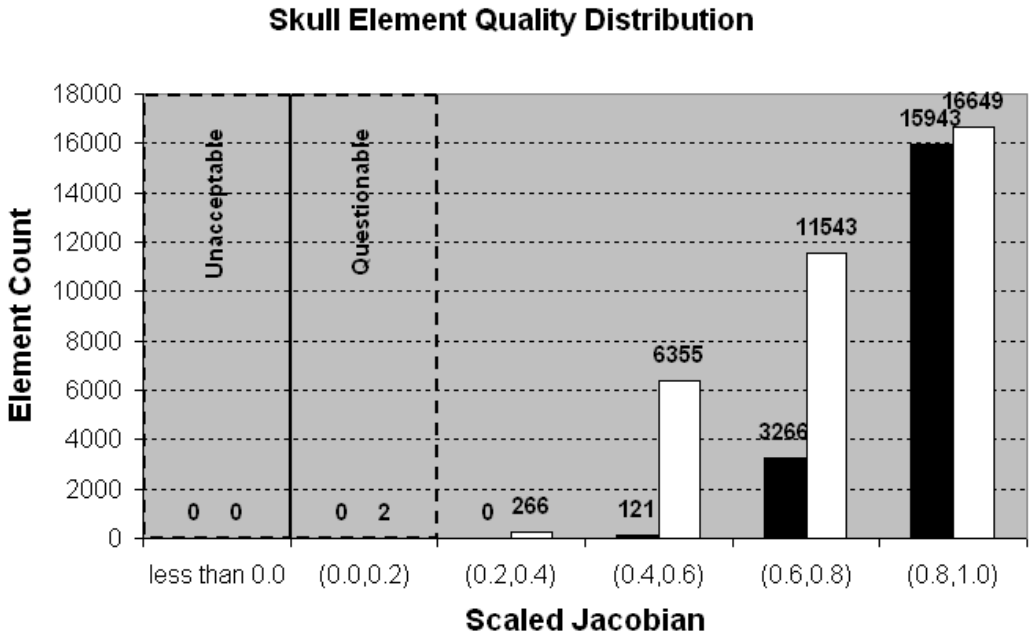

Figure 6.14. Distribution of element quality for the skull model (bone is shown in white and cranial cavity is shown in black.

6.2.4 Gear Model

The gear model is provided courtesy of Tim Tautges by the AIM@SHAPE Shape Repository (http://shapes.aim-at-shape.net/index.php). Generating a hexahedral mesh on this model is difficult because of the toroidal grooves on the top and bottom of the gear. The gear model would be readily multisweepable if the grooves on the top and bottom of the gear were not toroidal grooves but were squared off.

The hexahedral mesh of the gear model, shown in Figure 6.15, was generated in CUBIT and contains 269,028 hexahedra. The first step of generating this mesh was to simplify the geometry which readily accepts a mesh generated by sweeping. To simplify this geometry, first the toroidal grooves on the top and bottom of the gear were removed, and then the resulting surfaces on the top and bottom of the gear were extruded to create a single level for the top and bottom surfaces. The multisweeping [96] algorithm in CUBIT was used to generate a hexahedral mesh of the simplified geometry. (Multisweeping of this part was necessary due to the slight bevel on each of the individual gear teeth.) Due to several fits and starts of trying to get the multisweeping algorithm to generate the initial mesh, a much finer element size was utilized than desired. A sheet extraction algorithm was utilized following the multisweep meshing to cut the original element count from 700,000 hexahedra to 300,000 hexahedra.

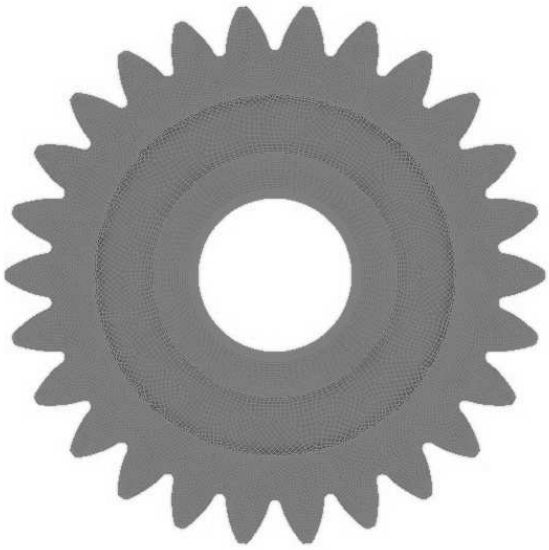

Figure 6.15. Hexahedral mesh of the gear model.

Once this initial coarsening of the mesh was accomplished, two sets of toroidal sheets (matching the original unsimplified geometry) were inserted and the exterior elements were discarded (refer to Figure 6.16 for a pictorial presentation of the meshing process for the gear). In the original model, the hub in the center of the gear is elevated from the plane of the rest of the gear (his feature was originally removed during the geometry simplification process). Two additional sets of planar sheets were then inserted and the exterior elements were discarded to obtain the final mesh topology for the model that matched the original unsimplified geometric description.

The boundary of the resulting mesh was smoothed using a Laplacian smoothing on the quadrilaterals, followed by a Laplacian smoothing of the hexahedral elements. Mesh untangling was then performed on the remaining elements with negative Jacobian metrics, and the entire mesh was optimized using both condition number and mean-ratio metrics. The final distribution of scaled Jacobian metrics is shown in Figure 6.17.

6.2.5 Gfwtp Model

The gfwtp model is provided courtesy of Tim Tautges by the AIM@SHAPE Shape Repository (http://shapes.aim-at-shape.net/index.php). Generating a hexahedral mesh in the gftwp model is difficult for traditional hexahedral meshing methods due to the numerous possible, but conflicting, sweep directions. Because of the conflicts that arise when determining a possible direction to extrude, extensive decomposition is required to generate a mesh with commonly available hexahedral algorithms. Decomposition of this part will also be difficult for any traditional structured hexahedral meshing algorithms due to the number of sweep direction associated with the model.

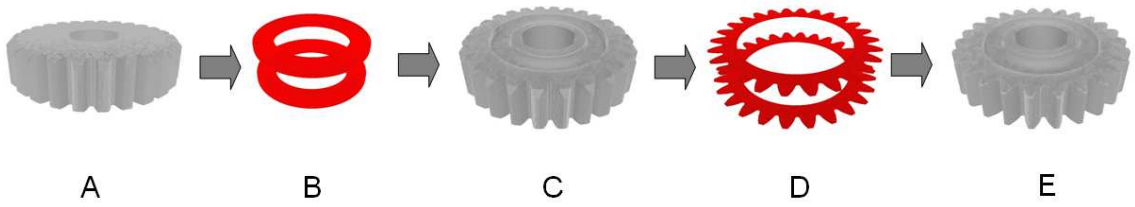

Figure 6.16. Pictorial flow chart demonstrating the mesh generation process for creating the hexahedral mesh of the gear. Triangle meshes (red) are utilized to guide placement of hexahedral sheets into the existing hexahedral meshes to achieve new meshes that are conformal with the original solid geometry.

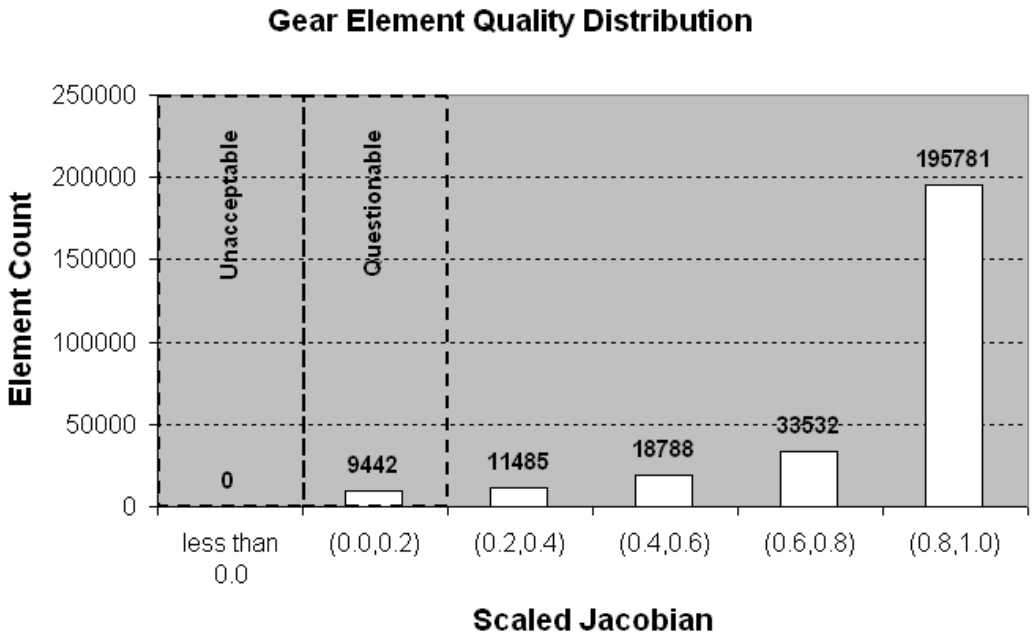

Figure 6.17. Distribution of element quality for the gear model.

The hexahedral mesh of the gfwtp model, shown in Figure 6.18, was generated in CUBIT and contains 19,927 hexahedra. This mesh was generated by first creating a mesh on a simplified model (shown in Figure 6.19) and then inserting sheets to capture the planar and cylindrical surfaces at the top of the model. Because the current mesh-cutting algorithm does not allow solid-modeling Boolean operations on multiple volumes currently, this mesh proved to be slightly difficult to generate. To obtain the mesh, the planar sheet was inserted into the original swept mesh, followed by the cylindrical surface with a hemispherical cap (shown in Figure 6.19). The hemispherical cap was added to the cylindrical surface to make the inserted sheet manifold within the hexahedral mesh. Once both sheets had been inserted, the elements exterior to the volume were removed, and the new and correct mesh topology was smoothed to the curves and surfaces on the new volume.

After fixing the mesh edges on the curves in place, the new surfaces were Laplacian smoothed, followed by a Laplacian smooth of the hexahedra. Mesh optimization was also performed using the mean-ratio metric for each hexahedral element. The resulting distribution of mesh quality based on the scaled Jacobian metric is shown in Figure 6.20.

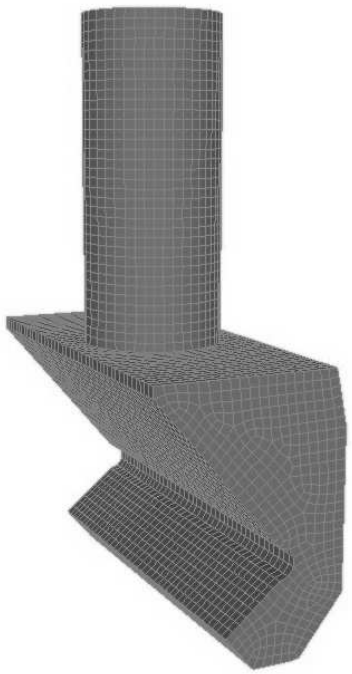

Figure 6.18. Hexahedral mesh of the gfwtp model.

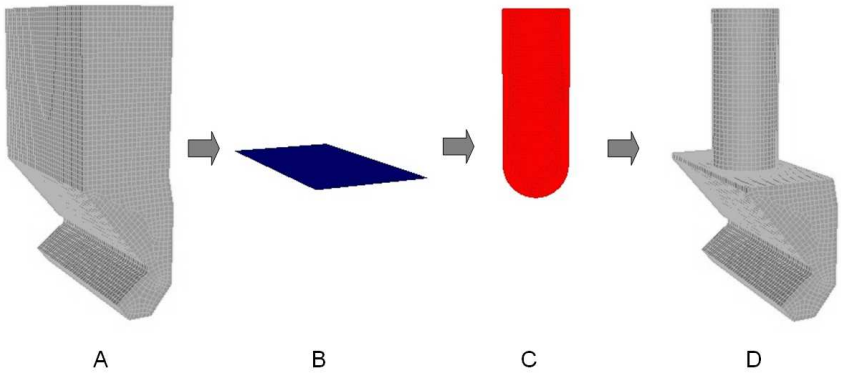

Figure 6.19. Pictorial flow chart demonstrating the mesh generation process for creating the hexahedral mesh of the gfwtp. Triangle meshes (blue and red) are utilized to guide placement of hexahedral sheets into the existing hexahedral meshes to achieve a new mesh that is conformal with the original solid geometry.

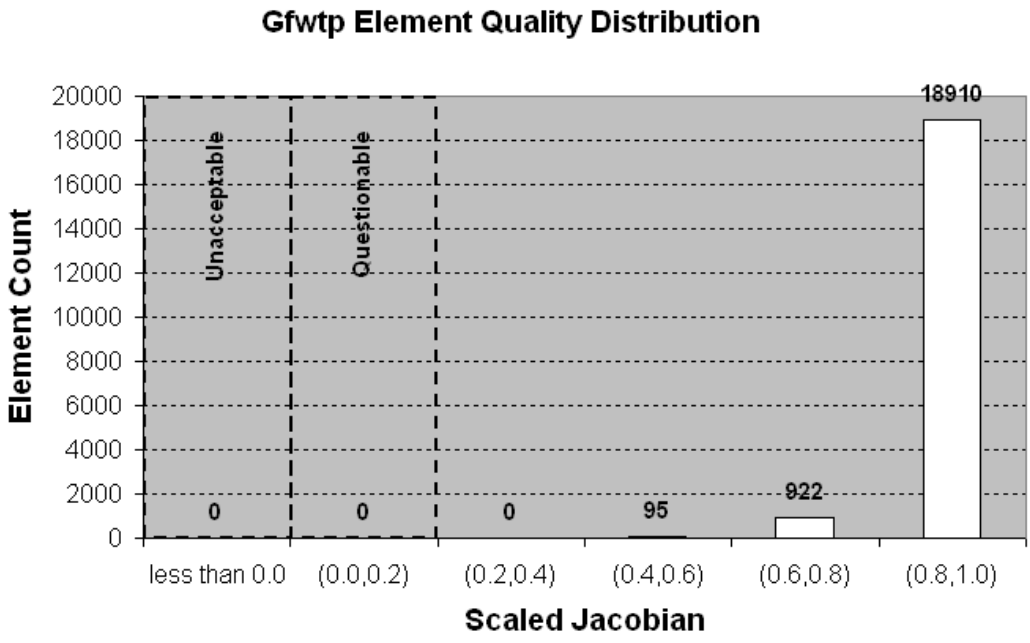

Figure 6.20. Distribution of element quality for the gfwtp model.

