Unstructured Three-Dimensional Mesh Generation with Controlled Anisotropy and Directionality

Soji Yamakawa

A Thesis Submitted to

The Department of Mechanical Engineering

in Partial Fulfillment of Requirements

for the Degree of Doctor of Philosophy

Carnegie Mellon University

Pittsburgh, PA

May 2002

ABSTRACT

A new computational method for automatic three-dimensional mesh generation is presented. The method creates a high quality mesh and can arbitrarily control anisotropy and directionality of a tetrahedral mesh and a hex-dominant mesh. The hex-dominant mesh can be converted to an all-hex mesh by a set of subdivision templates, called *HEXHOOP*, that creates a high quality all-hex mesh if the given hex-dominant mesh is of reasonably high quality. The proposed method is based on three new research outcomes: (1) a node-locating algorithm that obtains good node locations by packing volumetric cells in the target geometric domain with anisotropy and directionality taken into account; (2) an anisotropic node connecting algorithm using the *advancing front method* and the *local transformation method*; and (3) modular templates for hex-dominant to all-hex mesh conversion. Ellipsoidal bubbles are packed in the target domain to obtain good node locations for a tetrahedral mesh. The nodes are then connected to yield a tetrahedral mesh with anisotropy taken into account. Similarly, rectangular solid cells are packed in the target domain to obtain good node locations for a hex-dominant mesh. The nodes are then connected to yield a tetrahedral mesh, and the tetrahedral mesh is converted to a hex-dominant mesh by merging some tetrahedrons to create hexahedrons and prisms. A hex-dominant mesh can be converted to an all-hex mesh by a set of subdivision templates called *HEXHOOP*. A subdivision template is built by combining sub-modules, and a template subdivides a hexahedron or a prism into smaller hexahedrons while maintaining the interface conformity with neighboring elements. Tetrahedrons and pyramids included in the hex-dominant mesh are also subdivided accordingly. The proposed method thus covers three major types of finite element meshes; tetrahedral mesh, hex-dominant mesh and an all-hex mesh. It avoids ill shaped elements in a tetrahedral mesh and a hex-dominant mesh induced by badly located nodes. The HEXHOOP templates guarantee the interface conformity of an output all-hex mesh, and it creates a high quality all-hex mesh, providing that the input hex-dominant mesh is of reasonably high quality.

ACKNOWLEDGEMENTS

I am grateful to my research advisor, Professor Kenji Shimada, for giving me this opportunity to pursue a Ph.D degree in the Computer Integrated Engineering Laboratory in the Mechanical Engineering Department of Carnegie Mellon University. His guidance has been invaluable to my research.

I also wish to thank Professors Cristina Amon, Jack Beuth, Paul Heckbert and Jayathi Murthy for agreeing to be my thesis committee; their suggestions and comments gave me useful guidance in the research.

I wish to thank my friends at CMU for their friendship and support. I would especially like to thank Mr. Drew Feiner for his kindness, and encouragement during difficult time while I was adjusting to the new life in Pittsburgh.

This material is based in part on work supported under a NSF CAREER Award (no. 9985288).

Some of the CAD models used in this thesis are provided through the courtesy of Honda Motor Co., Ltd. Those models helped me test the developed techniques in a realistic environment.

I am grateful to Mr. Keisuke Inoue, Dr. Takayuki Itoh, Dr. Atsushi Yamada and Mr. Tomotake Furuhata of IBM Japan for insightful technical discussions.

I wish to thank my parents for encouragement and support. Without their help, this thesis would never have been completed.

TABLE OF CONTENTS

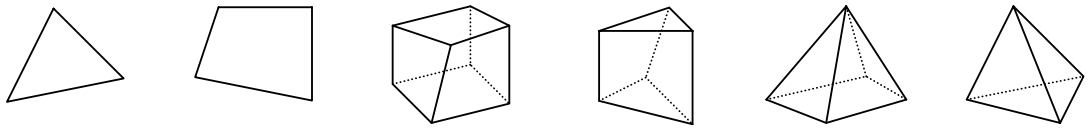
LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Introduction

This thesis presents a new computational method that creates an anisotropic tetrahedral mesh, an anisotropic hex-dominant mesh and an all-hex mesh. A tetrahedral mesh is a mesh that consists of exclusively tetrahedral elements, whereas a hex-dominant mesh consists of hexahedral elements, prism elements, pyramid elements and tetrahedral elements. An all-hex mesh is composed of exclusively hexahedral elements. A hexahedral element, a prism element, a pyramid element and a tetrahedral element are illustrated in Figure 1-1 (c), Figure 1-1 (d), Figure 1-1 (e) and Figure 1-1 (f) respectively. The proposed methods take a geometric domain as input, and subdivide, or mesh, the domain into these elements. The proposed tetrahedral meshing method, explained in Chapter 4, has two major advantages: (1) it avoids ill shaped elements that are induced when two nodes are located too close to each other; and (2) it can arbitrarily control the anisotropy of the mesh. The hex-dominant meshing method, explained in Chapter 5, has three major advantages: (1) it fills most of the volume of the target geometric domain with hexahedrons and prisms; (2) this method also avoids ill shaped elements induced by two nodes located too close to each other, and; (3) this method can arbitrarily control the anisotropy as well as the directionality of the mesh. The method explained in Chapter 6 solves heretofore a previously unsolved problem: hex-dominant to all-hex mesh conversion. As long as the quality of the input hex-dominant mesh is reasonably high, this method creates a all-hex mesh of reasonably good quality, and thus provides a new way of generating a high quality all-hex mesh.



(a) Triangle (b) Quadrilateral (c) Hexahedron (d) Prism (e) Pyramid (f) Tetrahedron

Elements for a 2D and surface domain Elements for a 3D domain

Figure 1-1 Typical elements

The proposed methods are based on two major research outcomes: (1) a node-locating algorithm that finds good node locations by packing volumetric cells on the boundary of and inside a given geometric

domain, and (2) modular templates that convert a hex-dominant mesh to an all-hex mesh. The node-locating algorithm is an expansion of a method called *bubble mesh* [1-6], and the proposed method packs volumetric cells: ellipsoidal bubbles for an anisotropic tetrahedral mesh, and rectangular solid cells for a hex-dominant mesh. The cells are moved to stable positions as determined by physically-based particle simulation, and the centers of these cells give good node locations. The proposed method then connects the nodes to create a tetrahedral mesh using a method called *advancing front* [7, 8], and quality of the tetrahedral mesh is then further improved by a method called *local transformation* [9]. If the designated output is a hex-dominant mesh or an all-hex mesh, some tetrahedrons are merged creating hexahedrons and prisms, and the mesh is converted to a hex-dominant mesh. In a hex-dominant mesh created by merging some tetrahedrons, some quadrilateral faces of hexahedrons and prisms are facing two tetrahedrons. These quadrilateral faces are called *non-conforming quadrilaterals*. If it is necessary to eliminate all non-conforming quadrilaterals, a method presented by Owen et al. [10] is applied. Their method inserts a pyramid element between two tetrahedrons and a hexahedron on each non-conforming quadrilateral. When a pyramid is inserted, the quadrilateral face of the pyramid faces a hexahedron, and the four triangular sides of the pyramid face two tetrahedrons. The two tetrahedrons are transformed so they conform to the four triangular faces of the pyramid. Various tetrahedron transformations for conforming to the triangular faces of a pyramid are explained in [10]. If the designated output is an all-hex mesh, elements included in the hex-dominant mesh are subdivided into smaller hexahedrons while maintaining interface conformity with neighboring elements by using a modular template called *HEXHOOP*. The overview of the proposed methods is also illustrated in Figure 1-2.

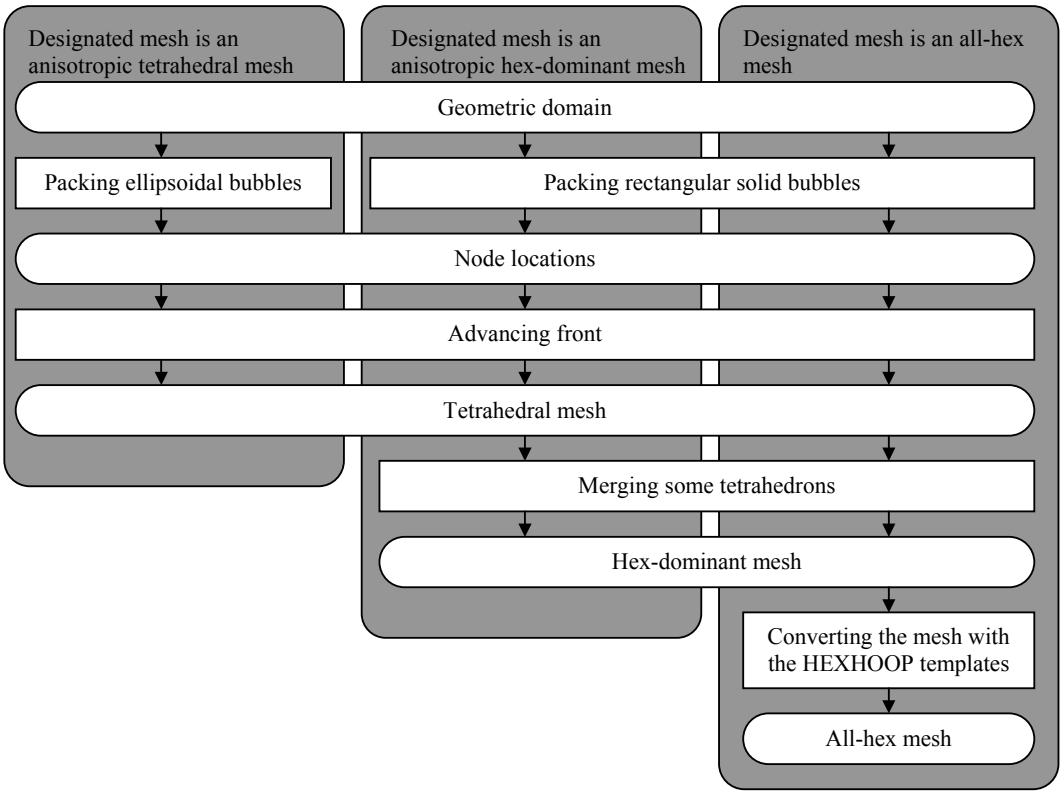


Figure 1-2 Overview of the proposed methods

The main application of the proposed method is the finite element method. The finite element method takes a mesh and boundary conditions as input and finds an approximate solution to a differential equation that governs a physical behavior over the target domain. The finite element method produces an approximate solution to a differential equation by a weighted sum of *shape functions*. A shape function is associated with a node and is defined by the elements that share the node. The weight of a shape function, called a *nodal value*, is the physical quantity at the node. Therefore the approximate solution is written in the following expression:

$$\mathbf{u}(\mathbf{x}) = \sum_i \phi_i(\mathbf{x})\mathbf{u}_i ,$$

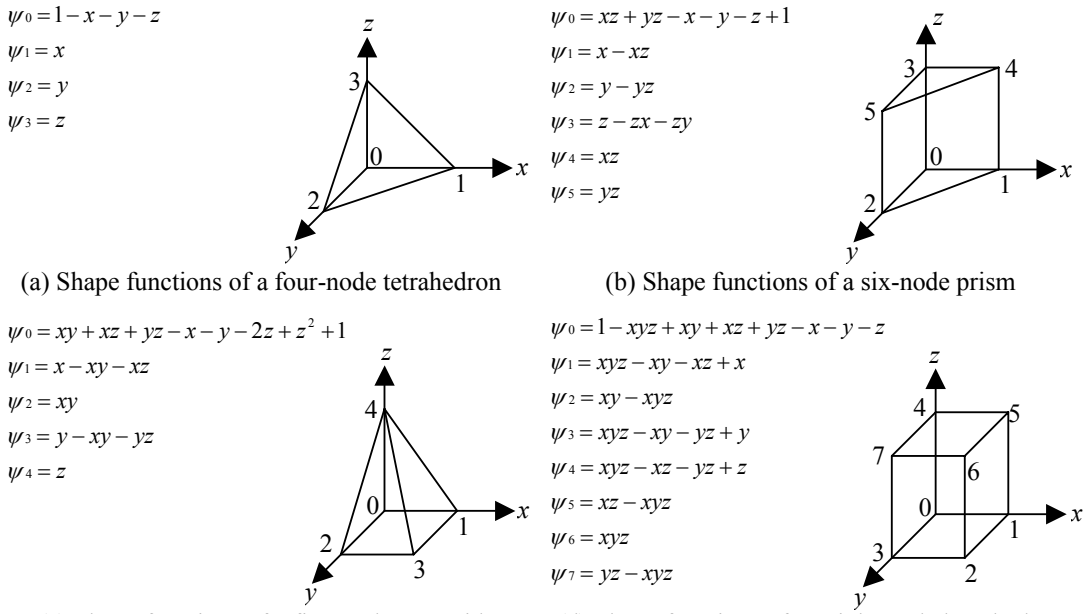
where $\mathbf{u}(\mathbf{x})$ is the approximate solution, and $\phi_i(\mathbf{x})$ is a shape function associated with the i th node, and $\mathbf{u}_i$ is a nodal value of the i th node. Since $\phi_i(\mathbf{x})$ is defined solely by the elements sharing the i th node, the only unknowns for completing the approximate solution, $\mathbf{u}(\mathbf{x})$, are nodal values, $\mathbf{u}_i$. The goal of the finite element method is to find the nodal values giving the approximate solution. To find these nodal values, the finite element method integrates the shape functions by a numerical integration scheme called *Gauss*

Legendre integration, and the matrix equation of the nodal values is created based on the integration. The finite element method then applies the *Gauss-Seidel method* or other such method for linear problems, or the *Newton-Raphson method* or other such method for non-linear problems to solve the matrix equation [11], from which the nodal values are found; and the approximate solution is finally completed. Since the finite element method gives a solution for an arbitrarily complex domain, the finite element method is widely used in industry and plays an important role in the design process.

The accuracy of the finite element solution, however, depends on the type and the quality of the mesh. A hex-dominant mesh gives a more accurate solution than a tetrahedral mesh, and an all-hex mesh generally gives a more accurate solution than a hex-dominant mesh for a given number of degrees of freedom [12]. The difference in accuracy between different types of meshes comes from the differences in the order of the shape functions. For example, the simplest forms of a three-dimensional element are an eight-node hexahedron, a six-node prism, a five-node pyramid, and a four-node tetrahedron as shown in Figure 1-1 (c), Figure 1-1 (d), Figure 1-1 (e) and Figure 1-1 (f) respectively. An eight-node hexahedron has tri-linear terms and bi-linear terms in its shape functions, whereas a six-node prism and a five-node pyramid have bi-linear terms in their shape functions. The shape functions of these elements are shown in Figure 1-3. On the other hand, a four-node tetrahedron has only linear terms in its shape functions. (A more detailed decriptions of the shape functions of those elements can be found in the reference [12].) Thus, a six-node prism and a five-node pyramid can approximate non-linear solution function more accurately than a four-node tetrahedron, and an eight-node hexahedron can approximate non-linear solution function more accurately than a prism and a pyramid. There are other forms of three-dimensional elements that have an extra node on each edge and/or face. Such elements have higher order shape functions. Nonetheless, a hexahedron with an extra node on each edge still has higher order shape functions than a tetrahedron with an extra node on each edge. Hence, a hex-dominant mesh gives a more accurate solution than a tetrahedral mesh, and an all-hex mesh theoretically gives a more accurate solution than a hex-dominant mesh for a given number of degrees of freedom [12]. Typically, a finite element analysis involving a large deformation needs an all-hex mesh because the solution function is highly non-linear; an all-hex mesh performs better than a tetrahedral mesh in such analysis. The quality of a mesh also affects the accuracy of a finite element solution. A mesh is considered to be a high quality mesh if it consists of

well shaped elements. (For a detailed discussion on mesh quality, see Section 3.2. In general a well shaped element is a chunky or stocky element, and an ill shaped element is a flat or thin element.) Ill shaped elements induce considerable error in the Gauss Legendre integration, which in turn means that the matrix equation of nodal values yields an inaccurate solution. It is believed that the error in Gauss Legendre integration becomes large for a flat or thin element because these ill shaped elements cause the Jacobian determinant to become very small. Since Gauss Legendre integration uses the Jacobian determinant, this induces significant numerical error in the integration.

Jacobian determinant, which is used in the Gauss Legendra integration (also explained in Section 3.2.2), becomes very small, and it yields a significant numerical error.



(a) Shape functions of a four-node tetrahedron (b) Shape functions of a six-node prism

(c) Shape functions of a five-node pyramid (d) Shape functions of an eight-node hexahedron

Figure 1-3 Shape functions of simplest forms of three-dimensional elements

Additionally, anisotropy, as well as the type and the quality of a mesh, affect the accuracy of the solution. When the physical quantity shows a strongly anisotropic behavior, the element must be oriented and stretched according to the anisotropy. Typically, such anisotropy is either computed from the second derivative, or Hessian matrix in multi-dimension, of the preliminary analysis solution [13-16], or from error estimation [17, 18]. For example, in the structural analysis of a carbon-fiber material, elements must be

oriented and stretched along the direction of the fiber, or in fluid mechanics analysis of a supersonic airflow, elements must be oriented and stretched along the shock layer. Anisotropic meshes have been applied in the literature to computational fluid mechanics problems [13-16], and heat transfer applications [19]. These researches have found that an anisotropic mesh gives a more accurate solution without significantly increasing the number of elements. For hexahedrons the directionality, which defines the orientations of the hexahedrons, must be taken into account. It is often required to have hexahedrons on the boundary aligned with the boundary surface and boundary edges.

Creating a high quality mesh is a challenging problem, especially in a three-dimensional domain. The mesh generation problem has become an important issue in the field of the engineering design. As shown in Table 1-1, the triangular and quadrilateral meshing problems have been well studied and their solutions are well developed. In contrast, although automatic tetrahedral mesh generation techniques are available, they often produce ill shaped elements, and they typically have a limited control of anisotropy. Furthermore, hexahedral mesh generation techniques are still in the development stage, and current methods often fail on complicated geometries. (A more detailed literature survey is presented in Chapter 2.) As a result, a considerable amount of manual intervention is required to create a three-dimensional mesh, and this manual intervention is the most time consuming portion of the analysis.

The proposed methods cover three previously unsolved three-dimensional mesh generation problems: high quality automatic anisotropic tetrahedral mesh generation, high quality automatic hex-dominant mesh generation, and fully-automatic all-hex mesh generation. A graphic representation of the problems solved is shown with shaded cells in Table 1-1. In the remainder of this chapter, a detailed problem statement is presented, followed by the organization of the thesis.

Table 1-1 Availabilities of mesh generation techniques

	Automation	High quality	Anisotropy and/or directionality
Triangular and Quadrilateral mesh	☺	☺	??
Tetrahedral mesh	☺	??	??
Hex-dominant mesh	☺	☹	☹
All-hex mesh	☹	☹	☹

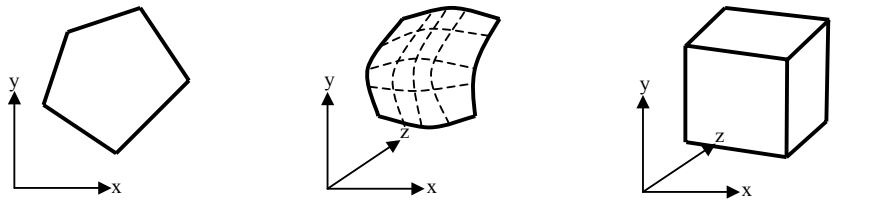
☺ …. Readily available

?? …. Does not always succeed

☹ …. Practically not available

1.2 Definition of the Mesh Generation Problem

The goal of a mesh generation problem is to subdivide a given target geometric domain into non-overlapping elements. A two-dimensional (2D) domain and a surface domain shown in Figure 1-4 (a) and Figure 1-4 (b) are subdivided into triangles and quadrilaterals, shown in Figure 1-1 (a) and Figure 1-1 (b). A three-dimensional (3D) domain shown in Figure 1-4 (c) is subdivided into hexahedrons, prisms, pyramids and tetrahedrons, shown in Figure 1-1 (c), Figure 1-1 (d), Figure 1-1 (e) and Figure 1-1 (f) respectively. The type of a mesh is classified by the type of the target domain and the types of elements included in the mesh as shown in Table 1-2. Many mesh generation algorithms are readily available for 2D and surface domains. In contrast, meshing techniques for a 3D domain are still under development, and the methods presented in this thesis are new techniques for meshing a 3D domain.



(a) A two-dimensional domain (b) A surface domain (c) A three-dimensional domain

Figure 1-4 Two-dimensional, surface, and three-dimensional meshes

Table 1-2 Classifications of meshes

Type of a domain	Type of a mesh	Type of elements included in the mesh
2D/ Surface	Triangular mesh	Triangles
	Quad-dominant mesh	Mixed quadrilaterals and triangles
	All-quad mesh	Quadrilaterals
3D	Tetrahedral mesh	Tetrahedrons
	Hex-dominant mesh	Mixed hexahedrons, prisms, pyramids and tetrahedrons
	All-hex mesh	Hexahedrons

In the mesh generation problem, an output mesh must be valid in terms of topology and geometry. The topological validity, or the interface-conformity, is satisfied if: (1) an interface between elements is used only by two elements, and if (2) no interface between two elements is exposed to the exterior of the target domain or to the third element. In a two-dimensional mesh, since there is only one type of interface, an edge, the interface-conformity is easily satisfied if every interior edge is shared by two elements. In a three-dimensional mesh, there are two types of interfaces, a triangular face and a quadrilateral face, and if a triangular face is connected to a quadrilateral face, the interface-conformity is violated because a triangular face can cover only half of a quadrilateral face, and the other half of the quadrilateral face is exposed either to the exterior of the domain or the third element. This constraint makes hex-dominant mesh generation very difficult unless pyramids are introduced to the mesh. However, some finite element solvers do not accept pyramids, but they typically accept a transition from a hexahedron or a prism to two tetrahedrons through a quadrilateral face. Such a quadrilateral face that is connected to two tetrahedrons is called a non-conforming quadrilateral. Thus the interface-conformity requirement for a hex-dominant mesh is relaxed, and two tetrahedrons can be connected to a hexahedron or a prism through a quadrilateral face. A more detailed discussion of the interface-conformity conditions for a hex-dominant mesh is described in Section 5.2, and Figure 5-2 and Figure 5-3 illustrate non-conforming quadrilateral.

A mesh is geometrically valid if the mesh satisfies two conditions: (1) there are no gaps or overlaps between adjacent elements, and (2) a Jacobian determinant, explained in Section 3.2.2, is positive everywhere in the mesh. There are small gaps or overlaps on the non-conforming quadrilaterals. However,

the finite element solver typically provides a method that reduces errors induced by the gaps and the overlaps on the non-conforming quadrilaterals.

The mesh quality affects the accuracy of the finite element method, and thus the mesh quality must be high, i.e., the mesh must consist of well shaped elements. If ill shaped elements are included in a mesh, the result of the finite element method often varies considerably from the true physical quantity. To identify the quality of the elements, many types of quality measurements are introduced. Typical quality measurements are tabulated in Table 1-3. Detailed explanations of the quality measurements are found in Section 3.2.