6.2.6 Engine Model

The Engine model is provided courtesy of ANSYS [2]. Hexahedral mesh generation of this model is difficult due to the coupling of the interior mesh with the actual engine block geometry. Decomposition of both pieces separately may be accomplished resulting in primitive objects which readily accept a hexahedral mesh; however, the coupling of both pieces in the hexahedral mesh requires a significant amount of decomposition to generate the conformal mesh of both the engine block and the interior chamber of the engine.

The hexahedral mesh of the engine model, shown in Figure 6.21, was generated using algorithms in both SCIRun and CUBIT, and contains 224,496 hexahedra in the engine casing and an additional 274,095 hexahedra in the interior of the engine. The mesh is completely conformal throughout the model, but is separated into the two material blocks. A transparent view of the geometry showing the engine casing and interior features is given in Figure 6.22.

The hexahedral mesh of the engine was generated by first creating a simplified geometry that could be meshed using a sweeping algorithm in CUBIT. Essentially, a cube was placed around the lower portion of the engine exposing part of the engine neck before the heat fins on the upper neck. A hexahedral mesh of this simplified geometry was created, then, in SCIRun, a set of sheets was placed to capture the features of the lower external engine geometry. The hexahedral elements external to the engine were discarded,

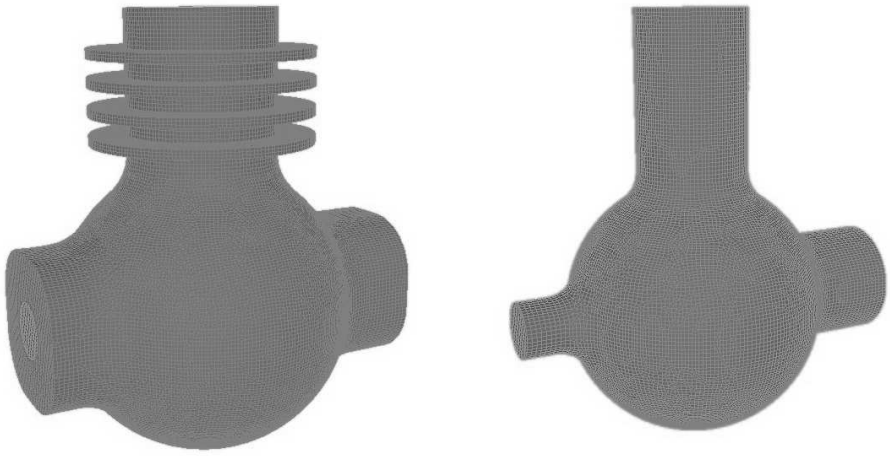

Figure 6.21. Hexahedral mesh of the engine model. The interior hexahedral mesh for the engine model is shown on the right.

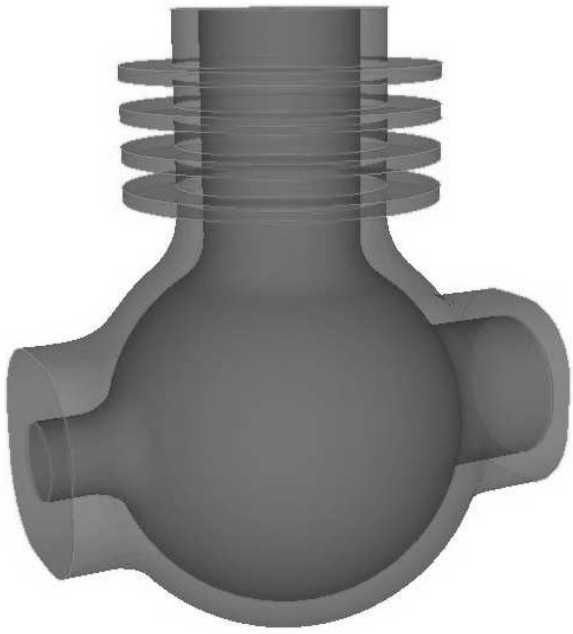

Figure 6.22. Transparent view of the engine and interior created from the facets of the hexahedral mesh of the engine model.

as shown in the process flow diagram in Figure 6.23. The placement of the first sheet resulted in a ring of doublets where the original sheet in the neck below the bottom heat fin and the newly inserted sheet overlapped. A boundary face collapse operation (refer to Chapter 4 for a description of this operation) was used to collapse this ring of hexahedron.

In CUBIT, the boundary of this mesh was smoothed using a centroidal area smoothing algorithm and the hexahedra were smoothed using Laplacian smoothing, followed by mesh untangling and optimization against the condition number metric. A second set of sheets was then inserted, using CUBIT, to capture the features of the internal engine. The new internal boundary was smoothed with centroidal area smoothing, and the hexahedra were again smoothed with a Laplacian smoothing algorithm. The engine casing had some additional mesh-untangling, and both meshes were then optimized against the condition number metric. The distribution of element quality for the scaled Jacobian metric for the engine mesh is shown in Figure 6.24.

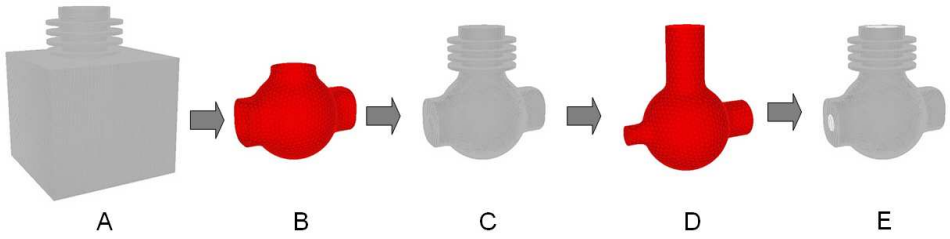

Figure 6.23. Pictorial flow chart demonstrating the mesh generation process for creating the hexahedral mesh of the engine. Triangle meshes (red) are utilized to guide placement of hexahedral sheets into the existing hexahedral mesh to achieve a new mesh that is conformal with the original solid geometry.

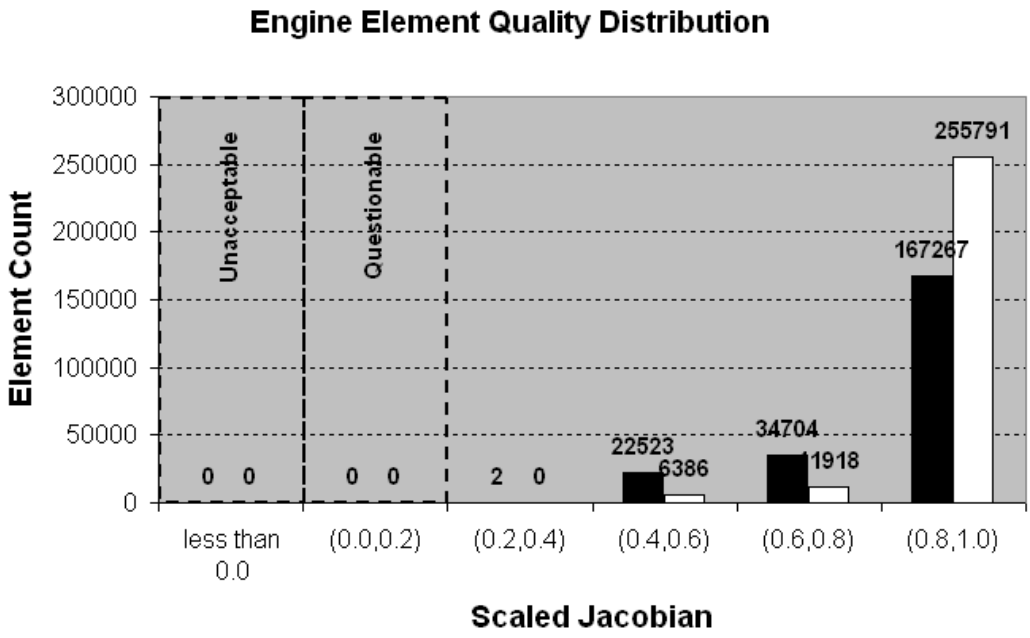

Figure 6.24. Distribution of element quality for engine model (hexahedral statistics for the interior of the engine are shown in white, and statistics for the engine casing are shown in black.

6.2.7 Part29 Model

The Part29 model is provided courtesy of ANSYS [2], and is not readily sweepable, nor does it contain any recognizably dominant hexahedral primitive shapes. Additionally, the curve along the bottom of the part fans out to a smooth fillet, which often results in poor quality hexahedra near the transition from a sharp to a smooth feature. The smooth fillets surrounding the model would traditionally be removed to enable more recognizable primitive shapes.

The hexahedral mesh, shown in Figure 6.25 contains 17,386 hexahedra. To generate this mesh, a simplified, multisweepable model was created (as shown in Figure 6.26) and meshed in CUBIT. A set of sheets, similar in shape to the upper surface, was inserted into the mesh, and the elements above this sheet were discarded. Then, a cylindrical hole was added by inserting an additional set of sheets using a triangle mesh on a cylindrical surface to guide the placement of these sheets.

The resulting mesh was smoothed using a centroidal area smoothing algorithm on the surfaces, and a Laplacian smoothing algorithm on the hexahedral elements. Finally, the entire mesh was optimized using a mean-ratio metric to obtain the mesh quality distribution shown in Figure 6.27.

The sheet insertion process in this example creates highly skewed elements in the front filleted area due to the near tangency of the two fundamental sheets used to capture the filleted geometric feature. The skewed elements are the 27 questionable elements shown in the mesh quality distribution in Figure 6.27, and similar in shape to doublet elements (described in Chapter 2). It is possible to improve the quality of these elements using pillowing, as described by Mitchell, et al. [67].

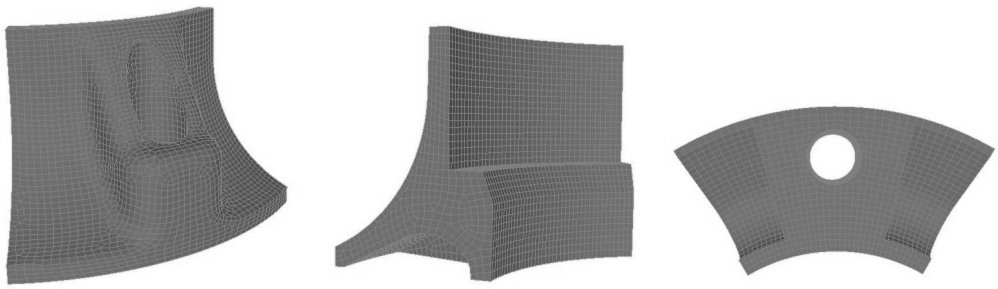

Figure 6.25. Hexahedral mesh of the part29 model. Images of the mesh from the front, back and bottom are shown, respectively.

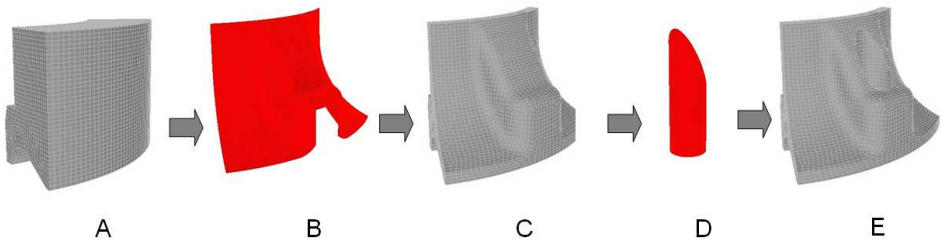

Figure 6.26. Pictorial flow chart demonstrating the mesh generation process for creating the hexahedral mesh of part29. Triangle meshes (red) are utilized to guide placement of hexahedral sheets into the existing hexahedral mesh to achieve a new mesh that is conformal with the original solid geometry.

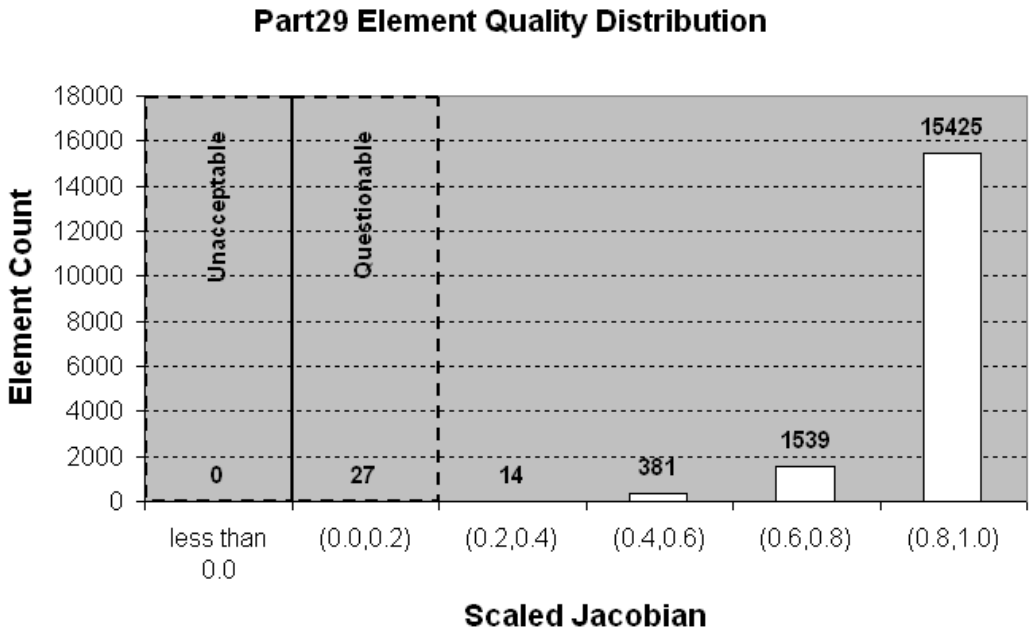

Figure 6.27. Distribution of element quality for the part29 model.