Table 1-3 Typical quality measurements

Type of the element	Description	Best value	Worst value
Triangles	Radius ratio (the radius of circumscribed circle divided by the radius of inscribed circle)	2.0	∞
	Minimum angle	60 deg	0 deg
Tetrahedrons	Radius ratio (the radius of circumscribed sphere divided by the radius of inscribed sphere)	3.0	∞
	Minimum dihedral angle	72 deg	0 deg
Quadrilaterals, Hexahedrons, Prisms and Pyramids	Jacobian determinant	N/A	$-\infty$
	Scaled Jacobian	+1.0	−1.0
	Aspect ratio (longest distance between two opposite faces divided by shortest distance between two opposite faces for a hex, longer distance between two opposite edges divided by shorter distance between two edges for a quad)	1.0	∞

Anisotropy and directionality must also be taken into account. For certain applications, such as computational fluid mechanics of supersonic airflow and solid mechanics of carbon-fiber materials, it is desired to have elements oriented and stretched according to a given anisotropy and directionality. For example, in a fluid mechanics application, elements must be stretched along shock layers, and in a solid mechanics application, if a material is not isotropic, such as carbon-fiber materials, elements must be aligned and stretched according to the direction of the fibers. For hexahedrons, the directionality, which

defines the orientations of the hexahedrons, must be considered. For examples, hexahedrons often must be aligned with the boundary surfaces and the boundary edges. Detailed explanation of the representation of anisotropy used in this thesis is found in Section 3.1. When a desired anisotropy is given, the mesh quality must be measured with anisotropy taken into account. Detailed discussion of this issue is found in Section 3.2.4.

The mesh generation problem can now be restated as follows: when a target 2D or 3D geometric domain is given, the domain must be subdivided into well shaped elements, and the mesh must be valid in terms of topology and geometry and be oriented and stretched according to the given anisotropy and directionality.

1.3 Organization of the Thesis

The organization of the thesis is as follows. Chapter 2 reviews previously published meshing techniques to show the pros and cons of the existing techniques. Chapter 2 also discusses the area that is not covered by the existing techniques but is covered by the proposed methods. Chapter 3 discusses the preliminary techniques that are the foundation of the proposed methods, including the representation of desired anisotropy, directionality and element size, mesh quality measurements, the modified Delaunay criterion, the 2D and surface bubble packing method, and the square packing method. Chapter 4 describes the proposed technique for anisotropic tetrahedral mesh generation, followed by the proposed technique for hex-dominant mesh generation in Chapter 5, and the proposed hex-dominant to all-hex mesh conversion technique in Chapter 6. Finally, the conclusion and the future research topic are presented in Chapter 7.

Chapter 2 Previous Research

This chapter reviews currently available meshing techniques. In spite of variety of approaches and attempts, no definitive algorithm that creates a high quality mesh of all kinds has been developed. Each algorithm has pros and cons, and thus different algorithms are employed depending on the quality requirements, desired mesh type, type of the analysis, and available resources. Although a literature survey of the mesh generation techniques is comprehensively covered by Owen [20], this chapter presents a recap of three-dimensional mesh generation techniques that are most related to the techniques presented in this thesis. The purpose of this chapter is to clarify the capabilities and limitations of currently available meshing techniques, to elucidate how the proposed methods overcome current limitations and to show the comparative advantages of the proposed methods over the current methods.

2.1 Delaunay Triangulation

The Delaunay triangulation method creates a Delaunay mesh, or a Delaunay triangulation, in which each element of the mesh satisfies the Delaunay criterion, also called a circum-sphere test for a tetrahedron and a circum-circle test for a triangle. A tetrahedron satisfies the Delaunay criterion if no node is located inside the circumscribed sphere of the tetrahedron, and a triangle satisfies the Delaunay criterion if no node is located inside the circumscribed circle of the triangle.

The most common algorithm for a Delaunay triangulation is called the *point insertion method*, or the *cavity method*. The point insertion method is first presented by Watson [21] and Bowyer [22]. Their method begins with a mesh consisting of only one element, called a *super-triangle* in 2D or a *super-tetrahedron* in 3D, which encloses the entire target domain. Nodes are then inserted in the mesh one by one. When a node is inserted in the mesh, some elements no longer satisfy the Delaunay criterion because their circum-spheres enclose the newly inserted node. Those elements are deleted from the mesh, and a void space, or a cavity, is created in the mesh. Each triangular face of the cavity is then connected to the newly inserted node in order to fill the cavity by new elements. The node insertion is repeated until the mesh gains enough resolution.

The point insertion method efficiently creates n-dimensional Delaunay triangulation. The method however, does not constrain the boundary, i.e., some faces and edges may intersect with the target domain boundary. George et al. present an algorithm that creates a boundary constrained Delaunay triangulation in three steps: (1) creating a Delaunay triangulation without constraining the boundary; (2) recovering the boundary edges, and (3) recovering the boundary faces [23]. During the boundary recovery step, their method re-connects the nodes of the tetrahedrons that intersect the boundary, and new nodes are inserted if the node re-connection cannot recover the boundary. Weatherill and Hassan present a different algorithm to create a boundary constrained Delaunay triangulation [24]. Their method also recovers boundary edges and boundary faces after constructing an unconstrained Delaunay triangulation. The difference is that their method recovers edges and faces by adding extra nodes at the intersections between the edges and the faces in the unconstrained Delaunay triangulation at the boundary, and superfluous nodes are deleted in the post process.

While the point insertion method works stably in 2D and usually creates a high quality triangular mesh, the method becomes unstable when creating a tetrahedral mesh if more than four nodes are lying on a sphere. Such degeneracy induces ill shaped elements, or in the worst case, creates self-intersecting elements. To prevent this degeneracy, Sugihara and Inagaki suggest an additional constraint for the cavity [25]. In their paper, they show that a cavity must be a star shape with respect to a newly inserted node, i.e., a line segment between the newly inserted node and each node on the cavity is entirely contained in the cavity.

With these extra constraints, the point insertion method produces a valid tetrahedral mesh. However, the degeneracy still induces a severely ill shaped element known as a *sliver*. A sliver is a tetrahedron made by warping a square slightly and adding two diagonal edges. A sliver is very flat and is ill shaped. George applies a volume threshold in order to circumvent the creation of slivers [26]. It is, however, not a fundamental solution to the problem because the volume threshold is not compatible with the Delaunay criterion. If the volume threshold is too large, the tetrahedrons that must be included in a cavity may be excluded because the resulting tetrahedrons violate the volume threshold, but if the volume threshold is too small it does not prevent creation of slivers. Shewchuk presents a method that reduces ill shaped tetrahedrons by adding a node at the center of the circum-sphere of an ill shaped element [27]. Since a

node is added at the center of the circum-sphere of an ill shaped element, the cavity includes the ill shaped element, and it is deleted as a result of the point insertion. This method effectively deletes slivers. However, adding some nodes in some region yields undesirable non-uniformity, and the method does not guarantee termination, i.e., killing a sliver potentially creates another sliver, and killing the newly created sliver may yield another sliver, and so forth possibly ad infinitum.

Several different algorithms that create a Delaunay triangulation are also proposed. Joe proposes an algorithm, called *local transformation*, that creates a Delaunay triangulation [28]. His method begins with an arbitrary tetrahedral mesh, and re-connects edges of the mesh until all the elements satisfy the Delaunay criterion. This local transformation method can also be used to improve the quality of the tetrahedrons [9]. Some experiments performed in this research indicate that his quality improvement method is very effective, and virtually all ill shaped tetrahedrons are eliminated if the nodes are ideally located. Fleischmann and Selberherr present an algorithm that creates a Delaunay triangulation by the modified advancing front method [8]. Their method is a combination of an advancing front technique and the Delaunay criterion, and will be explained in the next section.

The Delaunay triangulation method became popular because it efficiently creates a high quality triangular mesh. It also creates a tetrahedral mesh efficiently, and many of the tetrahedrons are in general well shaped. The Delaunay method, however, cannot avoid slivers when it creates a tetrahedral mesh, and ill shaped tetrahedrons can be created during the boundary recovery post-process. The method is also limited to triangular and tetrahedral mesh generation.

2.2 Advancing Front

The advancing front method creates elements starting from the boundary of the target geometric domain working inwards. The method begins with a boundary mesh, or a front, which encloses the target domain. If the target domain is a 2D or a surface domain, the front is a 1D mesh consisting of edges. If the target domain is a 3D domain, the front is a surface mesh consisting of triangles and/or quadrilaterals.

Once a front is created, one or more nodes are placed in the region enclosed by the front. The node locations are computed so that a well shaped element is created by connecting the new nodes and an

element of the front. The number of nodes connected to an element of the front depends on the type of the element to be created. If the target domain is a 2D or a surface domain, the front consists of edges, and a node is connected to an edge to create a triangle, or two nodes are connected to an edge to create a quadrilateral. If the target domain is a 3D domain, the front consists of triangles and/or quadrilaterals. A single node can be connected to a triangle to create a tetrahedron or to a quadrilateral to create a pyramid; two nodes can be connected to a quadrilateral to create a prism: or four nodes can be connected to a quadrilateral to create a hexahedron. If the following three conditions hold: (1) connecting the nodes to an element of the front yields a well shaped element, (2) the new element does not intersect with the front, and (3) the volume of the new element is entirely enclosed by the front, then the new element is added to the element list, and the front is updated so that the volume enclosed by the front excludes the volume of the new element. If all three conditions are not simultaneously satisfied, then some exception handling must be considered, such as re-considering node locations or using nodes from the opposing front instead of using the newly created nodes. This process is repeated until the volume enclosed by the front disappears, at which point the entire domain is meshed.

Lo presents an advancing front method that creates a triangular mesh for a 2D domain [29]. Peraire et al. present an advancing front algorithm that also creates a triangular mesh for a surface domain [14, 15]. Möller and Hansbo present an advancing front algorithm that creates a tetrahedral mesh [30]. Löner and Cebral present an advancing front algorithm that creates a tetrahedral mesh on a parallel computer [7, 31]. These methods create well shaped elements near the boundary in general. However, ill shaped elements are typically created where the front meets the opposing front.

Fleischmann and Selberherr present an algorithm that creates a Delaunay tetrahedrization using an advancing front technique [8]. Their method is different from typical advancing front methods in that their method first distributes nodes over the target domain, and then connects those nodes by an advancing front method constrained by the Delaunay criterion. Their method efficiently creates a Delaunay tetrahedral mesh, and if the node locations are well chosen, it can avoid ill shaped elements that typical advancing front methods create where the front collides the opposing front. In their paper, however, they do not present the node placement algorithm, and their method cannot create an anisotropic tetrahedral mesh.

Unlike the Delaunay triangulation method, the advancing front method also works for quadrilateral and hexahedral meshing. Blacker and Stephenson propose a method called *paving* for quadrilateral meshing [32]. They also present an algorithm called *plastering* for hexahedral meshing [33]. Their method places nodes using heuristics based on the geometry of the front and creates elements from the boundary inwards. Tautges et al. propose a method called *whisker weaving* [34], which first creates a topology of the hex mesh and then creates the geometry. While the advancing front method works well for quadrilateral meshing, attempts to expand the advancing front method to all-hex meshing were not so successful. Mitchell presents a necessary and sufficient condition of a quadrilateral mesh that conforms to an all-hex mesh [35]. According to his statement, any quadrilateral mesh of an even number of elements that encloses a volume has a conforming all-hex mesh. Hence, if the quadrilateral mesh satisfies Mitchell's condition, there must exist a topologically valid all-hex mesh that conforms to the quadrilateral mesh. In reality, however, the existing methods often fail to create an all-hex mesh.

Alternatively, advancing front methods that create a hex-dominant mesh have been proposed. Meyers et al. [36] and Tuchinsky and Clark [37] expand the plastering method, which is presented by Blacker and Meyers [33], to the hex-dominant mesh generation. Their method creates hexahedrons from the boundary inwards using the plastering method. If the plastering method terminates and if some volume remains to be meshed, their method fills the unmeshed volume with tetrahedrons. This method creates well shaped hexahedrons near the boundary surface; however, if the remaining volume that could not be meshed by the plastering method is flat or thin, the tetrahedrons that fill the remaining volume become ill shaped.

Owen et al. present a method called *H-Morph* [38], which converts a tetrahedral mesh to a hex-dominant mesh by creating hexahedrons one by one, starting from the domain boundary and moving inward. H-Morph is an expansion of their quadrilateral mesh generation technique called *Q-Morph* [39]. H-Morph fills most of the target geometry with hexahedrons, and most of these hexahedrons are well shaped. The H-Morph method, however, extensively modifies the input tetrahedral mesh and tends to yield ill shaped tetrahedrons. Existing hex-dominant methods for hex-dominant mesh generation tend to sacrifice the quality of the non-hex elements in favor of well shaped hexahedrons. Thus these methods are only viable when the finite element solver can tolerate those ill shaped non-hex elements.

Thompson and Soni present an algorithm that creates a quad-dominant or a hex-dominant mesh [40]. They introduce pentagonal elements to maintain conformity of a quad-dominant mesh. However, unless a finite element solver supports a pentagonal element, the output mesh cannot be used for finite element analysis. No hex-mesh result is presented in their paper.

2.3 Node Placement and Connection

Node placement and connection methods first find good node locations and then connect them using an existing method such as the Delaunay triangulation method or the advancing front method. This attempts to avoid ill shaped elements induced by two nodes located too close to each other. Since these methods typically use the Delaunay triangulation method or the advancing front method to connect nodes, they can also be viewed as variations of the Delaunay triangulation method or the advancing front method. However, the several methods differ in their node placement algorithms, and thus this section reviews node placement and connection strategies separately from the Delaunay triangulation method and the advancing front discussion.

Shimada and Gossard present an algorithm called *bubble mesh*, which creates an isotropic triangular and isotropic tetrahedral mesh [3, 5]. Their algorithm creates nodes by a physically-based particle simulation called *bubble packing*, and then the nodes are connected by the constrained Delaunay triangulation method. Shimada et al. expand the method to anisotropic triangular meshing [4], isotropic quadrilateral meshing [6], and anisotropic quadrilateral meshing [19]. Since the bubble packing method is one of the foundations of the technique presented in this thesis, a more detailed explanation of this method is found in Section 3.4 and Section 3.5.

Li et al. present a node placement algorithm called *biting* [41-43]. Their method progressively creates node locations from the boundary to the interior of a domain. The method first places nodes on the vertices of a target domain. When a node is placed, an elliptic region around the node is removed from the domain, and then new nodes are created at the intersections between the domain boundary and the elliptic region. This process is repeated until the entire domain is covered. This method produces quality node locations. Unfortunately no mesh results or quality measurements are provided in their papers [42, 43].

2.4 Grid-Based

Grid-based methods first create a structured grid over the target domain, and each interior cell of the grid is then converted to an element. Finally, the cells that intersect with the domain boundary (partially-interior cells) are transformed so that they fill the gaps between the interior cells and the domain boundary without intersecting the boundary and are then converted to elements.

Yerry and Shepard present a technique that creates a tetrahedral mesh based on an octree [44]. Their method first creates an octree from the target domain, and interior cells are subdivided into tetrahedrons while maintaining interface-conformity. Nodes are then inserted at the intersections between the domain boundary and the edges of partially-interior cells. These partially-interior cells are split into two sections, an interior section and an exterior section. Finally, the interior section of each partially-interior cell is further subdivided into tetrahedrons. Interface-conformity between tetrahedrons is maintained during the subdivision.

Schneiders presents an algorithm that generates a hexahedral mesh based on a structured grid [45]. His method first creates a structured grid called *an overlay grid*, which encloses the target domain. Each interior cell is converted to a hexahedral element. Partially-interior cells are transformed so that the cells fill the gaps between the interior cells and the domain boundary, and they are converted to hexahedral elements. His method, however, fails if the interior portion of a partially interior cell becomes a pyramid, consisting of a quadrilateral and four triangles. He states this problem, known as Schneiders' open problem, in his paper [45].

Maréchal proposes a method that creates an all-hex mesh based on an octree [46]. His method first creates an octree of the target geometric domain, and then interior cells are converted to hexahedral elements. Some interior quadrilaterals, however, face four or more neighbor hexahedrons because an octree cell may face other octree cells of different resolution. He introduces some templates that make the transition from four or more hexahedrons to a neighbor hexahedron to satisfy the interface-conformity requirement. Finally, partially-interior cells are transformed so that the intersections between the partially-interior cells and the domain boundary are eliminated, and they are then converted to hexahedral elements.

The grid-based methods often successfully create a mesh automatically, and interior elements, which are not exposed to the exterior of the domain, are well shaped. However, the major drawback to these methods is the low quality of the boundary elements, which are exposed to the exterior of the domain. When the gap between the interior cell and the domain boundary is thin or narrow, the partially-interior cell will be squeezed to fit the gap, and the elements converted from squeezed cells often becomes severely ill shaped. Due to the importance of boundary elements, on which boundary conditions are imposed, these elements need to be well shaped, and this drawback to grid-based methods significantly limits their usability.

2.5 Feature Decomposition

Feature decomposition methods subdivide a target geometric domain into simpler sub-domains, or so-called meshable sub-domains, and then each sub-domain is meshed independently. A sub-domain is typically meshed by applying a template mesh that automatically satisfies the interface-conformity requirement between sub-domains.

Several papers present methods that subdivide the domain by a variation of Medial Axis Transformation (MAT) [47-50]. Tam and Armstrong present a method that creates a quadrilateral mesh [51]. Their method extracts the medial axis of a domain by using Delaunay triangulation. The domain is then subdivided into simpler sub-domains, and the sub-domains are meshed independently. Armstrong et al. propose a method that creates a hexahedral mesh [52]. Their method extracts a medial surface from the Voronoi diagram of the domain, and the domain is then subdivided into meshable sub-domains. In their paper, they introduce various template hex-meshes that subdivide the sub-domains. Quadros et al., propose a method that creates a quadrilateral mesh called *LayTracks* [53]. Their method first extracts a medial axis, and each sub-domain is then meshed by the advancing front method, starting from the medial axis.

Sheffa and Bercovier propose a method that creates an all-hex mesh. Their method decomposes a domain into meshable sub-domains based on the *embedded Voronoi graph*, and each sub-domain is meshed into hexahedrons [54]. The embedded Voronoi graph is an approximation of the Voronoi diagram of a volume, which is presented by Etzion and Rappoport [55].

The feature decomposition methods create a high quality mesh, but only if the method successfully decomposes the domain into meshable sub-domains. However, decomposing the geometric domain into meshable sub-domains itself is a challenging problem, and it often fails if a complex target geometric domain is given.

2.6 Anisotropic Meshing

Anisotropic meshing techniques create a mesh with elements oriented and stretched according to the given anisotropy. Anisotropy is often obtained from a preliminary analysis result. For example the Hessian matrix of the preliminary analysis result is often used to obtain the ideal element size and orientation. Since the Hessian matrix is position-variant, desired element size and orientation change over the domain. An anisotropic mesh is required when the solution of an analysis shows strong anisotropic behavior. For example, in a fluid mechanics simulation, the second derivative of a physical quantity is zero or very small along a shock layer, but it is very large in the direction perpendicular to the shock layer. In such a case, elements should be aligned with the shock layer and stretched in the direction of the shock layer. Or, in structural analysis, if an object to be analyzed is composed of an anisotropic material such as glass fibers, elements should be stretched in the direction of the fibers. Several methods that obtain anisotropy based on a preliminary analysis result are presented in [13-18].

Most of the currently available anisotropic meshing techniques pertain to 2D and surface triangular meshing. Castro-Diaz et al. propose a technique that adapts a triangular mesh to an appropriate anisotropy for CFD applications [16]. The method takes an initial triangular mesh and computes a CFD solution. Anisotropy is then computed based on the results of the initial CFD solution, and the mesh is adapted to the computed anisotropy by adding, deleting and re-connecting nodes. This procedure is repeated until the estimated interpolation error in the solution becomes less than prescribed threshold.

Bossen and Heckbert introduce an anisotropic triangular mesh generation technique [56]. Their method first creates a constrained Delaunay mesh of the target domain. After creating an initial mesh, nodes are adaptively inserted or deleted. The mesh is then smoothed, refined or re-triangulated according to the given anisotropy. Repeating this process adapts the mesh to the given anisotropy. They present in their paper the modified Delaunay criterion, which is useful for connecting the nodes in an anisotropic fashion.

Since a 2x2 matrix can transform a circle into an ellipse, the modified Delaunay criterion uses a circum-ellipse test instead of a circum-circle test. Because the modified Delaunay criterion is used in the tetrahedral meshing technique presented in this thesis, further explanation of the modified Delaunay criterion is contained in Section 3.3.

Borouchaki et al. propose a method for creating anisotropic triangular and quadrilateral meshes [57]. They generalize the Delaunay kernel method [21, 22, 26] for anisotropic triangular meshing. Their method converts a triangular mesh into a quadrilateral mesh, if the designated output is a quadrilateral mesh.

Shimada et al. introduce a technique that creates an anisotropic triangular mesh on a curved surface [4]. The method packs ellipsoidal bubbles on a curved surface to obtain node locations. Those nodes are connected using the modified Delaunay triangulation. Viswanath et al. expand the method for anisotropic quadrilateral meshing [19]. Their method packs rectangular bubbles on the boundary and the inside of the two-dimensional domain to obtain node locations for an anisotropic quadrilateral mesh. Because this method consistently generates good node locations, this method results in virtually no ill shaped elements.

Garimella and Shephard present a method that creates a boundary-aligned anisotropic tetrahedral mesh for CFD applications [58, 59]. Shephard also discusses the method in his paper [60]. The method is later enhanced by Jansen [61] so that the method creates better shaped elements on the boundary. Their method first creates elements of the boundary layers through the advancing front method, and then creates elements for the remainder of the domain by another isotropic tetrahedrization technique. This method can control the density of the elements in a direction perpendicular to the boundary by adjusting the speed of the advancing front. The method, however, cannot control the anisotropy of the elements located far from the boundary.

2.7 Multi-Sweep

If the target domain is a prismatic shape, an all-hex mesh is easily created in two steps: (1) creating a quadrilateral face of the cross-section, and (2) creating hexahedrons by extruding, or sweeping, the quadrilateral mesh. The method is not limited to an all-hex mesh, and if the cross-section is meshed into a quad-dominant mesh, which consists of quadrilaterals and triangles, the sweeping method creates a hex-

dominant mesh consisting of hexahedrons and prisms. The mesh created by the sweeping method is called a *swept mesh*.

The multi-sweep method is an extension of the sweeping method, and it accepts a geometric domain with more than one cross-section. The multi-sweep method is presented by Shepherd et al. [62] and Lai et al. [63]. White et al. also present a method for reducing manual intervention in the multi-sweep method by choosing the sweeping directions and cross-sections automatically [64]. It first projects all the cross-sections onto a common plane, on which the cross-sections are merged. The merged cross-sections are then meshed into a quadrilateral mesh with all the edges of the cross-sections constrained, i.e., none of the elements of the quadrilateral mesh steps over an edge of the projected cross-sections. The quadrilateral mesh is then swept to create hexahedrons through the length of the target geometric domain. Hexahedrons located outside the target geometric domain are discarded during the sweeping process.

The multi-sweep technique creates a high quality all-hex mesh, when the target geometric domain is sweepable and the merged cross-section can be meshed into a high quality quadrilateral mesh. However, the method cannot be applied when the target geometric domain is not sweepable, and thus, it cannot be applied to a complex geometric domain.

2.8 Mesh Conversion

Mesh conversion techniques take a mesh as an input, and converts the mesh type to a different type by transforming elements. The mesh conversion technique can be applied when a mesh of the designated type is difficult to create, but it is easier to create a mesh of a different type that can be converted to a mesh of the designated type. For example, it is known that a quad-dominant mesh can easily be converted to an all-quad mesh by subdividing each triangle into three quadrilaterals and each quadrilateral into four quadrilaterals. (See Section 6.1 for more details.) There are relatively few mesh conversion techniques that can be applied to a 3D mesh, and this section reviews the major techniques.