6.2.8 Goose16 Model

The goose16 model is provided courtesy of ANSYS [2]. 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 prohibits common sweeping and blocking 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 Figure 6.28, was generated using algorithms in both SCIRun and CUBIT, and contains 57,114 hexahedra. The mesh quality distribution of scaled Jacobian measures for the goose16 mesh is shown in Figure 6.29.

The mesh was generated following the process flow shown in Figure 6.30. 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 Figure 6.30.

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 (refer to Chapter 4) 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 Figure 6.29.

Figure 6.28. Hexahedral mesh of the goose16 model showing images from the front and back of the mesh, respectively.

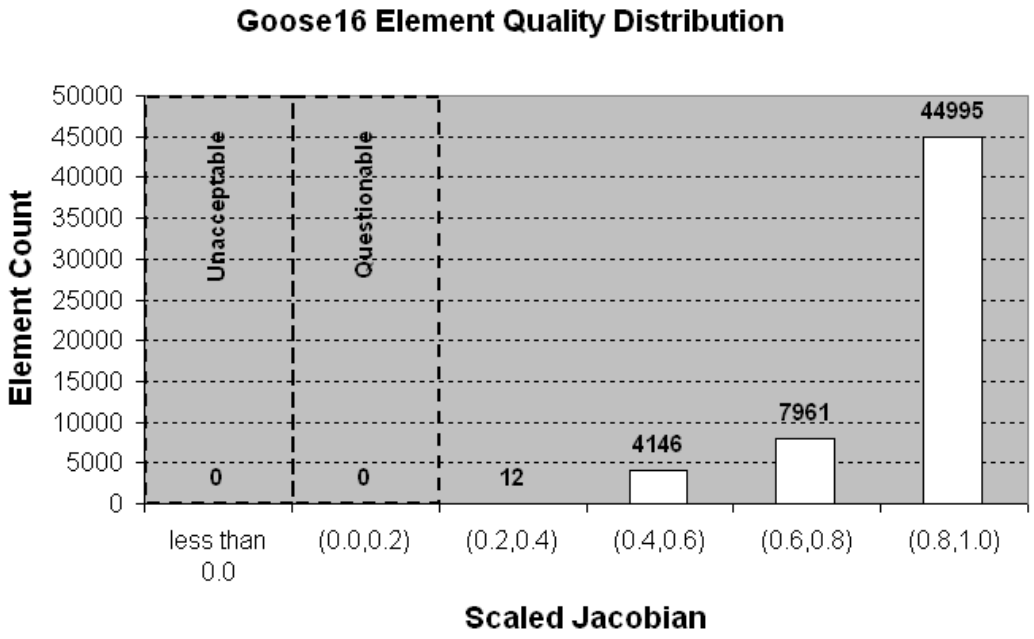

Figure 6.29. Distribution of element quality for the goose16 model.

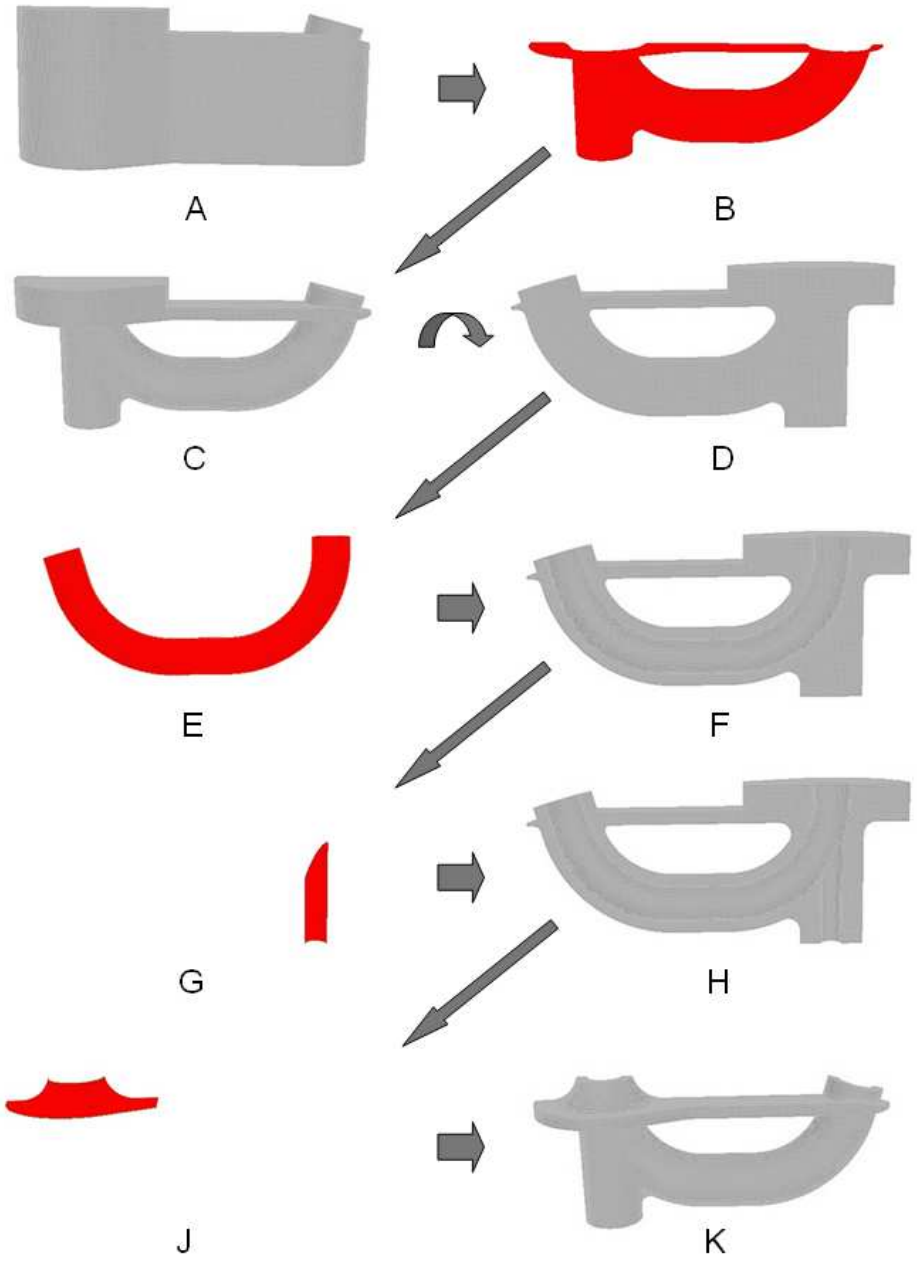

Figure 6.30. Pictorial flow chart demonstrating the mesh generation process for creating the hexahedral mesh of the goose16 model. Triangle meshes (red) are utilized to guide placement of hexahedral sheets into the existing hexahedral mesh to achieve a new mesh that is conformal with the original solid geometry.

6.2.9 Test1 Model

The Test1 model is provided courtesy of ANSYS [2]. The complexity of the geometry in this model makes it nearly impossible to create a hexahedral mesh using traditional methods of hexahedral mesh generation.

The hexahedral mesh of the test1 model, shown in Figure 6.31, was generated using algorithms in both SCIRun and CUBIT, and contains 158,372 hexahedra with a mesh quality distribution of scaled Jacobian metrics as shown in Figure 6.32.

The mesh was generated following the process flow shown in Figure 6.33. Specifically, a bounding box matching the original dimensions of the test1 part was created and a regular grid was placed in the bounding box. A set of sheets was created and inserted into this mesh to capture the general features of the internal geometry of the test1 model. This was first attempted in CUBIT, but the solid-modeling Boolean operation on the geometry failed and distorted the inserted sheet of elements making the resulting mesh unusable. The same sheet configuration was then tried in SCIRun without the benefit of the solid-modeling operation occurring on the geometry. This operation in SCIRun resulted in a correct mesh topology, but all of the elements at the boundary were rounded without the correct curve definitions to map the resulting string of mesh edges. The mesh was then exported and reloaded into CUBIT using the solid-modeling creation algorithms [78] to try and recover the hard curves where possible.

Several additional sheet insertions were then performed to create the major holes on both sides of the model, as well as all of the mounting holes in the original geometry. The resulting geometry created from the hexahedral facets is shown in Figure 6.34. In this image, it is possible to distinguish the hard curves from the final set of sheet insertions compared to the rounded-looking curves that resulted from the initial failed solid-modeling Boolean operation.

This final mesh was optimized using an optimization algorithm focused on the mean-ratio metric for the hexahedral elements. The resulting mesh quality distribution of scaled Jacobian metrics is shown in Figure 6.32. The final mesh has 307 hexahedral elements with negative Jacobians. Figure 6.35 shows the location of these elements that are mainly found in the regions where the elements are have become rounded due to the Boolean failure described earlier.

Figure 6.31. Hexahedral mesh of the test1 model.

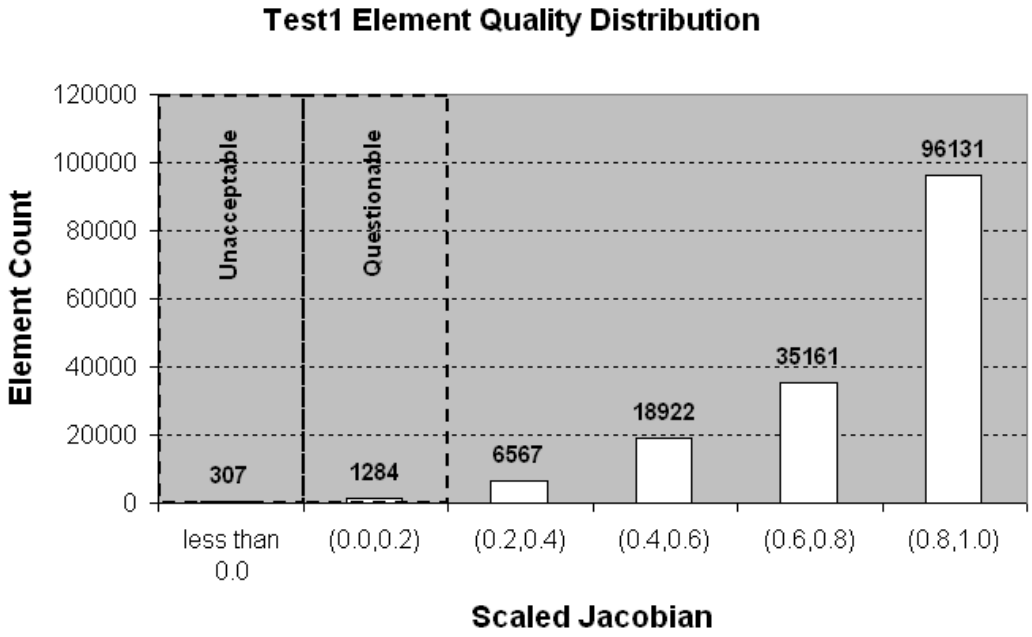

Figure 6.32. Distribution of element quality for the Test1 model.

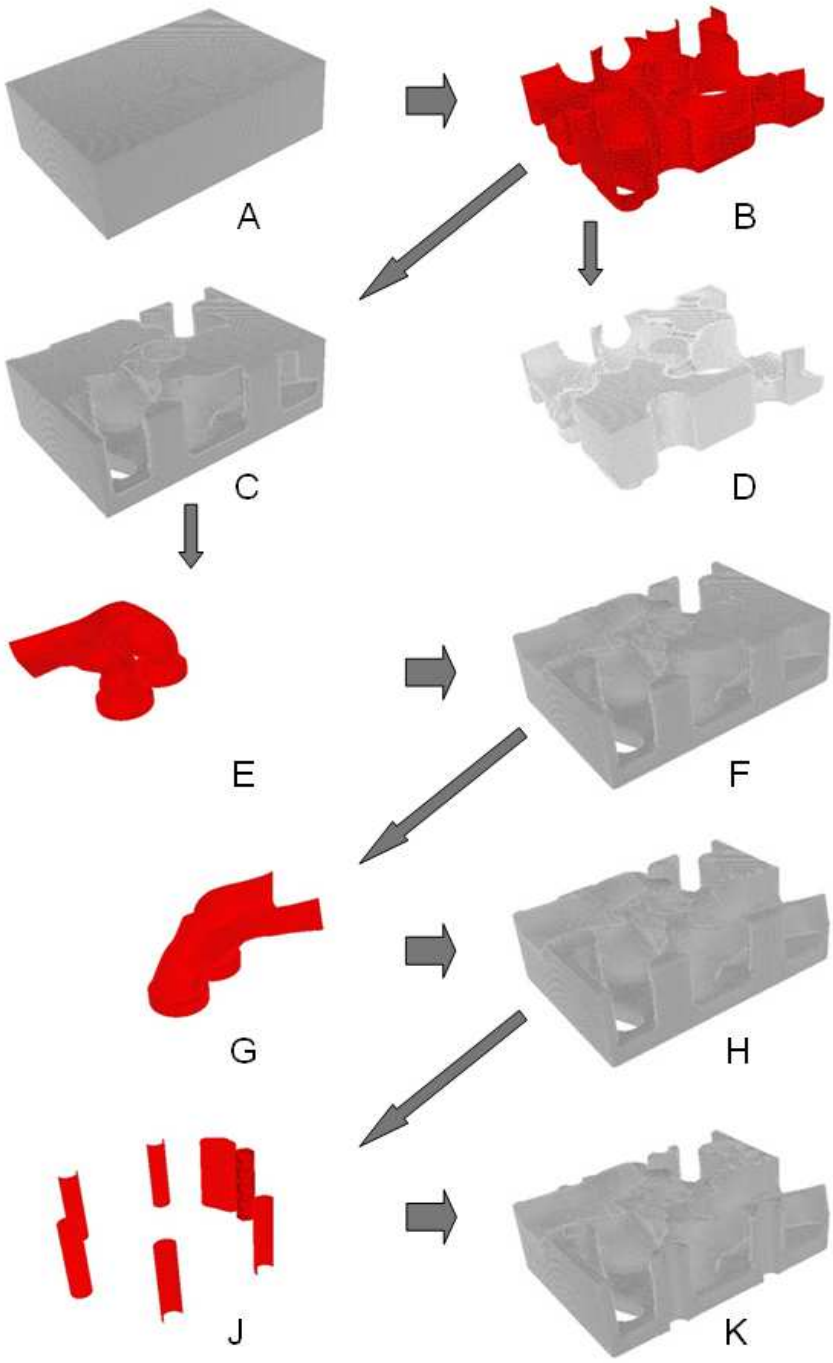

Figure 6.33. Pictorial flow chart demonstrating the mesh generation process for creating the hexahedral mesh of the test1 model. Triangle meshes (red) are utilized to guide placement of hexahedral sheets into the existing hexahedral mesh to achieve a new mesh that is conformal with the original solid geometry. The yellow mesh is the complementary hexahedral mesh produced by the previous sheet insertion.