Mitchell proposes a template called *Geode*, which converts a hex-dominant mesh consisting of hexahedrons, pyramids and tetrahedrons to an all-hex mesh [35]. His method inserts a layer of Geode templates between a volume filled with hexahedrons and a volume filled with tetrahedrons and pyramids.

Each pyramid is then subdivided into two tetrahedrons by the plane made by the diagonal of the quadrilateral face of the pyramid and the node of the pyramid not shared by the quadrilateral face. Each tetrahedron is subdivided into four hexahedrons by adding a node at the geometric center of the tetrahedron and four nodes at the geometric centers of four faces of the tetrahedron and six nodes at the centers of six edges of the tetrahedron, and each hexahedron is subdivided into eight hexahedrons by adding a node at the geometric center of the hexahedron and six nodes at the geometric centers of six faces of the hexahedron and twelve nodes at the centers of twelve edges of the hexahedron. The Geode template maintains the interface conformity while subdividing elements into hexahedrons. The Geode template, however, can be placed only when a volume filled with non-hexahedral elements is a shell-like volume, i.e., the volume is enclosed by a quadrilateral mesh. This requirement limits the application of the Geode template.

Li and Cheng propose a set of conversion templates that converts a uniform all-hex mesh to a graded all-hex mesh [65]. Their method first replaces some hexahedrons enclosing the region that needs a higher resolution, and then the hexahedrons enclosed by their templates are subdivided by the rule presented in their paper. Their method successfully increases the resolution of designated portions of the mesh, and the method creates a high quality graded all-hex mesh if the input all-hex mesh is of reasonably good quality. However, their method can increase the resolution but cannot lower the resolution. No methods are available that can lower the resolutions of some portions of the all-hex mesh. Kwak and Im propose a similar approach for remeshing in metal forming simulation [66]. Their method starts with a grid-based method to create an all-hex mesh, and applies hexahedron templates to refine some designated regions.

Meshkat and Talmor present an algorithm that converts a tetrahedral mesh into a hex-dominant mesh based on graph theory [67]. Their method takes a tetrahedral mesh as input and creates a graph that represents the topology of the tetrahedral mesh. Their method then searches the graph for a pattern that can be converted to a hexahedron or a prism. When a pattern is found, a new hexahedron or a new prism is created, and the graph is updated accordingly. The result of their method significantly depends on the input tetrahedral mesh, and a high quality tetrahedral mesh, which includes many near-equilateral tetrahedrons, does not necessarily yield many high quality hexahedrons. However, in their paper, no appropriate tetrahedral mesh generation method is presented.

2.9 Mesh Smoothing

The mesh smoothing techniques move nodes to new locations to improve the mesh quality. Since many existing automatic mesh generation techniques usually cannot create a high quality mesh by itself, the mesh smoothing technique must be applied as a post-process and is practically part of the mesh generation procedure.

Laplacian smoothing is the most commonly used smoothing scheme. It moves a node to the geometric center of the neighbor nodes, i.e., if a node is neighbor to n nodes, and ith neighbor is located at $\mathbf{x}_i$, the node is moved to the new location $\sum_{i=1}^{n} \mathbf{x}_i / n$. Although Laplacian smoothing is easy to implement, and it always converges, it does not always improve the mesh quality and may even produce inverted elements. To prevent the degradation of the mesh quality, Laplacian smoothing is often applied with a constraint: a node is moved only when the mesh quality improves, and this constrained version of Laplacian smoothing is called *smart Laplacian*. Although smart Laplacian does not worsen the mesh quality, smart Laplacian is often not effective in eliminating ill shaped elements.

Knupp proposes a smoothing scheme that improves mesh quality through a numerical optimization [68, 69]. His method constructs the cost function for the optimizer based on the Jacobian condition number. The cost function is a function of the position of a node, and the minimization of the cost function improves the mesh quality. This cost function, however, does not eliminate inverted elements. To eliminate inverted elements, his method uses a different cost function based on the Jacobian determinant. (See Section 3.2.2 for the definition of the Jacobian determinant).

Freitag and Plassmann introduce a smoothing scheme for a triangular mesh and a quadrilateral mesh that applies a linear programming scheme to eliminate inverted elements and to improve the mesh quality [70, 71]. Their method is based on two properties of the Jacobian determinant: (1) the Jacobian determinant of an element at a node is a linear function of the node location, and (2) the movement of the node only affects Jacobian determinants within the neighbor of the node. Their cost function for the optimizer is the minimum Jacobian determinant among all Jacobian determinants affected by the movement of the node, and their method maximizes the cost function with a linear programming scheme. The method continues the optimization until all Jacobian determinants becomes positive, i.e., all inverted elements are

eliminated. Their method can be directly expanded to a tetrahedral mesh, a hex-dominant mesh and an all-hex mesh.

Zhou and Shimada propose an angle-based smoothing scheme for a triangular mesh and a quadrilateral mesh that improves the quality of the elements by equating the angles of elements around a node [72]. Their method moves each node so that the angles incident to the node become as equal as possible. Their method tends to reduce the number of flat elements and is less likely to produce inverted element. Their method can be expanded to a tetrahedral mesh by equalizing the solid angles incident to a node. However, some experiments done in this research showed that the angle-based method does not improve the quality of a hex-dominant and an all-hex mesh.

Calvo and Idelsohn present an optimization based mesh smoothing scheme [73]. Their method takes scaled Jacobian (in their paper, they call normalized Jacobian) as cost function, and tries to the smallest scaled Jacobian larger. The scaled Jacobian is a quality measurement for a hexahedron and a prism, and it measures orthogonality of the element sharing a common node. The definition of the scaled Jacobian is found in Section 3.2.2. Their method applies the gradient based scheme, and the simulated annealing scheme, which moves the node randomly and takes the one movement that makes scaled Jacobian largest. If the two schemes fail, their method randomly moves the node and its neighbor nodes to reduce the chance of the lockup. Some quality improvements are shown in their paper by using some example all-hex meshes.

2.10 Summary of Previous Research

In summary, although triangular and quadrilateral meshing techniques have been well developed, 3D meshing techniques are still in the development stage. Although some anisotropic tetrahedral meshing techniques have previously been published [30, 58-61], none of them can flexibly control the anisotropy. Nor can any create a hex-dominant mesh without sacrificing the quality of non-hex elements, and furthermore none of them can create an all-hex mesh automatically for a general geometric domain. The applicability of the existing methods for each mesh type is summarized in Table 2-1, and it shows that existing methods can deal with isotropic triangular meshes, anisotropic triangular meshes, isotropic quadrilateral meshes, anisotropic quadrilateral meshes, and isotropic tetrahedral meshes. However, the methods that create isotropic hex-dominant meshes and isotropic all-hex meshes often fail, have some

limitations, or cannot avoid ill shaped elements. And, none can create anisotropic tetrahedral meshes, anisotropic hex-dominant meshes and anisotropic all-hex meshes. Among those unsolved meshing problems, the methods presented in later chapters cover anisotropic and isotropic tetrahedral meshing, anisotropic and isotropic hex-dominant meshing, and isotropic all-hex meshing.

Table 2-1 Applicability of the existing methods

	Isotropic					Anisotropic				
	Tri	Quad	Tet	Hdom	Hex	Tri	Quad	Tet	Hdom	Hex
Delaunay triangulation	☺	X	☺	X	X	X	X	X	X	X
Advancing front	☺	☺	☺	☹	☹	X	X	☹	X	X
Node-placement and connection	☺	☺	X	X	X	☺	☺	X	X	X
Grid-based	☹	☹	☹	☹	☹	X	X	X	X	X
Feature decomposition	☹	☹	☹	☹	☹	X	X	X	X	X
Anisotropic meshing	N/A	N/A	N/A	N/A	N/A	☺	☺	☹	X	X
Multi-sweep	N/A	N/A	N/A	☹	☹	X	X	X	X	X
Mesh conversion	N/A	☺	N/A	☹	☹	X	X	X	X	X

☺ Available

☹ Available but often fails, cannot avoid ill shaped elements, or has significant limitation

X Not available

Tri Triangular mesh

Quad Quadrilateral mesh

Tet Tetrahedral mesh

Hdom Hex-dominant mesh

Hex All-hex mesh

Chapter 3 Preliminary

This chapter discusses fundamental techniques used by the meshing methods presented in this thesis. All materials discussed in this chapter are previous work. Section 3.1 explains the representation of anisotropy, directionality and element size, which is necessary for both the anisotropic tetrahedral mesh generation technique presented in Chapter 4 and the anisotropic hex-dominant mesh generation technique presented in Chapter 5. Section 3.2 discusses the mesh quality measurements used in the proposed methods. Section 3.3 presents the modified Delaunay criterion, which is an anisotropic version of the Delaunay criterion, used in the proposed anisotropic tetrahedral mesh generation technique. Finally, Section 3.4 and Section 3.5 discuss two node locating algorithms: the *bubble packing method*, which is used for anisotropic triangular meshing, and the *square packing method*, which is used for quadrilateral meshing.

3.1 Desired Anisotropy, Directionality and Element Size

This section explains the representation of a desired anisotropy, directionality and element size used in the proposed methods. The tetrahedral and hex-dominant meshing methods, presented in Chapter 4 and Chapter 5, create a set of good node locations before creating elements. To create the good node locations, the meshing methods must recognize the ideal shape of the element, which is defined by an anisotropy, element size, and/or directionality, at any point on the boundary of and inside the target domain.

In the proposed methods, a 3x3 matrix represents the desired anisotropy, directionality and element size. In a 2D coordinate system, a 2x2 matrix can completely represent the anisotropy and element size [16, 56]. Similarly, a 3x3 matrix can completely represent the anisotropy and element size in a 3D coordinate system [4], and it is constructed from three orthogonal principal directions and three scalar values. Note that the three orthogonal principal directions define directionality, while three scalar values alone define an element size. Thus, the three orthogonal unit vectors $\mathbf{u} = (u_x \quad u_y \quad u_z)^T$, $\mathbf{v} = (v_x \quad v_y \quad v_z)^T$, $\mathbf{w} = (w_x \quad w_y \quad w_z)^T$, and the aspect ratio given by three positive scalar values d_u, d_v and d_w uniquely describe all three quantities: an anisotropy, directionality and element size. Using ($\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$) and (d_u, d_v, d_w), two matrices $\mathbf{R}$ and $\mathbf{S}$ are built as:

$$\mathbf{R} = \begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix}, \quad \mathbf{S} = \begin{pmatrix} d_u & 0 & 0 \\ 0 & d_v & 0 \\ 0 & 0 & d_w \end{pmatrix}.$$

By multiplying $\mathbf{R}$ and $\mathbf{S}$, a 3x3 matrix that represents anisotropy, directionality and element size is obtained as:

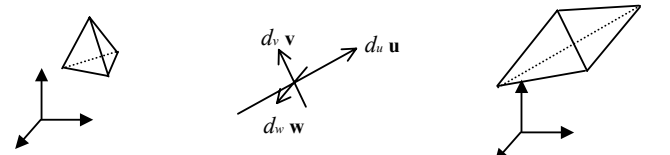
$$\mathbf{M} = \mathbf{RS} = \begin{pmatrix} d_u u_x & d_v v_x & d_w w_x \\ d_u u_y & d_v v_y & d_w w_y \\ d_u u_z & d_v v_z & d_w w_z \end{pmatrix}.$$

Since an anisotropy, directionality and element size may vary over the domain, matrix $\mathbf{M}$ is a function of position as:

$$\mathbf{M} = \mathbf{M}(\mathbf{x}),$$

where $\mathbf{x}$ is a position on the boundary of and inside the target geometric domain. The function, $\mathbf{M} = \mathbf{M}(\mathbf{x})$, is given as input to the proposed anisotropic tetrahedral mesh generation technique explained in Chapter 4 and the anisotropic hex-dominant mesh generation technique explained in Chapter 5. It defines ideal shapes of elements at any point in the target geometric domain. However, when a given $\mathbf{M}(\mathbf{x})$ changes rapidly with distance, it is often impossible to satisfy the desired anisotropy everywhere in the mesh. In such a case, the quality of the output mesh may degrade. The relation between the quality of the output mesh and the gradient of $\mathbf{M}(\mathbf{x})$ is, however, very difficult to quantify, and is a topic of future research.