Figure 6.34. Image of the geometry generated from the hexahedral mesh of the test1 model.

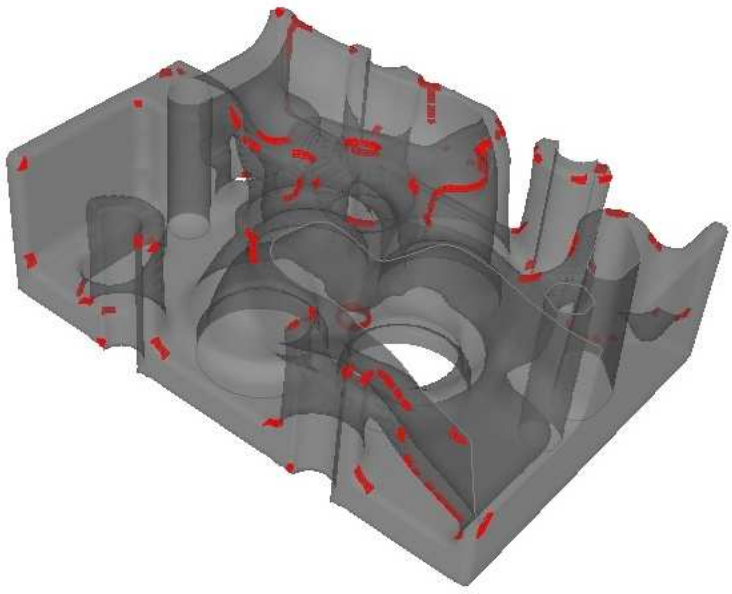

Figure 6.35. Locations of the negative Jacobian hexahedral elements in the test1 model.

CHAPTER 7

CONCLUSION

7.1 Conclusion

At the beginning of this dissertation, we defined three goals that were to be accomplished by this research. We conclude by summarizing these three goals, and discussing how these goals are satisfied by this research. Specifically, these goals were:

1. Define the necessary constraints for producing quality hexahedral meshes in arbitrary solids.

2. Define the basic geometric and topologic structures needed to create a hexahedral mesh in an arbitrary solid, and demonstrate minimization of hexahedral meshes by manipulating these structures.

3. Demonstrate hexahedral mesh generation for a collection of arbitrary solids of increasing complexity.

In Chapter 2 we developed a list of requirements that must be satisfied for a valid hexahedral mesh to be generated for a given geometric solid. These requirements were developed in both the primal and dual spaces, and the relationship between these two spaces was discussed. The requirements outlined were strict and specific in an effort to describe hexahedral meshes suitable for use in common numerical methods, such as finite element analysis, with specific quality metrics defined. These requirements also establish a guideline for comparing a hexahedral mesh generation algorithm's ability to satisfy and/or optimize these constraints in building a hexahedral mesh.

In Chapter 3, we presented a review of common hexahedral mesh generation algorithms available and currently under development. We compared these algorithms according to the constraints detailed in Chapter 2 and indicated the algorithm's strengths and weaknesses in satisfying these constraints. We also outlined several mesh modification,

smoothing, and optimization methods that have been developed to augment existing algorithms in producing more suitable quality and/or topology within a previously created mesh.

In Chapter 4, we defined the concept of a 'fundamental' hexahedral mesh by borrowing concepts from wave theory and minimizing the elements needed to generate a hexahedral mesh as defined by the hexahedral mesh generation constraints outlined in Chapter 2. We demonstrated several operations to convert a nonfundamental hexahedral mesh into a fundamental mesh, and demonstrated how these operations can be utilized to improve hexahedral element quality an existing mesh. We showed that for a given geometric solid there may be several topologies that satisfy the requirements for being a fundamental hexahedral mesh, and asserted that there exists a set of operations that could be utilized to convert one fundamental mesh to another fundamental mesh. We also asserted that the minimal mesh in a geometric solid is contained in the set of fundamental meshes for that solid. 'Secondary sheets' were defined and we showed that these sheets can be removed to minimize the hexahedral mesh while retaining the mesh topology needed to remain conformal with the topology of the geometric solid.

In Chapter 5, we reviewed techniques for generate isosurfaces from volumetric data. Building on these techniques, we developed enhancements to an existing algorithm that generated quadrilateral isosurfaces, as well as a corresponding hexahedral topology, and demonstrated dramatic improvements to the overall hexahedral element quality produced by this method. With these enhancements, this method can now produce hexahedral meshes suitable for computational analysis, while maintaining high topologic regularity within the hexahedral topology.

Additionally, in Chapter 5, we also demonstrated a method for inserting hexahedral sheets into existing, arbitrary, hexahedral topologies utilizing a triangle mesh on a manifold surface to guide the placement of the new hexahedral sheets. This method was implemented in SCIRun and was utilized to create hexahedral meshes of increasingly complex geometries defined by a single boundary surface. The meshes produced were generally of exceptionally high quality, suitable for computational analysis.

In Chapter 6, we built on ideas developed in Chapters 2 and 4 to demonstrate the utilization of two intersecting hexahedral sheets to generate the mesh topology needed for capturing geometric curves. Using the sheet insertion techniques developed in Chapter 6, we demonstrated creation of mesh topologies that match geometric topologies for

volumes with multiple surface boundaries. We demonstrated hexahedral mesh generation for several geometric objects of increasing complexity, producing hexahedral meshes with element quality readily suitable for computational analysis.

The multisurface meshes shown in Chapter 6 started with a simplified geometric boundary that was amenable to meshing using one of the structured or semistructured schemes as discussed in Chapter 3. With the additional sheets added to these mesh topologies, we can improve the algorithmic categorizations, as shown in Table 7.1.

In Chapters 5 and 6, the algorithmic modifications focused on improving the structured and semistructured algorithms by improving their ability to satisfy the boundary constraints for hexahedral mesh generation. Other algorithmic paradigms could also be modified to improve specific weaknesses in hexahedral mesh constraint satisfaction. Some possible areas of future research will be addressed in the next section.

7.2 Future Research

While the techniques and research described above are extremely useful in generating and improving hexahedral meshes, this research also raises many areas of possible

Table 7.1. Satisfaction of the hexahedral mesh generation constraints using a structured or semistructured algorithm augmented by sheet insertion to capture additional boundary constraints.

Constraint Class	Description
Topologic	**Strong**-Structured and semistructured schemes strongly satisfy the topologic requirements for hexahedral mesh generation. Additional sheets are inserted into this mesh will still satisfy the topologic constraints for hexahedral meshes.
Boundary	**Strong**-Boundary constraints which are not met satisfactorily by the original structured or semistructured mesh topology can be resolved by the introduction of the fundamental sheets necessary to completely satisfy the boundary constraints.
Quality	**Moderate**-Mesh quality is typically high throughout the majority of the volume, with some possible quality degradation (due to sheet curvature or reduced regularity) occurring near the boundaries.

future research to improve the techniques described and to answer additional important questions. The remainder of this dissertation will focus on a few of these questions.

7.2.1 Hexahedral Mesh Generation Constraints

The hexahedral mesh generation constraints have become the major backbone of this research. In the course of preparing these lists, several questions were formulated that still do not seem to have a sufficient solution. Some of these questions will be outlined in this section.

7.2.1.1 Linear programming. In Chapter 2 we drew an analogy from linear programming to hexahedral mesh generation, and outlined a list of constraints that need to be satisfied for a hexahedral mesh to be deemed an adequate alternative geometric representation to an original geometric solid model definition. In essence, a hexahedra is a complex polytope that can also be represented by a set of equations, $Ax \leq b$, where A (for a hexahedron) is a 6x3 matrix of the half-spaces defining the planes of the hexahedra and b is a vector defining the bounds of the half-spaces. In optimization theory, a set of inequalities of the form $Ax \leq b$ can be posed as a linear programming problem.

The biggest hurdle to converting hexahedral mesh generation to a linear program is that hexahedral meshing involves creation of new elements in a space where no elements currently exist. This implies that the linear program would need to be dynamic with the set of constraints changing with each new element (or sheet) that is added. Developing a dynamic linear program to allow spontaneous creation (or deletion) of new mesh elements, or additions to the original set of constraints) would be an area of future research.

7.2.1.2 Global untangling. The original whisker weaving [101, 29] research and development ended with an algorithm that was guaranteed to generate a hexahedral mesh topology given some flexibility to modify the quadrilateral boundary to remove self-intersecting dual loops. However, the quality of these hexahedral topologies was never sufficient for subsequent use in computational analysis. It was hoped and anticipated that research in mesh untangling, mesh optimization, and local modifications to the mesh topology would remedy these problems [44]. While tremendous progress has been made in these areas, a generalized algorithm like Whisker Weaving has yet to offer sufficient automation and quality to be generally useful.

This begs the following question: Assuming all topologic constraints are satisfied within a mesh, does there always exist an untangled solution for that mesh? Current mesh smoothing and optimization algorithms, including mesh untangling algorithms, are

based on optimization algorithms for finding local extrema. Assuming that all topologic criteria for a hexahedral mesh have been satisfied, what are the conditions that prevent mesh untangling? Is it possible to prove that only tangled solutions exist for some mesh topologies and geometries using global optimization algorithms? Assuming that a mesh with a tangled solution exists, is it possible to show that the tangling is caused by boundary effects? Can the boundary effects be mitigated with additional algorithms that may change the mesh topology but improve the geometry of the sheets?

A specific case in point is the mesh developed by Suzuki, et al. [100] in solution to the open problem of Schneider's pyramid [90]. While all of the topologic constraints were satisfied in generating the mesh for this problem, an untangled solution with this topology has not been given, despite numerous untangling and optimizations of the elements. It is known that untangling is a nonconvex problem, and the question of whether, or not, this mesh can be untangled remains open.

7.2.1.3 Quality potential metrics. For a given mesh topology, the curvature and orthogonality of the sheets is constrained by the sheet's connectivity with adjacent sheets. Vachal, et al. [106] have proposed a mesh untangling algorithm utilizing a linear program to locate the feasible region for nodal placement to obtain an untangled solution for the mesh. Another potential analysis could be the comparison of the potential quality of a mesh based on the boundary of the feasible regions for each of the elements, i.e., is it possible to calculate ranges of potential element quality for a mesh given a mesh topology and a fixed boundary? Or, in other words, given two mesh topologies for a given geometry, is it possible to show that one mesh topology is likely to have higher quality after optimization than the other mesh topology?

A metric for predicting the potential quality of a given hexahedral topology might be useful in assessing the usefulness of topological changes to the mesh, such as 'flips' [6, 102], and even utilizing this metric as an objective function for controlling the placement of these modifications in a mesh. This metric may also be useful in showing why it may not be feasible to generate high quality hexahedral meshes from a given set of boundary quadrilaterals within a given solid.

7.2.2 Hexahedral Mesh Generation Algorithms

In this research, we have utilized the constraints developed in Chapter 2 and the intuition developed in Chapter 4 to augment the structured and semistructured hexahedral mesh generation algorithms in an effort to extend the class of geometries for which they

are applicable. In this augmentation, we specifically focused on the weakness of these algorithms to satisfy the boundary constraints in a hexahedral mesh for any geometry beyond the primitive shapes that they were designed to accommodate.

The next question that should be asked is: Is it possible to extend any other current hexahedral meshing algorithms to incorporate more implicitly constraint-satisfying methods to improve the quality of the mesh produced or expand the set of geometries for which they are applicable? In this section, we will discuss possible extensions to other existing algorithms that could be incorporated to enable these methods to be used more broadly in hexahedral meshing.

7.2.2.1 Clean-up of THex meshes. The major advantage of a THex, or tet-to-hex, mesh (i.e., a mesh that is created by first generating a tetrahedral mesh of the geometry and then splitting each tetrahedral element into four hexahedral elements, as outlined in Chapter 3) is that this process is highly automatable. The disadvantage of this method is the quality and topology of the meshes produced is not desirable for utilization in subsequent computational analysis.

In Chapter 4, we demonstrated the use of a face-collapse operation to alter the structure and topology of the sheets to produce fundamental meshes from nonfundamental meshes. It might also be possible to use these mesh flipping operations to locally modify the structure and topology of the spherical sheets present in THex meshes to produce flatter sheets with increased topological regularity.