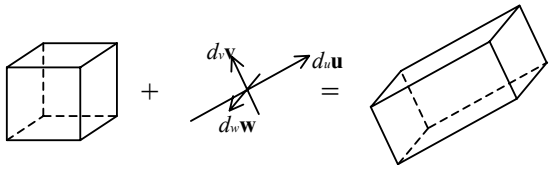
Matrix $\mathbf{M}$ is interpreted as a transformation that defines an ideal tetrahedron as shown in Figure 3-1. An equilateral tetrahedron shown in Figure 3-1 (a) is transformed into an ideal tetrahedron shown in Figure 3-1 (c) by matrix $\mathbf{M}$, defined by $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$, d_u, d_v and d_w, shown in Figure 3-1 (b).



(a) An equilateral tetrahedron (b) Anisotropy (c) An ideal tetrahedron for given anisotropy

Figure 3-1 Transformation from an equilateral tetrahedron to an ideal tetrahedron for the given anisotropy

Matrix $\mathbf{M}$ also defines an ideal hexahedron as shown in Figure 3-2. When three principal directions $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ and scalar values d_u, d_v and d_w are given as shown in Figure 3-2 (b), a unit cube shown in Figure 3-2 (a) is transformed into a rectangular solid shown in Figure 3-2 (c) by matrix $\mathbf{M}$, and the rectangular solid is an ideal hexahedron with respect to the anisotropy, directionality and element size defined by $\mathbf{M}$.



(a) A unit cube | (b) Three principal directions and three scalar values | (c) An ideal hexahedron for given anisotropy and directionality

Figure 3-2 Transformation from a unit cube to an ideal hexahedron for the given anisotropy and directionality

The expression $\mathbf{M} = \mathbf{RS}$ is different from the expression used in other papers [4, 16, 56]. In these papers, anisotropy and element size is expressed as $\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T$. This expression, $\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T$ is equal to $(\mathbf{M}^{-1})^T\mathbf{M}^{-1}$ and has sufficient information to represent an anisotropy and element size. However, when $d_u = d_v = d_w$, the expression, $\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T$, cannot represent directionality. This degeneracy can be seen by letting $d_u = d_v = d_w = d$. $\mathbf{S}$ is then simplified to:

$$S = \begin{pmatrix} d & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & d \end{pmatrix} = d \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Substituting this expression for $\mathbf{S}$ into $\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T$ yields:

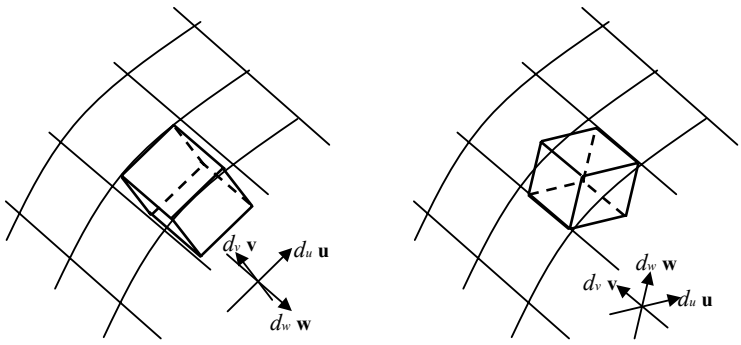
$$\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T = \frac{1}{d^2}\mathbf{R}\mathbf{R}^T.$$

However, since $\mathbf{R}$ consists of three orthogonal column vectors of unit length, $\mathbf{R}^T = \mathbf{R}^{-1}$. As a result, the expression $\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T$ becomes:

$$\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T = \frac{1}{d^2}\mathbf{R}\mathbf{R}^{-1} = \frac{1}{d^2}\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Thus, when $d_u = d_v = d_w$, the value of $\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T$ is independent of $\mathbf{R}$; therefore this expression cannot distinguish different orientations.

Although the ambiguity of the orientation is not significant in tetrahedral meshing, it is very inconvenient for the creation of hexahedrons. A hexahedron often must be oriented so that it is aligned with the boundary of the geometric domain as shown in Figure 3-3 (a). In such a case the user must explicitly specify the directionality of hexahedrons so that they are appropriately oriented along the boundary. However, if $d_u = d_v = d_w$, the expression $\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T$ cannot distinguish the boundary-aligned hexahedron shown in Figure 3-3 (a) and the boundary-unaligned hexahedron shown in Figure 3-3 (b). Therefore the user cannot explicitly tell the meshing program to align hexahedrons with the boundary with $\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T$. To avoid this problem, the meshing methods presented in this thesis use the matrix $\mathbf{M} = \mathbf{RS}$ to represent anisotropy, directionality and element size.



(a) A boundary aligned hexahedron and corresponding principal directions and scalar values

(b) A boundary unaligned hexahedron and corresponding principal directions and scalar values

Figure 3-3 A boundary aligned hexahedron and a boundary unaligned hexahedron

The conversion between these two representations of anisotropy is straightforward. If matrix **M** is given as:

$$\mathbf{M} = \mathbf{RS} = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix},$$

three principal vectors $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$, and three scalar values d_u, d_v and d_w are computed as:

$$d_u = \sqrt{{m_{11}}^2 + {m_{21}}^2 + {m_{31}}^2},$$
$$d_v = \sqrt{{m_{12}}^2 + {m_{22}}^2 + {m_{32}}^2},$$
$$d_w = \sqrt{{m_{13}}^2 + {m_{23}}^2 + {m_{33}}^2},$$
$$\mathbf{u} = \left(m_{11}/d_u \quad m_{21}/d_u \quad m_{31}/d_u\right)^T,$$
$$\mathbf{v} = \left(m_{12}/d_v \quad m_{22}/d_v \quad m_{32}/d_v\right)^T,$$
$$\mathbf{w} = \left(m_{13}/d_w \quad m_{23}/d_w \quad m_{33}/d_w\right)^T.$$

And the expression of the anisotropy, $\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T$ is re-constructed from $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$, d_u, d_v and d_w.

Conversely, the commonly used expression, $\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T$ may be converted to an equivalent $\mathbf{M}$ matrix. The three vectors $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ are the eigenvectors of $\mathbf{R}(\mathbf{S}^{-1})^2\mathbf{R}^T$, and d_u, d_v and d_w are inverse of the square-root of eigenvalues of the matrix. This eigenvalue problem can be solved by a standard algorithm such as the Jacobi method [74]. Once $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$ and d_u, d_v, d_w are known, $\mathbf{M}$ can be constructed.

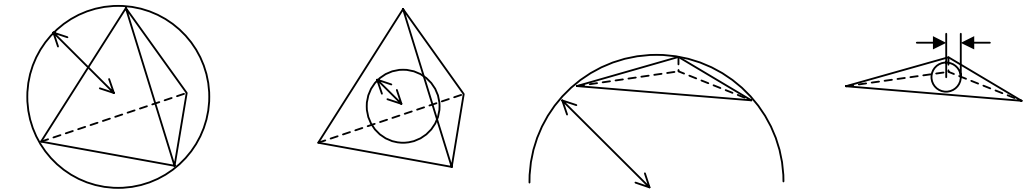
3.2 Quality Measurement

This section explains the mesh quality measurements used in the proposed methods. The proposed methods use standard metrics for mesh quality measurements, radius ratio for tetrahedrons, scaled Jacobian for hexahedrons and prisms, and aspect ratio for hexahedrons. Some other commonly used quality measurements are also reviewed.

3.2.1 Radius Ratio for Isotropic Mesh

The proposed methods measure the quality of tetrahedrons based on the radius ratio. While the proposed methods only apply the radius ratio to tetrahedrons, the quantity is also defined for triangles. The radius ratio of a tetrahedron is the radius of the circumscribed sphere divided by the radius of the inscribed sphere. Similarly, the radius ratio of a triangle is the radius of the circumscribed circle divided by the radius of the inscribed circle. The radius ratio of an equilateral tetrahedron is 3.0, and the radius ratio of an equilateral triangle is 2.0. Figure 3-4 shows the circumscribed and inscribed spheres for both a well shaped

tetrahedron and an ill shaped tetrahedron. The circumscribed sphere of the ill shaped tetrahedron tends to be larger than that of the well shaped tetrahedron. Furthermore, the inscribed sphere of the ill shaped tetrahedron is smaller compared to that of the well shaped tetrahedron. As a result, ill shaped tetrahedrons have larger radius ratio than well shaped tetrahedrons. This can clearly be seen by considering the two extreme cases of tetrahedrons. When a tetrahedron is equilateral, its radius ratio is 3.0. However, if the tetrahedron is completely flat, its radius ratio grows infinitely large. Therefore the radius ratio must be as small as possible.



A circumscribed sphere of a well shaped tetrahedron | An inscribed sphere of a well shaped tetrahedron | A circumscribed sphere of an ill shaped tetrahedron | An inscribed sphere of an ill shaped tetrahedron

Figure 3-4 Circumscribed and inscribed spheres of a well shaped and an ill shaped tetrahedron

3.2.2 Scaled Jacobian and Jacobian Determinant for Isotropic Mesh

The proposed methods measure the quality of hexahedrons and prisms by scaled Jacobian [69]. Although scaled Jacobian can be defined for a tetrahedron as well as a hexahedron and a prism, the proposed method does not use scaled Jacobian to measure the quality of a tetrahedron because radius ratio is more appropriate for a tetrahedron. If element e is a tetrahedron, a prism or a hexahedron, and if nodes $\mathbf{n}$, $\mathbf{m}_{e1}$, $\mathbf{m}_{e2}$ and $\mathbf{m}_{e3}$ are four nodes of element e as shown in Figure 3-5, then the scaled Jacobian of element e at node $\mathbf{n}$, $Js_e(\mathbf{n})$, is defined as:

$$Js_e(\mathbf{n}) = \frac{(\mathbf{m}_{e1}-\mathbf{n})\cdot(\mathbf{m}_{e2}-\mathbf{n})\times(\mathbf{m}_{e3}-\mathbf{n})}{\|\mathbf{m}_{e1}-\mathbf{n}\|\|\mathbf{m}_{e2}-\mathbf{n}\|\|\mathbf{m}_{e3}-\mathbf{n}\|}.$$

This scaled Jacobian can be interpreted as the volume of the parallelepiped defined by the three vectors $\mathbf{m}_{e1}-\mathbf{n}$, $\mathbf{m}_{e2}-\mathbf{n}$, and $\mathbf{m}_{e3}-\mathbf{n}$ divided by the maximum volume achievable with edges of that length. Nodes $\mathbf{m}_{e1}$, $\mathbf{m}_{e2}$ and $\mathbf{m}_{e3}$ are ordered so that $Js_e(\mathbf{n})>0$ if and only if element e is not inverted at node $\mathbf{n}$. The

value of $Js_e(\mathbf{n})$ becomes maximum value, 1.0, only when the element satisfies two conditions: (1) edges $\mathbf{n} - \mathbf{m}_{e1}$, $\mathbf{n} - \mathbf{m}_{e2}$ and $\mathbf{n} - \mathbf{m}_{e3}$ are perpendicular to each other, and (2) the element is not inverted at node $\mathbf{n}$. Since all angles of the quadrilateral faces of a hexahedron must be as close to 90 degrees as possible, $Js_e(\mathbf{n})$ of a hexahedron must be as large as possible. If $Js_e(\mathbf{n})$ is close to zero, it indicates that element e is flat at $\mathbf{n}$, and if $Js_e(\mathbf{n})$ is negative, element e is inverted at node $\mathbf{n}$. $Js_e(\mathbf{n})$, however, represents quality at only one point, and is not appropriate to measure the quality of the element. For example, a hexahedron that gives large scaled Jacobian at a node may give small or even negative scaled Jacobian at another node. Scaled Jacobian is thus measured at every node of the element, and the minimum one, called *minimum scaled Jacobian*, is taken as a quality measure of the element. The range of the minimum scaled Jacobian of a hexahedron is –1.0 to 1.0, of a prism –0.866 to 0.866, of a tetrahedron –0.707 to 0.707.