In Figure 7.1 is shown a portion of a THex mesh showing the regular circular patterns of the sheets (three are shown by dotted lines) throughout the mesh. Using a series of

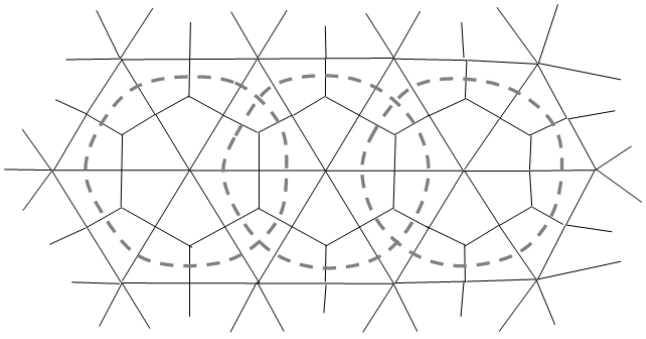

Figure 7.1. An example THex mesh showing a characteristic spherical sheet structure. Three sheets within the mesh are highlighted as dashed lines.

alternating face collapse operations and sheet extractions, as shown in Figure 7.2, we can elongate the three original sheets into the single elliptical, sheet shown in image D. While this improves the quality of the elements in the elongated sheet, the goal should be to improve the regularity and planarity of all of the sheets in the region. The sheet structure of the mesh after the face-collapse and extraction operations is shown in Image A of Figure 7.3, while a desired sheet structure is shown in Image B of Figure 7.3. The mesh resulting from the desired sheet structure is shown in Figure 7.4.

The obvious questions that arise from this example are: Can these operations be generalized for arbitrary nodal valences in a THex mesh? Can these operations be directed such that the entire mesh can be optimized rather than just localized areas within the mesh? Can this process be automated? Are there additional flip operations and/or refinement operations that would improve the performance and overall quality between steps?

7.2.2.2 Whisker weaving. An algorithm that guarantees hexahedral topology [29] is extremely powerful. It should be possible to gain an understanding of where quality constraints are not being satisfied and implement improvements to the algorithm

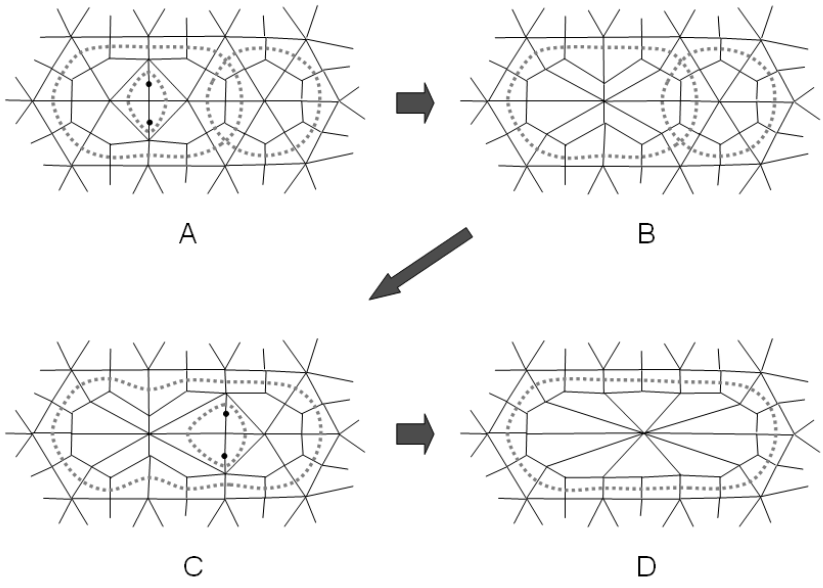

Figure 7.2. A series of face collapse and sheet extraction operations can be utilized to transform the original three spherical sheets into a single elongated elliptical sheet. The increased planarity improves the element quality and regularity of the hexahedral elements within the sheet.

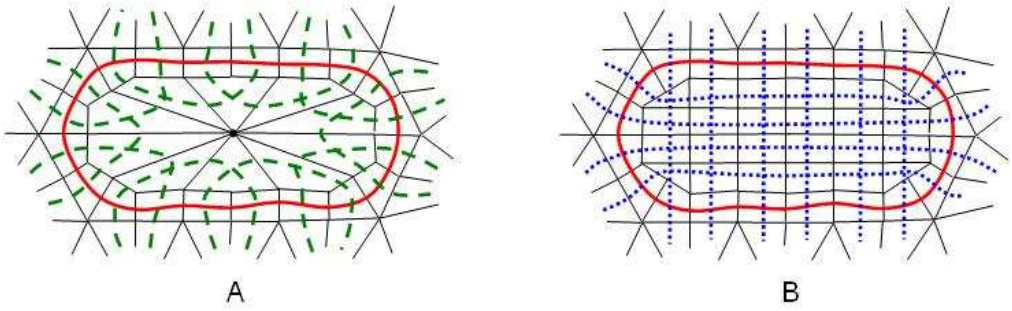

Figure 7.3. The sheet structure of the thex mesh after the face collapse and sheet extraction operations is shown on the left, while a desired sheet structure is shown on the right.

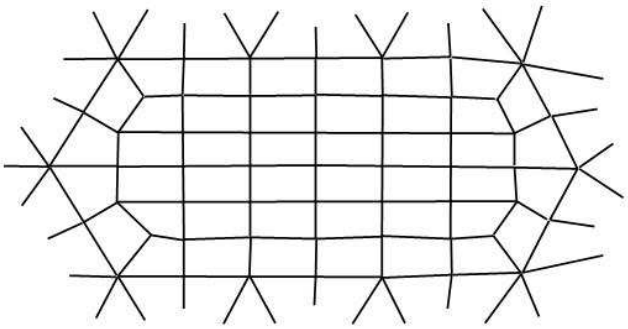

Figure 7.4. The modified THex mesh resulting from the desired sheet structure shown in Figure 7.3.

to satisfy these constraints. These efforts might include better decisions in the advancing front methodology to order preference to sheets that will have more planarity, metrics for evaluating surface meshes for potential quality problems in the final hexahedral mesh, and tools for modifying the surface meshes that improve the quality of the hexahedral mesh inferred by these surface meshes.

Additionally, currently all of the known whisker weaving algorithms [101, 29, 69, 17] remove self-intersections in the dual loops of the quadrilateral mesh on the boundary. Suzuki, et al. [100] have shown that it might be possible in many instances to generate sheet structures with self-intersections of the sheet being allowed. The addition of this intuition would remove a great deal of the initial surface modification required prior to

the algorithm proceeding and may additionally improve the quality problems that are often induced in the modifications to the surface mesh initially.

7.2.2.3 R-adaptive hexahedral mesh generation. With the sheet insertion techniques described in this research, it may be possible to develop an algorithm that utilizes an r-adaptive operator to place analytic features (i.e., shock fronts, boundary layers) into existing meshes (see Figure 7.5). The advantage of placing such features is that it would be possible to maintain an orthogonal, and highly adapted, hexahedral mesh topology to improve error estimates in an analysis utilizing these hexahedral meshes. This method might be similar to the volumetric isosurface creation techniques described in Chapter 5 using a contour dictated by the analysis representing a shock front, or similar, to place the hexahedral sheet in an r-adaptive manner to capture the analytic feature. Additionally, a dicing [61] technique could be utilized, in conjunction with r-type smoothing and a geometric bias to place several layers of elements with increasing thickness, to generate an adaptive set of layers similar to techniques currently utilized to create high-aspect ratio regions within a mesh suitable to capturing boundary effects within the simulation [34, 21].

7.2.3 Fundamental Hexahedral Meshes

The concept of a fundamental mesh for a given geometric object, or set of objects, has been useful in developing the intuition necessary to generate the hexahedral mesh examples shown in Chapters 5 and 6. Additionally, there is a great deal of research that

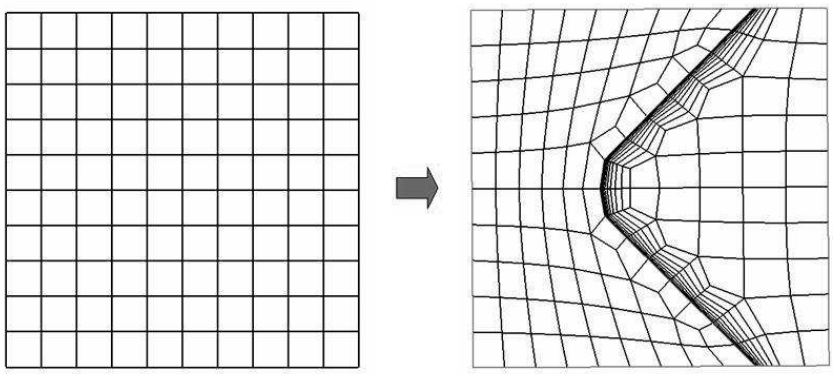

Figure 7.5. An example demonstrating sheet placement which might be utilized for r-type adaptive capture of analytic features within an existing mesh.

could still be completed in utilizing topology modifications (i.e., flips) to improve and transform existing hexahedral meshes. We will outline a couple of avenues of possible future research in this section.

7.2.3.1 Schneider's pyramid. Schneider's introduced a problem in hexahedral mesh generation [90] of a pyramid with a simple primitive quadrilateral mesh on each of the boundary surfaces of the pyramid (see Figure 7.6). The boundary of this mesh contains two chordal loops: one is basically circular, while the second is highly complex containing eight self-intersections (the complex loop is shown in Figure 7.7). Suzuki, et al. [100] created these two interior surfaces to form a solution to Schneider's open pyramid problem (see Figure 7.7). This solution contains 20 hexahedral elements, although many of these hexes have doublet faces. The solution presented by Suzuki is a fundamental mesh of a geometric pyramid where the number of sheets required to produce the mesh is minimized. Due to the complexity of the sheet, the quality is compromised as predicted by the hexahedral quality constraints outlined in Chapter 2.

Another hexahedral solution for a geometric pyramid is to divide the pyramid in half across one of the diagonals of the base and use a tet-to-hex scheme on the two resulting tetrahedra (see Figure 7.8). This solution contains 8 hexahedra, and is also a fundamental mesh of the pyramid containing seven sheets (one for each surface of the pyramid plus two additional sheets aligned with the diagonal cut that satisfy interior topologic constraints for the geometric vertex at the apex of the pyramid).

If *Assertion 2* is correct as outlined in Chapter 4, then there should exist a set of hexahedral flip operations to convert one fundamental solution for the split pyramid into the solution proposed for Schneider's pyramid by Suzuki, et al. The set of 'flips' necessary to convert the pyramid problem from one form to another is currently an open problem.

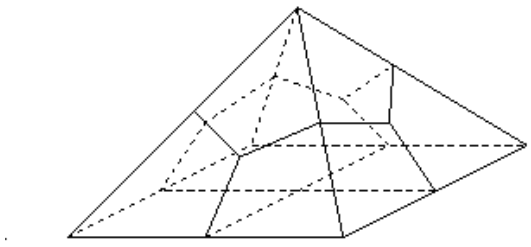

Figure 7.6. Schneider's pyramid open problem.

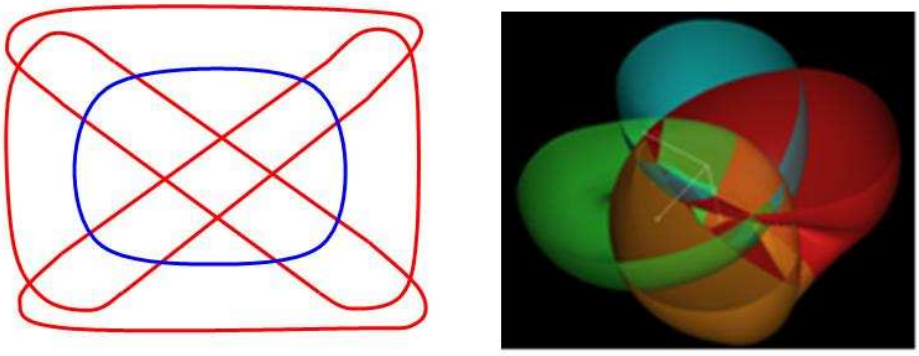

Figure 7.7. The image on the left shows the two chordal loops from the dual of the quadrilateral boundary mesh (mapped to a plane) as inferred from Schneider's pyramid. The image on the right is an interior surface generated by Suzuki, et al. [100] that fits the complex chordal loop.

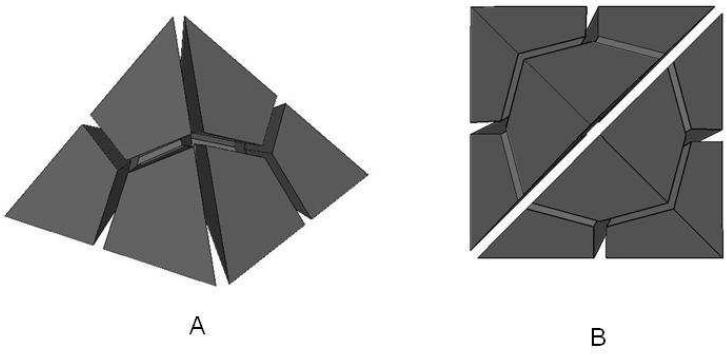

Figure 7.8. Image A is an isometric view of a hexahedral mesh of a 4-sided pyramid, created by splitting the pyramid in two down one diagonal of the pyramids base through the apex and meshing the resulting tetrahedron with a tet-to-hex primitive. Image B is a top view of the same mesh. The hexahedra have been shrunk by 15% to aid in visualization of the individual elements.

The desired set of 'flip' operations do not involve element removal, as occurs with the collapse operations outlined in Chapter 4. Rather, what is desired is the ability to create perform the topological operation and satisfy the hexahedral topological constraints without forcing an element removal. We can mimic these types of operations using a combination of collapse, sheet insertion and local pillowing operations, but a complete understanding of the necessary operations is an area of future research.

7.2.3.2 Automated decompositions. Several algorithms have been developed to automate the decomposition of a geometric object into primitive shapes that are readily meshable with hexahedral elements [12, 57, 103, 53, 80, 81, 1]. In many cases, the fundamental sheets for a geometric solid can be determined directly from the solid-model definition. If a minimal set of sheets can be determined that satisfies the topologic and boundary constraints for a given solid object, then this set of sheets can also be utilized to determine a decomposition of the solid into topological cubes that are readily meshable with hexahedra. In addition, it should be possible to utilize this set of sheets to determine regions of the volume that would be sweepable and further minimize the number of decompositions needed within the volume to realize a decomposed set of objects that are readily meshed with existing hexahedral meshing algorithms.

7.2.3.3 Generalized coarsening. In Chapter 4, we demonstrated hexahedral coarsening using a sheet extraction algorithm that preserved the mesh topology necessary for conforming to the geometric topology of the solid. *A priori* flagging of the fundamental and/or boundary sheets, in an effort to restrict removal, coupled with face collapse and other similar flip operations for modifying mesh topology, might be utilized to develop an adaptive coarsening algorithm for hexahedral meshes.

If a generalized coarsening algorithm could be developed, then this algorithm along with other generalized hexahedral refinement algorithms [104, 38, 88], would offer a tremendous capability for generating anisotropic hexahedral meshes after an initial hexahedral mesh had been developed.

7.2.3.4 Automated fundamental mesh recovery. The techniques described in Chapter 4 recovered a fundamental mesh from the set of boundary sheets by performing face collapse, and sheet insertion and extraction operations by hand. An area of possible future research involves automating this process.

Utilizing similar operations, it might also be possible to develop an algorithm that rearranges mesh topology to improve geometric quality of the sheets. Current smoothing algorithms modify the placement of nodes within a mesh in order to improve the quality of the mesh without altering the mesh topology (that is, smoothing algorithms do not change the connectivity of an element with adjacent elements). Historically, the research in mesh flipping algorithms has focused on developing comprehensive lists of possible flip operations [6, 7, 102], or performing a 'flip' operation, and then comparing the quality of the meshes before and after the operation [44]. However, in some cases it may take many

flip operations performed strategically in succession before the ultimate potential quality of the mesh can be realized. Is it possible to develop an effective objective function that could be utilized to drive these flip operations to realize dramatic improvements in overall mesh quality?

7.2.4 Hexahedral Isosurfacing

Isosurfacing techniques are becoming extremely popular as a method of model generation that bypasses the alternative solid-model generation step which can be costly and difficult. While current isosurfacing techniques continue to improve, mesh generation and clean-up from these models poses several challenges unique to this field of research. Additional improvements to the techniques outlined in this report will be necessary in order to create meshes with increasing geometric complexity and resolution. We outline two items that need to be addressed in the near future to make the solution outlined in this report a feasible mesh generation option.

7.2.4.1 Adaptive sizing. Sizing of meshes determines the level of resolution captured from the geometric data. Recent techniques for refining hexahedral meshes [104, 38, 88] would make it possible to adaptively size the hexahedra in a mesh prior to insertion of a sheet. This would enable more complex structures to be captured within the mesh, while minimizing the number of elements required in the original mesh into which a sheet is inserted.

Additionally, it is possible that recent work by Varadahn, et al. [107] for providing guarantees on topological correctness for triangle isosurface extraction could be modified to provide similar guarantees for the hexahedral meshes generated with the algorithms developed for this dissertation. Metrics and automated tools for detecting geometric feature loss and topological incongruities would prevent inconsistencies from arising due to improper mesh sizing as shown in Figure 7.9.

Topology preservation metrics coupled with adaptive octree sizing for features and conformal hexahedral refinement techniques would provide additional security to geometric fidelity preservation using the sheet insertion techniques described earlier. These techniques would also enable generation of more complex meshes which are currently limited by computer memory and efficiency issues as meshes become larger due to decreased mesh sizes to ensure proper geometric capture of features in complex models.

7.2.4.2 Convex/concave feature detection and adjustment. The technique that was outlined for inserting hexahedral sheets into existing hexahedral mesh using a

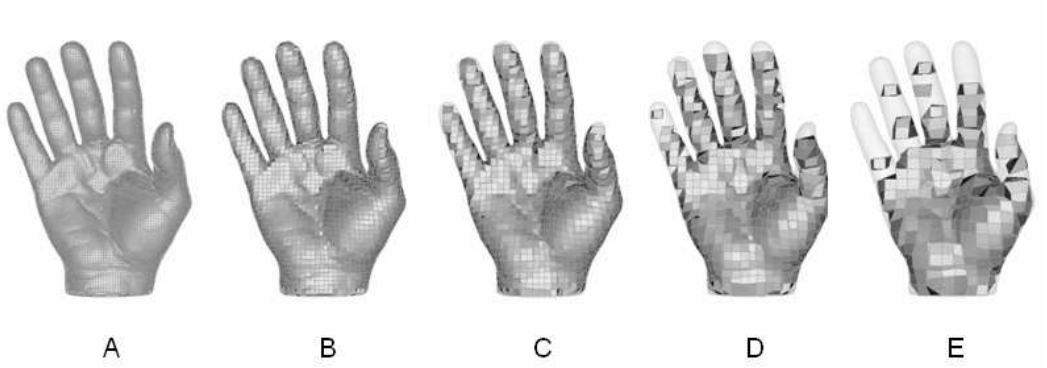

Figure 7.9. As element sizing decreases, incorrect topology may result. Topology preservation metrics, similar to those developed by Varadhan [107], coupled with adaptive hexahedral sizing could be utilized to prevent incorrect topology. Image A of the hand contains 81,922 hexahedra. Image B contains 15,670 hexahedra. Image C contains 6,703 hexahedra. Image D contains 2,745 hexahedra, and Image E contains 971 hexahedra. (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))

guiding triangle mesh on a manifold surface operates by dividing the existing hexahedral mesh into three groups of elements: 1)the hexes intersected by the triangle mesh, 2)the hexes to one side of the triangle mesh (i.e., Side1), and 3)the mesh on the opposite side of the triangle mesh (i.e., Side2). An example of this separation can be seen in Figure 7.10, where the intersected hexes are drawn in red and the two sides are drawn in green and blue, respectively. A pillowing operation is performed that shrinks two groups of hexahedral elements to create a gap between them and then generating the correct connectivity between the two groups to maintain conformal hexahedra (refer back to Chapter 3 for a more in-depth description of this process).

Before pillowing, the set of intersected hexes is grouped with either the hexes in Side1 or Side2, and the user is given the option to determine which group all of these hexes should be added. For models of high convexity it is beneficial to add the intersected hexes with the side that is interior to the surface that results in better capture of the convex features. For models with high concavity, it is typically better to place these intersected hexes with the side that is exterior to the isosurface. This schema works well so long as the volume can be classified as entirely convex, or entirely concave.

In some cases, however, making this distinction is difficult as a volume may have both highly convex and highly concave regions (the brain and the dragon models from

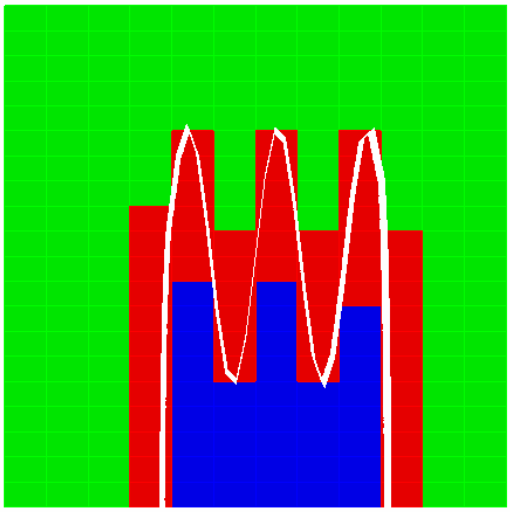

Figure 7.10. An example showing a group of hexahedra with a intersecting triangle mesh (in white). The hexes intersected by the triangles are shown in red, and the two sides (Side1 and Side2) created by the separation are shown in green and blue, respectively.

Chapter 5 are both good examples of this phenomenon). In the example shown above, the different meshes are obtained by placing the intersected elements in one side or the other. Because this example contains both highly convex and concave regions, adding the intersected hexes in total to either side results in boundaries that are not adequately homeomorphic to the original triangle mesh being inserted. This can be seen clearly in Figure 7.11, where if the intersected hexes are added to Side1 the tips of the fingers are cut off in the mesh while the indentations are resolved nicely, but if the intersected hexes are added to side2 the fingers are resolved nicely but the indentations are lost.

In these cases, it would be better to split the intersected hexahedra appropriately between the two sides resulting in a group of hexahedra that are more homeomorphic to the original triangle mesh being inserted. Referring to Figure 7.12, an alternative grouping that would result in a mesh that better resolves the original triangle mesh would be to place the green and light blue hexahedra together in one side, and place the yellow and dark blue hexahedra in the second side prior to performing the pillowing operation. This grouping results in a set of hexahedra which is more directly isomorphic to the original set of triangles which improves the resulting quality of the hexahedral mesh and the geometric fidelity to the original model.

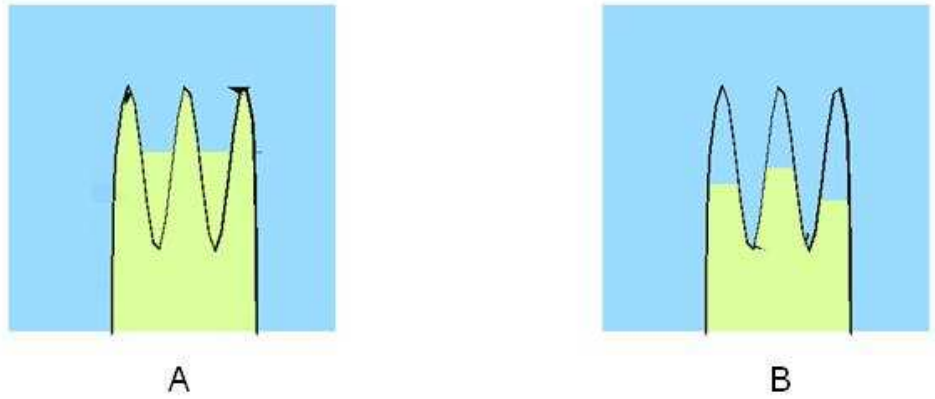

Figure 7.11. In a model with both high convex and highly concave regions, it may not be possible to add all of the intersected hexes to a single side without losing features present in the original triangle mesh. Image A shows a mesh with the intersected hexes to the elements of the light green mesh with the tips of the fingers are cut off in the mesh while the indentations are resolved nicely. Image B indicates the mesh that results when the intersected hexes are added to light blue mesh resulting in the fingers being resolved nicely but the indentations being lost.

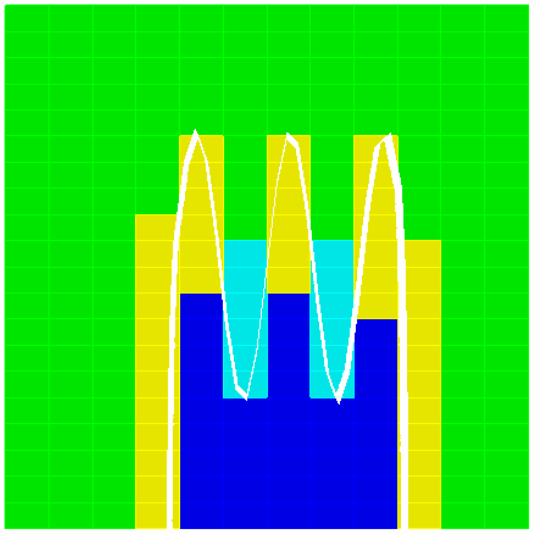

Figure 7.12. A better grouping that results in a hexahedral boundary that is more closely homeomorphic to the original triangle mesh would be to place the yellow and dark blue elements together and the green and light blue elements in a separate group.

7.2.4.3 Nonmanifold entity adjustment. A second problem encountered with placing a sheet of elements occurs when the boundary of the group of hexahedra being pillowed is nonmanifold. An example of a nonmanifold group of elements is shown in Figure 7.13 that is interior to a given triangular surface shown surrounding the hexahedra. In this figure, the boundary of the hexahedral set has a single 'pinch' point. In creating a sheet of elements using the methodology described in Chapter 5 and shown in Figure 7.14, the single node at the pinch point needs to be projected twice, rather than a single time, as is the case for the other nodes on the boundary. In the 3D case, this double projection also requires splitting the elements in the opposite side to maintain conformal topology.

Nonmanifold 'pinching' of the boundary in a group of hexes can occur at edges or at nodes. While these cases are relatively easy to detect, utilizing the methodology for projecting these elements in multiple directions and updating the opposite group can be challenging.

A couple of alternatives to the double projection method are available. The first is simply to increase the number of elements in the area by decreasing the element size in the original hexahedral mesh. The second method is simply to add (or remove) an element from the opposite side to resolve the nonmanifold circumstance in the boundary

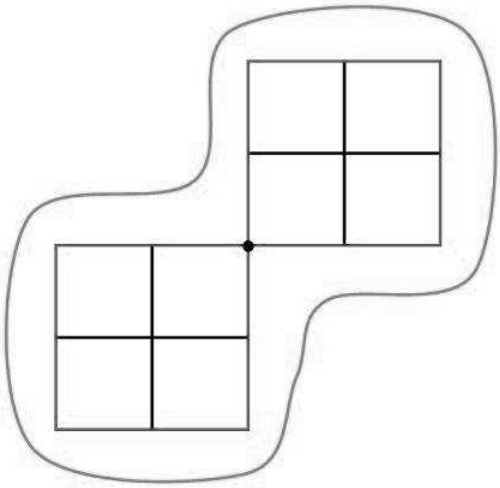

Figure 7.13. The boundary of the group of hexahedra is manifold with a single 'pinch' point at the node in the center.

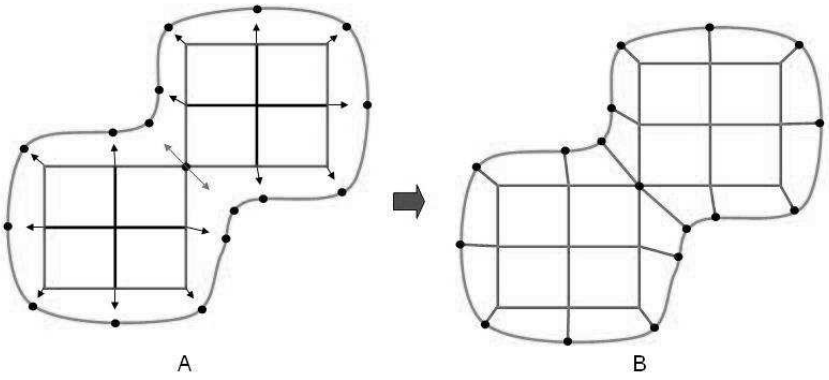

Figure 7.14. At the location of the nonmanifold node, projections must occur in two directions to properly create the sheet around the group of hexahedra, as shown in the image on the right. Alternative methods to double projection of the nonmanifold nodes avoid the nonmanifold entities and are easier to implement.

of the hexahedral group.