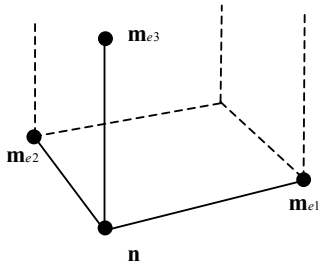


Figure 3-5 Scaled Jacobian at node n

The numerator of the scaled Jacobian, $(\mathbf{m}_{e1} - \mathbf{n}) \cdot (\mathbf{m}_{e2} - \mathbf{n}) \times (\mathbf{m}_{e3} - \mathbf{n})$, is known as the Jacobian determinant, which must be positive everywhere in the mesh. A negative Jacobian determinant indicates that the element is inverted, and the Jacobian determinant thus can tell whether the mesh is geometrically valid. However, the magnitude of Jacobian determinant depends not only on the shape of the element but also on the size of the element, and the range of the Jacobian determinant is negative infinity to positive infinity. For example, if an element is very large, the Jacobian determinant at a point within the element can be large no matter how flat the element. Conversely, a well shaped element can give very small Jacobian determinant if the size of the element is small. Hence, scaled Jacobian is more appropriate than Jacobian determinant to measure the shape quality.

3.2.3 Aspect Ratio for Isotropic Mesh

Aspect ratio is a supplemental quality measurement of a hexahedron, and it captures thin rectangular solids, which a minimum scaled Jacobian cannot capture. For example, although a thin rectangular solid shown in Figure 3-6 is an ill shaped hexahedron, the minimum scaled Jacobian of a thin rectangular solid is 1.0 since every two of three edges sharing a node makes 90 degrees, and it is not inverted. Thus, a supplemental quality measurement is required to capture a thin rectangular solid.

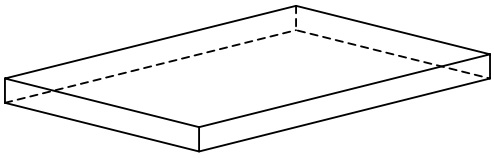


Figure 3-6 A thin rectangular solid, which is not captured by minimum scaled Jacobian

The definition of the aspect ratio is as follows. Let e_{c1}, e_{c2} and e_{c3} the distances between the geometric centers of opposite faces in a hex element as illustrated in Figure 3-7. The aspect ratio of the element is then defined as the maximum of e_{ci}/e_{cj}, $i, j = \{1,2,3\}$. The ideal aspect ratio is 1.0, and the worst aspect ratio is ∞. The aspect ratio successfully captures a thin rectangular solid since a thin rectangular solid shows a very high aspect ratio.

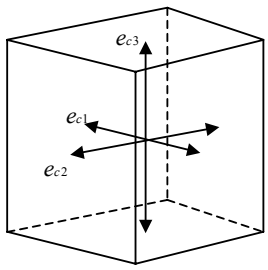


Figure 3-7 Computing aspect ratio

3.2.4 Anisotropic Quality Measurement

The quality of the element must be measured with the anisotropy taken into account, and when anisotropy $\mathbf{M}$ is given, the proposed method transforms the element by $\mathbf{M}^{-1}$ before measuring its quality. For

example, if the anisotropy is not taken into account, a tetrahedron shown in Figure 3-1 (c) is not considered a well shaped tetrahedron because it is thin, and its radius ratio is large. However, if the anisotropy $\mathbf{M}$ is given as Figure 3-1 (b), the tetrahedron shown in Figure 3-1 (c) is transformed to a tetrahedron shown in Figure 3-1 (a) by $\mathbf{M}^{-1}$. The radius ratio of the tetrahedron shown in Figure 3-1 (a) is small, and the element is thus considered a well shaped element.

3.2.5 Other Quality Measurements

In addition to the ones explained above, many other quality measurements have also been proposed. Pébay presents a survey on quality measurements for triangles and quadrilaterals [75]. Knupp presents various quality measurements for tetrahedrons and hexahedrons [68, 69]. However, no comprehensive survey on the commonly used quality measurements for 3D elements has been published, and many of the quality measurements for 3D elements are indeed redundant or insensitive to some particular case of ill shaped elements.

For example, the *minimum dihedral angle* is often used for measuring the quality of a tetrahedron. A dihedral angle is an angle between two adjacent faces. Since a tetrahedron has six edges, six pairs of adjacent faces exist in a tetrahedron, and thus six dihedral angles are defined for a tetrahedron. If the minimum dihedral angle of six dihedral angles of a tetrahedron is small, the tetrahedron is flat and ill shaped. However, an ill shaped tetrahedron does not necessarily give a small dihedral angle. A tetrahedron shown in Figure 3-8, called a *needle*, gives a reasonably large minimum dihedral angle. A needle has a summit node far from an opposite triangle, and the projection of the summit node on the plane of opposite triangle is located inside or near the opposite triangle. Nonetheless, the needle is very thin, and thus it is ill shaped. In other words, a small minimum dihedral angle implies the tetrahedron is ill shaped, but an ill shaped tetrahedron does not necessarily give a small minimum dihedral angle. Although the minimum dihedral angle is popularly used as a quality measurement of a tetrahedron, it is inappropriate because it cannot capture a particular type of ill shaped tetrahedron -- namely needles.

Any quality measurement is appropriate if it satisfies one condition: it does not give a better score for an ill shaped element than for a well shaped element. Radius ratio always becomes larger when a tetrahedron becomes flatter. The minimum scaled Jacobian always becomes smaller when a hexahedron or

a prism becomes distorted. Aspect ratio always becomes larger when a hexahedron becomes flatter. Thus, the methods presented in a later chapter use the radius ratio for tetrahedrons as quality measurement, minimum scaled Jacobian for prisms, minimum scaled Jacobian and aspect ratio for hexahedrons.

Figure 3-8 A needle: a tetrahedron that minimum dihedral angle does not capture as an ill shaped tetrahedron

3.2.6 Quality Measurements and Error in Finite Element Solution

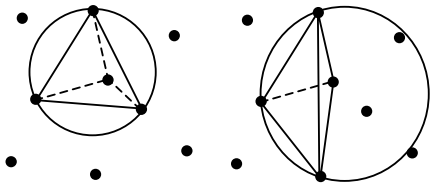
Although the three quality measurements used in this thesis (radius ratio, minimum scaled Jacobian and aspect ratio) measures an element's shape quality, the relation between the shape quality and error in a finite element solution is not clear. It is known that if the shape quality is very high, the error in a finite element solution is small. However, when the shape quality is not very high, or even low, it is very difficult to estimate error in a finite element solution; some ill shaped elements may not affect a finite element solution, or only one ill shaped element may destroy a finite element solution. Unfortunately, no systematic research on the effect of ill shaped elements has been presented.

Ideally, the mesh quality should be measured by the error estimation of a finite element analysis. However, such quality measurements are not yet developed and require future research.

3.3 Modified Delaunay Criterion

The modified Delaunay criterion is the anisotropic version of the Delaunay criterion, and the method presented in Chapter 4 uses the modified Delaunay criterion while creating an anisotropic tetrahedral mesh. The original Delaunay criterion is based on the *circumsphere test*. If no nodes are inside the circumscribed sphere of a tetrahedron, the tetrahedron is said to be valid with respect to the circumsphere test. The circumsphere test tends to choose an equilateral tetrahedron rather than a flat tetrahedron because the circumscribed sphere of an equilateral tetrahedron is smaller than the circumscribed sphere of a flat tetrahedron of the same size, and the circumscribed sphere of the equilateral tetrahedron is less likely to

enclose a node if the nodes are uniformly distributed. For example, the circumscribed sphere of an equilateral tetrahedron shown in Figure 3-9 (a) is small and does not enclose any nodes, while the circumscribed sphere of a flat tetrahedron shown in Figure 3-9 (b) is large and encloses two nodes.



(a) Circumscribed sphere of an equilateral tetrahedron

(b) Circumscribed sphere of a flat tetrahedron

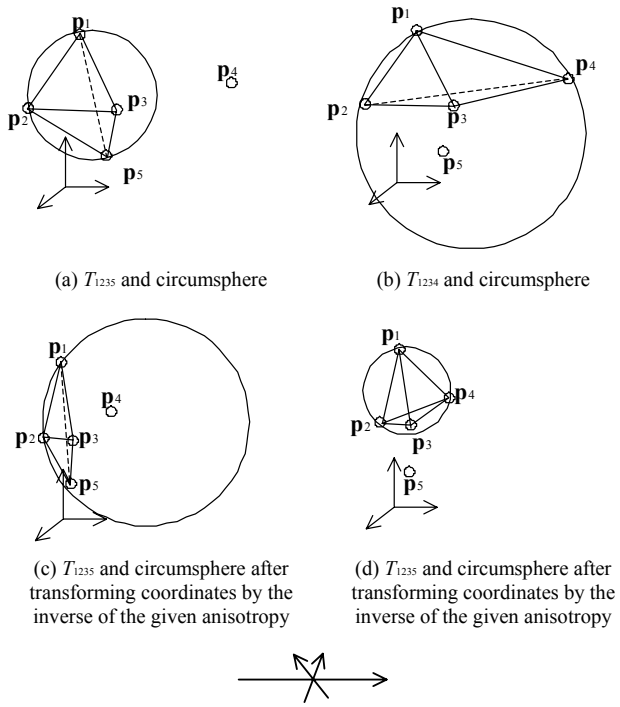
Figure 3-9 Circumsphere test

The modified Delaunay criterion expands the circumsphere test so that it takes anisotropy into account. When anisotropy $\mathbf{M}$ is given, the modified Delaunay criterion applies the circumsphere test after transforming the coordinates by $\mathbf{M}^{-1}$. The modified Delaunay criterion tends to choose an anisotropy-compliant tetrahedron, i.e., a tetrahedron oriented and stretched according to the given anisotropy, rather than a non-anisotropy-compliant tetrahedron, because the shape of an anisotropy-compliant tetrahedron becomes close to an equilateral tetrahedron after transformation by $\mathbf{M}^{-1}$.

For instance, Figure 3-10 shows a comparison between the original Delaunay criterion and the modified Delaunay criterion. There are five nodes denoted as $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$ and $\mathbf{p}_5$. Two tetrahedrons T_{1235}, which consists of $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and $\mathbf{p}_5$, and T_{1234}, which consists of $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ and $\mathbf{p}_4$, are tested by the original Delaunay criterion in Figure 3-10 (a) and Figure 3-10 (b), and by the modified Delaunay criterion in Figure 3-10 (c) and Figure 3-10 (d). The coordinates in Figure 3-10 (c) and Figure 3-10 (d) are transformed by the inverse of the given anisotropy shown in Figure 3-10 (e).

Since $\mathbf{p}_4$ is exterior to the circumscribed sphere of T_{1235} shown in Figure 3-10 (a) and $\mathbf{p}_5$ is interior to the circumscribed sphere of T_{1234} shown in Figure 3-10 (b), T_{1235} satisfies the Delaunay criterion and T_{1234} does not satisfy the Delaunay criterion. On the other hand, after transforming coordinates by the inverse of the given anisotropy, $\mathbf{p}_4$ is interior to the circumscribed sphere of T_{1235} as shown in Figure 3-10 (c), and $\mathbf{p}_5$

is exterior to the circumscribed sphere of T_{1234} as shown in Figure 3-10 (d), and, T_{1235} does not satisfy the modified Delaunay criterion, and T_{1234} satisfies the modified Delaunay criterion.