In the SCIRun algorithm for hexahedral sheet insertion, we have utilized an iterative method of detecting the nonmanifold edges and adding a hexahedral element to the group where the nonmanifold edge is detected by removing an element from the opposite side. We did not anticipate the case of nonmanifold nodes occurring in this algorithm, and assumed that appropriately sizing the original hexahedral mesh would prevent the case of nonmanifold nodes from occurring. However, in the case of the brain model, shown in Chapter 5, we encountered 150 cases of nonmanifold nodes due in part to the complexity of the triangle mesh of the surface. While it should be relatively straight-forward to detect and resolve these additional cases, this is still an area of future work.

In CUBIT, we partially utilized an existing pillowing algorithm to insert the hexahedral sheets into the mesh after separating the hexahedra into appropriate groups with the triangle mesh. However, while it appears that there are several cases where the hexahedra along the boundary of the two groups are exchanged to avoid some of the nonmanifold cases, there may still be several cases that are left unhandled. We believe that some of these unhandled cases may be the cause of some of the subsequent solid-modeling Boolean failures exhibited by the CUBIT version of the algorithm. A reimplementation of the pillowing algorithm in CUBIT utilizing the version developed for SCIRun might resolve some of these problems.

7.2.5 Multisurface Hexahedral Meshing

Traditional hexahedral mesh generation has been highly coupled with solid modeling packages to enable geometric decomposition and primitive recognition needed to generate the hexahedral meshes with the algorithms commonly available. The solution presented in this research deviates slightly from traditional approaches of bottoms-up mesh and geometry creation, and in some cases couples the mesh and geometry creation in a single process. Due to this deviation, some of the operations we are trying to perform can be taxing to the solid modeling engine. Addressing these problems is needed for these techniques to be viable alternatives to generating hexahedral meshes.

7.2.5.1 Robust combined sheet insertion and solid-model Boolean operations. One of the most difficult problems in creating the models shown in Chapter 6 is simultaneously creating and mapping the newly introduced mesh topology to the appropriate geometric domain. Creating the hexahedral topology is relatively straightforward using the algorithms described previously. However, to create the new solid model we often utilized this new topology in creating the new geometric features in the original solid-model. This can be problematic, especially since the insertion step requires a shrinking of the elements often resulting in inversions.

A more robust alternative would be to create the new geometric features utilizing the given triangle surface and then mapping the elements from the newly created sheet to the appropriate features introduced by the solid-modeling operation (i.e., appropriately assigning the nodes of the new sheet to the newly created surfaces, curves or vertices in the solid model. While this is mainly a development effort, it is still an area of ongoing activity.

7.2.5.2 Multivolume cuts and geometry capture. In CUBIT, the insertion of sheets is currently limited to a single volume at a time. This is necessary to adhere to a strict database requirement in ensuring that an entity containing a mesh is never left in an invalid state. This requirement is slowly changing as sheet-based algorithms are developed, including refinement and pillowing operations. In order to generate some hexahedral meshes, it will become necessary to insert sheets and perform solid-model Boolean operations across several volumes simultaneously. While there do not appear to be any road blocks to enabling multivolume insertions to occur, there is an increased complexity in the management of the geometric and mesh databases associated with the development of this functionality. This development will be a topic of future efforts.

REFERENCES

[1] P. Y. Ang and C. G. Armstrong. Adaptive curvature-sensitive meshing of the medial axis. In *Proceedings, 10th International Meshing Roundtable*, pages 155–165. Sandia National Laboratories, 2001.

[2] ANSYS. ANSYS, http://www.ansys.com, January 2007.

[3] ANSYS. ANSYS ICEM CFD, http://www.ansys.com/products/icemcfd.asp, October 2006.

[4] C. L. Bajaj, V. Pascucci, and D. R. Schikore. Fast isocontouring for improved interactivity. In *1996 Volume Visualization Symposium*, pages 39–46, 1996.

[5] S. E. Benzley, E. Perry, K. Merkley, and B. Clark. A comparison of all hexagonal and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. In *Proceedings, 4th International Meshing Roundtable*, pages 179–191. Sandia National Laboratories, October 1995.

[6] M. Bern and D. Eppstein. Flipping cubical meshes. In *Proceedings, 10th International Meshing Roundtable*, pages 19–29. Sandia National Laboratories, October 2001.

[7] M. Bern, D. Eppstein, and J. Erickson. Flipping cubical meshes. *Engineering with Computers*, 18(3):173–187, 2002.

[8] T. D. Blacker. Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32:811–847, 1991.

[9] T. D. Blacker. The cooper tool. In *Proceedings, 5th International Meshing Roundtable*, pages 13–30. Sandia National Laboratories, October 1996.

[10] T. D. Blacker. Meeting the challenge for automated conformal hexahedral meshing. In *Proceedings, 9th International Meshing Roundtable*, pages 11–19. Sandia National Laboratories, October 2000.

[11] T. D. Blacker and R. J. Meyers. Seams and wedges in plastering: A 3D hexahedral mesh generation algorithm. *Engineering With Computers*, 2(9):83–93, 1993.

[12] T. D. Blacker, J. L. Mitchiner, L. R. Phillips, and Y. Lin. Knowledge system approach to automated two-dimensional quadrilateral mesh generation. *Computers in Engineering*, 3:153–162, 1988.

[13] M. J. Borden, S. E. Benzley, and J. F. Shepherd. Coarsening and sheet extraction for all-hexahedral meshes. In *Proceedings, 11th International Meshing Roundtable*, pages 147–152. Sandia National Laboratories, September 2002.

[14] M. J. Borden, J. F. Shepherd, and S. E. Benzley. Mesh cutting: Fitting simple all-hexahedral meshes to complex geometries. In *Proceedings, 8th International Society of Grid Generation Conference*, 2002.

[15] M. Brewer, L. Freitag-Diachin, P. Knupp, T. Leurent, and D. J. Melander. The MESQUITE mesh quality improvement toolkit. In *Proceedings, 12th International Meshing Roundtable*, pages 239–250. Sandia National Laboratories, September 2003.

[16] M. L. Bussler and A. Ramesh. The eight-node hexahedral elements in FEA of part designs. *Foundry Management and Technology*, pages 26–28, November 1993.

[17] N. A. Calvo and S. R. Idelsohn. All-hexahedral element meshing: Generation of the dual mesh by recurrent subdivision. *Computer Methods in Applied Mechanics and Engineering*, 182:371–378, 2000.

[18] C. D. Carbonera and J. F. Shepherd. A constructive approach to constrained hexahedral mesh generation. In *Proceedings, 15th International Meshing Roundtable*, pages 435–452. Sandia National Laboratories, September 2006.

[19] A. O. Cifuentes and A. Kalbag. A performance study of tetrahedral and hexahedral elements in 3-D finite element structural analysis. *Finite Elements in Analysis and Design*, 12(3-4):313–318, 1992.

[20] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Optimal isosurface extraction from irregular volume data. *1996 Volume Visualization Symposium*, pages 31–38, 1996. ISBN 0-89791-741-3.

[21] J. C. Clements, T. D. Blacker, and S. E. Benzley. Automated mesh generation in boundary layer regions using special elements. *AMD - Trends in Unstructured Mesh Generation*, pages 137–143, July 1997.

[22] W. A. Cook and W. R. Oakes. Mapping methods for generating three-dimensional meshes. *Computers In Mechanical Engineering*, CIME Research Supplement:67–72, August 1982.

[23] The CUBIT Geometry and Mesh Generation Toolkit, Sandia National Laboratories, http://cubit.sandia.gov/, 2007.

[24] J. F. Dannenhoffer(III). A block-structuring technique for general geometries. In *29th Aerospace Sciences Meeting and Exhibit*, volume AIAA-91-0145, January 1991.

[25] G. Dhatt and G. Touzot. *The Finite Element Method Displayed*. John Wiley and Sons, 1984.

[26] L. F. Diachin, P. Knupp, T. Munson, and S. Shontz. A comparison of inexact Newton and coordinate descent mesh optimization techniques. In *Proceedings, 13th International Meshing Roundtable*, pages 243–254. Sandia National Laboratories, September 2004.

[27] D. Eppstein. Linear complexity hexahedral mesh generation. In *12th ACM Symposium on Computational Geometry*, pages 58–67. ACM, 1996.

[28] D. A. Field. Laplacian smoothing and Delaunay triangulation. *Communications in Applied Numerical Methods*, 4:709–712, 1988.

[29] N. T. Folwell and S. A. Mitchell. Reliable whisker weaving via curve contraction. *Engineering With Computers*, 15:292–302, 1999.

[30] L. Freitag. On combining Laplacian and optimization-based mesh smoothing techniques. *AMD Trends in Unstructured Mesh Generation, ASME*, 220:37–43, 1997.

[31] L. A. Freitag and P. Plassmann. Local optimization-based simplicial mesh untangling and improvement. *International Journal for Numerical Methods in Engineering*, 49(1):109–125, September 10-20, 2000.

[32] I. Fujishiro, Y. Maeda, H. Sato, and Y. Takeshima. Volumetric data exploration using interval volume. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):144–155, 1996.

[33] J. Galtier, F. Hurtado, M. Noy, S. Perennes, and J. Urrutia. Simultaneous edge flipping in triangulations. *International Journal of Computational Geometry and Applications*, 13(2):113–133, 2003.

[34] R. V. Garimella and M. S. Shephard. Boundary layer mesh generation for viscous flow simulations. *International Journal for Numerical Methods in Engineering*, 49(1):193–218, 2000.

[35] S. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *1998 Volume Visualization Symposium*, pages 23–30, 1998.

[36] B. Grover, J. F. Shepherd, and S. E. Benzley. Quadrilateral mesh generation and modification for surfaces by dual creation and manipulation. In *Proceedings, 8th International Society of Grid Generation Conference*, 2002.

[37] P. Hansbo. Generalized Laplacian smoothing of unstructured grids. *Communications in Numerical Methods in Engineering*, 11:455–464, 1995.

[38] N. Harris, S. E. Benzley, and S. J. Owen. Conformal refinement of all-hexahedral meshes based on multiple twist plane insertion. In *Proceedings, 13th International Meshing Roundtable*, pages 157–168. Sandia National Laboratories, September 2004.

[39] F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. *Discrete Computational Geometry*, 22(3):333–346, 1999.

[40] S. R. Jankovich, S. E. Benzley, J. F. Shepherd, and S. A. Mitchell. The graft tool: An all-hexahedral transition algorithm for creating multi-directional swept volume mesh. In *Proceedings, 8th International Meshing Roundtable*, pages 387–392. Sandia National Laboratories, October 1999.

[41] T. R. Jones, F. Durand, and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics*, 22(3):943–949, 2003.

[42] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *ACM Transactions on Graphics*, 21(3):339–346, 2002.

[43] P. Knupp. Next-generation sweep tool: A method for generating all-hex meshes on two-and-one-half dimensional geometries. In *Proceedings, 7th International Meshing Roundtable*, pages 505–513. Sandia National Laboratories, October 1998.

[44] P. Knupp and S. A. Mitchell. Integration of mesh optimization with 3D all-hex mesh generation, LDRD subcase 3504340000, final report. SAND 99-2852, October 1999.

[45] P. M. Knupp. Winslow smoothing on two-dimensional unstructured meshes. In *Proceedings, 7th International Meshing Roundtable*, pages 449–457. Sandia National Laboratories, 1998.

[46] P. M. Knupp. Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities: Part II - a framework for volume mesh optimization and the condition number of the Jacobian matrix. *International Journal for Numerical Methods in Engineering*, 48:1165–1185, 2000.

[47] P. M. Knupp. Algebraic mesh quality metrics. *SIAM J. Sci. Comput.*, 23(1):193–218, 2001.

[48] P. M. Knupp. Hexahedral and tetrahedral mesh shape optimization. *International Journal for Numerical Methods in Engineering*, 58(1):319–332, 2003.

[49] P. M. Knupp. Hexahedral mesh untangling and algebraic mesh quality metrics. In *Proceedings, 9th International Meshing Roundtable*, pages 173–183. Sandia National Laboratories, October 2000.

[50] J. Kraftcheck. *Virtual Geometry: A Mechanism for Modification of CAD Model Topology for Improved Meshability.* Published Master's Thesis, Department of Mechanical Engineering, University of Wisconsin, December 2000.

[51] M. Lai, S. E. Benzley, G. D. Sjaardema, and T. J. Tautges. A multiple source and target sweeping method for generating all-hexahedral finite element meshes. In *Proceedings, 5th International Meshing Roundtable*, pages 217–228. Sandia National Laboratories, October 1996.

[52] M. Lai, S. E. Benzley, and D. R. White. Automated hexahedral mesh generation by generalized multiple source to multiple target sweeping. *International Journal for Numerical Methods in Engineering*, 49(1):261–275, September 2000.

[53] T. S. Li, R. M. McKeag, and C. G. Armstrong. Hexahedral meshing using midpoint subdivision and integer programming. *Computer Methods in Applied Mechanics and Engineering*, 124:171–193, 1995.

[54] Y. Livnat, H.-W. Shen, and C. R. Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84, 1996. ISSN 1077-2626.

[55] W. E. Lorenson and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (Proceedings of SIGGRAPH '87)*, 21(4):163–169, 1987.

[56] M. Loriot. TetMesh-GHS3D v3.1 the fast, reliable, high quality tetrahedral mesh generator and optimiser, http://www.simulog.fr/mesh/tetmesh3p1d-wp.pdf, 2006.

[57] Y. Lu, R. Gadh, and T. J. Tautges. Volume decomposition and feature recognition for hexahedral mesh generation. In *Proceedings, 8th International Meshing Roundtable*, pages 269–280. Sandia National Laboratories, October 1999.

[58] S. Means, A. J. Smith, J. F. Shepherd, J. Shadid, J. Fowler, R. Wojcikiewicz, T. Mazel, G. D. Smith, and B. S. Wilson. Reaction modeling of calcium dynamics with realistic ER geometry. *Biophysical Journal*, 91(2):537–557, 15 July 2006.

[59] D. L. Meier. Multidimensional astrophysical structural and dynamical analysis. I. development of a nonlinear finite element approach. *Astrophys. J.*, 518:788–813, 1999.

[60] D. J. Melander. *Generation of Multi-Million Element Meshes for Solid Model-Based Geometries: The Dicer Algorithm.* Published Master's Thesis, Brigham Young University, April 1997.

[61] D. J. Melander, T. J. Tautges, and S. E. Benzley. Generation of multi-million element meshes for solid model-based geometries: The dicer algorithm. *AMD - Trends in Unstructured Mesh Generation*, 220:131–135, July 1997.

[62] MESQUITE: The Mesh Quality Improvement Toolkit, Terascale Simulation Tools and Technology Center (TSTT), http://www.tstt-scidac.org/research/mesquite.html, 2005.

[63] K. A. Milton. Finite-element quantum field theory. In *Proceedings of the XIVth International Symposium on Lattice Field Theory*, volume Nucl. Phys. B(Proc. Suppl.) 53 (1997), pages 847–849, 1996.

[64] S. A. Mitchell. A characterization of the quadrilateral meshes of a surface which admit a compatible hexahedral mesh of the enclosed volumes. In *13th Annual Symposium on Theoretical Aspects of Computer Science*, volume Lecture Notes in Computer Science: 1046, pages 465–476, 1996.

[65] S. A. Mitchell. Choosing corners of rectangles for mapped meshing. In *13th Annual Symposium on Computational Geometry*, pages 87–93. ACM Press, June 1997.

[66] S. A. Mitchell. High fidelity interval assignment. In *Proceedings, 6th International Meshing Roundtable*, pages 33–44. Sandia National Laboratories, October 1997.

[67] S. A. Mitchell and T. J. Tautges. Pillowing doublets: Refining a mesh to ensure that faces share at most one edge. In *Proceedings, 4th International Meshing Roundtable*, pages 231–240. Sandia National Laboratories, October 1995.

[68] K. Miyoshi and T. D. Blacker. Hexahedral mesh generation using multi-axis cooper algorithm. In *Proceedings, 9th International Meshing Roundtable*, pages 89–97. Sandia National Laboratories, October 2000.

[69] M. Muller-Hannemann. Hexahedral mesh generation by successive dual cycle elimination. In *Proceedings, 7th International Meshing Roundtable*, pages 365–378. Sandia National Laboratories, October 1998.

[70] P. J. Murdoch. *The Spatial Twist Continuum: A Dual Representation of the All Hexahedral Finite Element Mesh.* Published Doctoral Dissertation, Brigham Young University, December 1995.

[71] P. J. Murdoch and S. E. Benzley. The spatial twist continuum. In *Proceedings, 4th International Meshing Roundtable*, pages 243–251. Sandia National Laboratories, October 1995.

[72] G. M. Nielson and B. Hamann. The asymptotic decider: Removing the ambiguity in marching cubes. In *IEEE Visualization*, pages 83–91, 1991.

[73] G. M. Nielson and J. Sung. Interval volume tetrahedrization. In *IEEE Visualization*, pages 221–228, 1997.

[74] The open problems project - problem 27: Hexahedral meshing, http://maven.smith.edu/ orourke/TOPP/P27.html#Problem.27, 2006.

[75] S. J. Owen. A survey of unstructured mesh generation technology, http://www.andrew.cmu.edu/user/sowen/survey/index.html, September 1998.

[76] S. J. Owen and S. Saigal. H-morph an indirect approach to advancing front hex meshing. *International Journal for Numerical Methods in Engineering*, 49(1):289–312, September 2000.

[77] S. J. Owen, M. L. Staten, S. A. Canann, , and S. Saigal. Q-morph an indirect approach to advancing front quad meshing. *International Journal for Numerical Methods in Engineering*, 9(44):1317–1340, March 1999.

[78] S. J. Owen, D. R. White, and T. J. Tautges. Facet-based surfaces for 3D mesh generation. In *Proceedings, 11th International Meshing Roundtable*, pages 297–312. Sandia National Laboratories, 2002.

[79] Pointwise. Gridgen - reliable CFD meshing, http://www.pointwise.com/gridgen/, October 2006.

[80] M. A. Price and C. G. Armstrong. Hexahedral mesh generation by medial surface subdivision: Part I. *International Journal for Numerical Methods in Engineering*, 38(19):3335–3359, 1995.

[81] M. A. Price and C. G. Armstrong. Hexahedral mesh generation by medial surface subdivision: Part II. *International Journal for Numerical Methods in Engineering*, 40:111–136, 1997.

[82] S. Richards, S. E. Benzley, J. F. Shepherd, and M. B. Stephenson. DTHexing: An improved all-hexahedral meshing scheme using general coarsening tools. SAND2003-2724A and SAND2003-2818P, 2003.

[83] X. Roca and J. Sarrate. An automatic and general least-squares projection procedure for sweep meshing. In *Proceedings, 15th International Meshing Roundtable*, pages 487–506. Sandia National Laboratories, September 2006.

[84] X. Roca, J. Sarrate, and A. Huerta. Surface mesh projection for hexahedral mesh generation by sweeping. In *Proceedings, 13th International Meshing Roundtable*, volume SAND 2004-3765C, pages 169–180. Sandia National Laboratories, September 2004.

[85] X. Roca, J. Sarrate, and A. Huerta. A new least-squares approximation of affine mappings for sweep algorithms. In *Proceedings, 14th International Meshing Roundtable*, pages 433–448. Sandia National Laboratories, September 2005.

[86] C. Scheidegger and J. Schreiner. Afront, http://sourceforge.net/projects/afront/, January 2007.

[87] R. Schneiders. A grid-based algorithm for the generation of hexahedral element meshes. *Engineering With Computers*, 12:168–177, 1996.

[88] R. Schneiders. Refining quadrilateral and hexahedral element meshes. In *5th International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 679–688. Mississippi State University, 1996.

[89] R. Schneiders. An algorithm for the generation of hexahedral element meshes based on an octree technique. In *Proceedings, 6th International Meshing Roundtable*, pages 183–194. Sandia National Laboratories, October 1997.

[90] Schneiders Pyramid Open Problem, http://www-users.informatik.rwth-aachen.de/ roberts/open.html, 2006.

[91] J. Schreiner and C. Scheidegger. Algorithm for separating hexahedra given a triangle mesh. *SCI Institute Technical Report*, UUSCI-2007, 2007.

[92] SCIRun: A Scientific Computing Problem Solving Environment. Scientific Computing and Imaging Institute (SCI), http://software.sci.utah.edu/scirun.html, 2002.

[93] M. A. Scott, M. N. Earp, S. E. Benzley, and M. B. Stephenson. Adaptive sweeping techniques. In *Proceedings, 14th International Meshing Roundtable*, pages 417–432. Sandia National Laboratories, September 2005.

[94] M. S. Shephard and M. K. Georges. Three-dimensional mesh generation by finite octree technique. *International Journal for Numerical Methods in Engineering*, 32:709–749, 1991.

[95] J. F. Shepherd, S. E. Benzley, and S. A. Mitchell. Interval assignment for volumes with holes. *International Journal for Numerical Methods in Engineering*, 49(1):277–288, September 2000.

[96] J. F. Shepherd, S. A. Mitchell, P. Knupp, and D. R. White. Methods for multisweep automation. In *Proceedings, 9th International Meshing Roundtable*, pages 77–87. Sandia National Laboratories, October 2000.

[97] M. L. Staten, S. A. Canann, and S. J. Owen. BMSWEEP: Locating interior nodes during sweeping. In *Proceedings, 7th International Meshing Roundtable*, pages 7–18. Sandia National Laboratories, October 1998.

[98] M. L. Staten, R. A. Kerr, S. J. Owen, and T. D. Blacker. Unconstrained paving and plastering: Progress update. In *Proceedings, 15th International Meshing Roundtable*, pages 469–486. Sandia National Laboratories, September 2006.

[99] M. L. Staten, S. J. Owen, and T. D. Blacker. Unconstrained paving and plastering: A new idea for all hexahedral mesh generation. In *Proceedings, 14th International Meshing Roundtable*, pages 399–416. Sandia National Laboratories, September 2005.

[100] T. Suzuki, S. Takahashi, and J. F. Shepherd. Practical interior surface generation method for all-hexahedral meshing. In *Proceedings, 14th International Meshing Roundtable*, pages 377–397. Sandia National Laboratories, September 2005.

[101] T. J. Tautges, T. D. Blacker, and S. A. Mitchell. Whisker weaving: A connectivity-based method for constructing all-hexahedral finite element meshes. *International Journal for Numerical Methods in Engineering*, 39:3327–3349, 1996.

[102] T. J. Tautges and S. Knoop. Topology modification of hexahedral meshes using atomic dual-based operations. In *Proceedings, 12th International Meshing Roundtable*, pages 415–423. Sandia National Laboratories, September 2003.

[103] T. J. Tautges, S. Liu, Y. Lu, J. Kraftcheck, and R. Gadh. Feature recognition applications in mesh generation. *AMD-Trends in Unstructured Mesh Generation*, 220:117–121, July 1997.

[104] K. Tchon, J. Dompierre, and R. Camarero. Conformal refinement of all-quadrilateral and all-hexahedral meshes according to an anisotropic metric. In *Proceedings, 11th International Meshing Roundtable*, pages 231–242. Sandia National Laboratories, September 2002.

[105] B. Thurston. Geometry in action: Hexahedral decomposition of polyhedra, http://www.ics.uci.edu/ eppstein/gina/thurston-hexahedra, October 1993.

[106] P. Vachal, R. V. Garimella, and M. J. Shashkov. Mesh untangling. LAU-UR-02-7271, T-7 Summer Report 2002.

[107] G. Varadhan, S. Krishnan, T. Sriram, and D. Manocha. Topology preserving surface extraction using adaptive subdivision. *ACM International Conference Proceeding Series; Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 71:235–244, 2004.

[108] The Verdict Mesh Verification Library, Sandia National Laboratories, http://cubit.sandia.gov/verdict.html, 2007.

[109] K. S. Walton, S. E. Benzley, and J. F. Shepherd. Sculpting: An improved inside-out scheme for all-hexahedral meshing. In *Proceedings, 11th International Meshing Roundtable*, pages 153–159. Sandia National Laboratories, September 2002.

[110] F. R. S. Weiler and R. Schneiders. Automatic geometry-adaptive generation of quadrilateral and hexahedral element meshes for FEM. In *Proceedings, 5th International Conference on Numerical Grid Generation in Computational Field Simulations*, pages 689–697. Mississippi State University, April 1996.

[111] V. I. Weingarten. The controversy over hex or tet meshing. *Machine Design*, pages 74–78, April 18, 1994.

[112] D. R. White. *Automatic Quadrilateral and Hexahedral Meshing of Pseudo-Cartesian Geometries Using Virtual Subdivision.* Published Master's Thesis, Brigham Young University, June 1996.

[113] D. R. White, M. Lai, S. E. Benzley, and G. D. Sjaardema. Automated hexahedral mesh generation by virtual decomposition. In *Proceedings, 4th International Meshing Roundtable*, pages 165–176. Sandia National Laboratories, October 1995.

[114] D. R. White, R. W. Leland, S. Saigal, and S. J. Owen. The meshing complexity of a solid: An introduction. In *Proceedings, 10th International Meshing Roundtable*, pages 373–384. Sandia National Laboratories, October 2001.

[115] D. R. White, S. Saigal, and S. J. Owen. Meshing complexity of single part CAD models. In *Proceedings, 12th International Meshing Roundtable*, pages 121–134. Sandia National Laboratories, September 2003.

[116] D. R. White and T. J. Tautges. Automatic scheme selection for toolkit hex meshing. *International Journal for Numerical Methods in Engineering*, 49(1):127–144, September 2000.

[117] M. Whitely, D. R. White, S. E. Benzley, and T. D. Blacker. Two and three-quarter dimensional meshing facilitators. *Engineering With Computers*, 12:155–167, 1996.

[118] J. Wilhelms and A. V. Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, 1992.

[119] P. Wolfenbarger, J. Jung, C. R. Dohrmann, W. R. Witkowski, M. J. Panthaki, and W. H. Gerstle. A global minimization-based, automatic quadrilateral meshing algorithm. In *Proceedings, 7th International Meshing Roundtable*, pages 87–103. Sandia National Laboratories, October 1998.

[120] XYZ Scientific Applications, Inc. Truegrid: High quality hexahedral grid and mesh generation for fluids and structures, http://www.truegrid.com, October 2006.

[121] M. A. Yerry and M. S. Shephard. Three-dimensional mesh generation by modified octree technique. *International Journal for Numerical Methods in Engineering*, 20:1965–1990, 1984.

[122] Y. Zhang and C. Bajaj. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. *Computer Methods in Applied Mechanics and Engineering (CMAME)*, 195(9-12):942–960, 2006.

[123] Y. Zhang and C. Bajaj. Adaptive and quality quadrilateral/hexahedral meshing from volumetric imaging data. In *Proceedings, 13th International Meshing Roundtable*, pages 365–376. Sandia National Laboratories, September 2005.

[124] Y. Zhang, C. Bajaj, and B.-S. Sohn. 3D finite element meshing from imaging data. *The special issue of Computer Methods in Applied Mechanics and Engineering (CMAME) on Unstructured Mesh Generation*, 194(48-49):5083–5106, 2005.

[125] Y. Zhang, C. Bajaj, and G. Xu. Surface smoothing and quality improvement of quadrilateral/hexahedral meshes with geometric flow. In *Proceedings, 14th International Meshing Roundtable*, pages 449–468. Sandia National Laboratories, September 2005.

[126] T. Zhou and K. Shimada. An angle-based approach to two-dimensional mesh smoothing. In *Proceedings, 9th International Meshing Roundtable*, pages 373–384. Sandia National Laboratories, 2000.