(a) T_{1235} and circumsphere

(b) T_{1234} and circumsphere

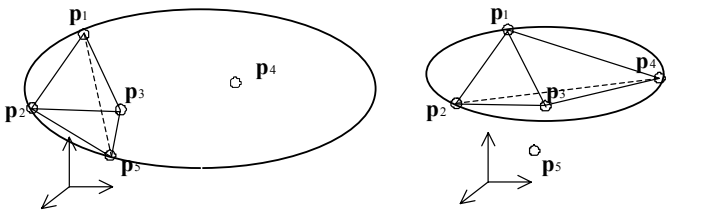
(c) T_{1235} and circumsphere after transforming coordinates by the inverse of the given anisotropy

(d) T_{1235} and circumsphere after transforming coordinates by the inverse of the given anisotropy

(e) Anisotropy for (c) and (d)

Figure 3-10 The original Delaunay criterion and the modified Delaunay criterion

The circumsphere test of the modified Delaunay criterion can be also viewed as the circum ellipsoid test. A circumscribed sphere of a tetrahedron in the coordinate system transformed by $\mathbf{M}^{-1}$ is equivalent to a circumscribed ellipsoid in the untransformed coordinate system. For example, the circumscribed spheres of tetrahedrons T_{1235} and T_{1234} in the transformed coordinate system shown in Figure 3-10 (c) and Figure 3-10 (d) are equivalent to the circumscribed ellipsoids in the untransformed coordinate system shown in Figure 3-11 (a) and Figure 3-11 (b) respectively.



(a) Circumscribed ellipsoid of T_{1235} (b) Circumscribed ellipsoid of T_{1235}

Figure 3-11 Circumscribed ellipsoids of the tetrahedrons in the untransformed coordinate system

3.4 Bubble Packing

The proposed anisotropic tetrahedral meshing method is an expansion of the *bubble packing method*, which packs ellipsoidal bubbles in the target geometric domain to obtain good node locations for an anisotropic triangular mesh [2-5]. This section explains the concept of the bubble packing method.

3.4.1 Basic Idea of the Bubble Packing Method

The idea of the bubble packing method has come from the behavior of bubbles in nature. Real bubbles packed in a finite domain will form a hexagonal pattern, which is an appropriate node configuration for a triangular mesh. Good node locations for a triangular mesh are thus obtained by packing spherical bubbles in the target 2D/surface domain, and the triangular elements are created by connecting the nodes as shown in Figure 3-12.

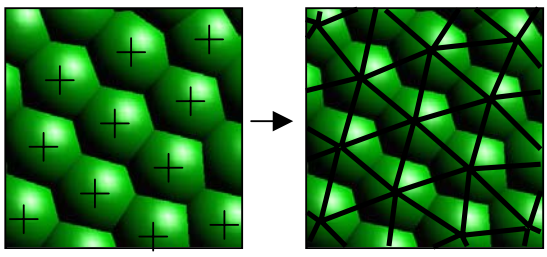


Figure 3-12 Bubble packing for an isotropic triangular mesh

The bubble packing method is expanded to anisotropic triangular meshing by replacing spherical bubbles with ellipsoidal bubbles. When an anisotropy is given as shown in Figure 3-13 (b), a sphere shown

in Figure 3-13 (a) is transformed into an ellipsoid shown in Figure 3-13 (c) by the given anisotropy. Hence, if ellipsoidal bubbles are closely packed in a 2D/surface domain, the centers of the ellipsoidal bubbles will give good node locations for an anisotropic triangular mesh as shown in Figure 3-14. After obtaining node locations, the nodes are connected by a standard node connecting algorithm such as Delaunay triangulation with the modified Delaunay criterion [16, 56]. Thus, if a computer program can find a configuration of tightly packed ellipsoidal bubbles, it can create a high quality anisotropic triangular mesh.

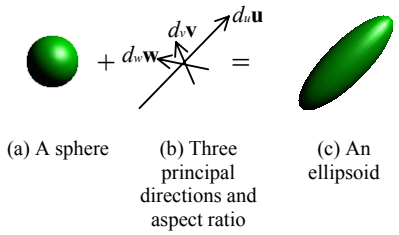


Figure 3-13 An anisotropic bubble

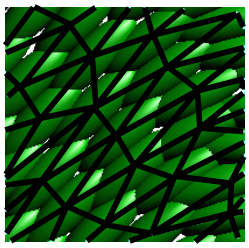


Figure 3-14 Ellipsoidal bubble packing for an anisotropic triangular mesh

To find a configuration of tightly packed ellipsoidal bubbles, the bubble packing method runs a physically-based simulation of the motions of the ellipsoidal bubbles. The bubble packing method creates ellipsoidal bubbles in the target geometric domain, and the bubbles are moved to the stable positions by a physically-based particle simulation. To obtain the best node locations, the bubbles must be closely packed inside the domain. Closely packed bubbles will have minimum gaps and overlaps. If an appropriate number of bubbles are closely packed in the domain with respect to the given anisotropy, bubbles will move to a stable configuration in which all the proximity-based inter-bubble forces are balanced, as described in the next section.

3.4.2 Computation of the Motion of the Bubbles

To run the simulation, the equation of motion that governs the dynamic behavior of the bubbles must be derived. Once the equation of motion is derived, a standard numerical integration scheme such as Euler's method or the 4th order Runge-Kutta method can simulate the motion of the bubbles. The equation of motion is written in the form:

$$m\ddot{\mathbf{x}} = \mathbf{F}$$

where m is the mass of the bubble, $\mathbf{x}$ is the position of the bubble, and $\mathbf{F}$ is the total force acting on the bubble.

In the bubble packing method, forces acting on the bubbles are computed approximately based on a mass-spring-damper model. In this model, two kinds of forces act on the bubbles. One is an inter-bubble force, which is analogous to a non-linear spring. The inter-bubble force is a proximity force and acts only between two adjacent bubbles. The other type of force is viscous dumping.

In the mass-spring-damper model, each bubble has two state variables -- position and velocity, and two attributes -- mass and damping coefficient. The state variables of ith bubble are denoted as:

$\mathbf{x}_i$ …. Position of bubble i

$\dot{\mathbf{x}}_i$ …. Velocity of bubble i

The bubble packing method assumes that all bubbles have the same mass m and the same damping coefficient c as:

$$m_i = m$$

$$c_i = c$$

With those state variables and attributes in place, the forces acting on the bubbles can be formulated.

The inter-bubble force is approximated as a force produced by a non-linear spring connecting the centers of adjacent bubbles. Figure 3-15 shows an example of the non-linear spring. To compute the inter-bubble force between bubble i and bubble j, the current length and the neutral length of the spring are needed. The current length l of the spring between bubble i and bubble j is computed as:

$$l = \left\| \mathbf{x}_i - \mathbf{x}_j \right\|$$

where $\mathbf{x}_i$ is the center of bubble i, $\mathbf{x}_j$ is the center of bubble j, and $\| \cdot \|$ denotes the Euclidean norm.

The neutral length of the spring l_0 is defined as:

$$l_0 = l_i + l_j$$

where l_i is the distance from point $\mathbf{x}_i$ to the intersection of line segment $\mathbf{x}_i$ - $\mathbf{x}_j$ and the boundary of bubble i. l_j is the distance from point $\mathbf{x}_j$ to the intersection of line segment $\mathbf{x}_i$ - $\mathbf{x}_j$ and the boundary of bubble j.

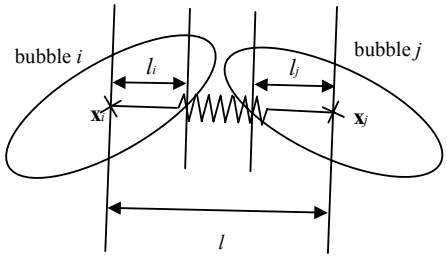
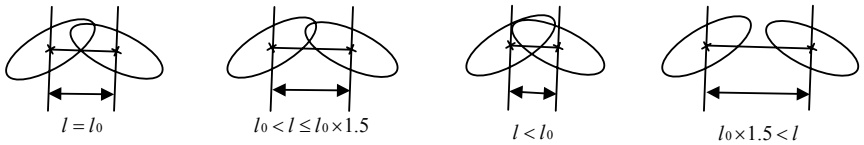


Figure 3-15 Non-linear spring between two adjacent bubbles

After finding l and l_0, the inter-bubble force is computed as a function of the ratio of l against l_0. Two adjacent bubbles either attract, repel or do not interact depending on the value of l/l_0. Figure 3-16 shows how the bubbles interact: (1) when $l/l_0 = 1$, two bubbles are at the stable distance and do not either attract or repel as shown in Figure 3-16 (a), (2) when $l/l_0 < 1$, they repel each other as shown in Figure 3-16 (b), (3) when $1 < l/l_0 < 1.5$, they attract each other as shown in Figure 3-16 (c), and (4) when $1.5 < l/l_0$, again they do not interact as shown in Figure 3-16 (d).



(a) stable (b) attracting (c) repelling (d) No interaction

Figure 3-16 Interactions between two adjacent bubbles

In the bubble packing method, the magnitude of the force of the non-linear spring is defined as the following cubic function:

$$f(w) = \begin{cases} k(1.25w^3 - 2.375w^2 + 1.125) & \text{for } 0.0 \le w \le 1.5 \\ 0 & \text{otherwise} \end{cases}$$

where w is the ratio of l and l_0, i.e., $w = l / l_0$, and k is a constant. The plot of this function is shown in Figure 3-17. Finally, the inter-bubble force acting on bubble i produced by bubble j is obtained as:

$$\mathbf{f} = f(w)\frac{\mathbf{x}_i - \mathbf{x}_j}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|}$$

Since multiple bubbles can be adjacent to a given bubble i, the total inter-bubble force acting on the ith bubble can be written as:

$$\mathbf{f}_i = \sum_j \mathbf{f}_{ij}$$

where

$$\mathbf{f}_{ij} = \begin{cases} \text{Force produced by bubble } j \text{ against bubble } i & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases},$$

and j is all bubbles such that $f(w_j) \neq 0$.

To make the system of bubbles converge to a stable configuration, damping force must be added to the system. A damping force acting on bubble i is proportional to the velocity of the bubble, and it is defined as:

$$-c\dot{\mathbf{x}}_i$$

The total force acting on bubble i is then written as:

$$\mathbf{F} = \mathbf{f}_i - c\mathbf{x}_i$$

Finally, by substituting $\mathbf{F}$ into the equation of motion, $m\ddot{\mathbf{x}} = \mathbf{F}$, the second order ordinary differential equation that describes the motion of the bubbles is written as:

$$m\ddot{\mathbf{x}}_i + c\dot{\mathbf{x}}_i = \mathbf{f}_i$$

A numerical integration scheme such as Euler's method or the 4th order Runge-Kutta method can solve this ordinary differential equation.

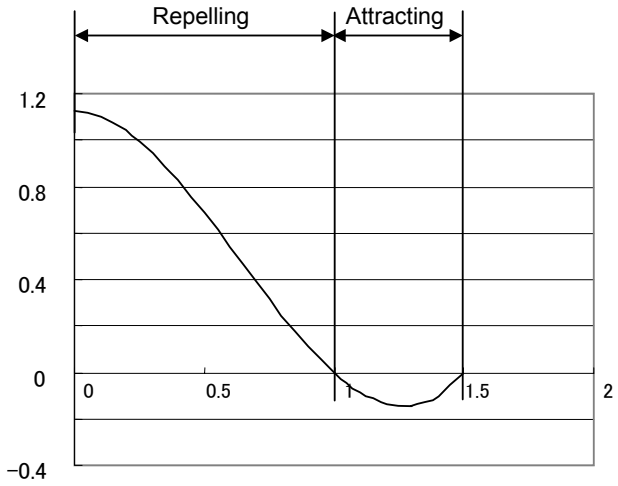


Figure 3-17 Plot of *f*(*w*)

3.4.3 Constraining Edge Bubbles and Face Bubbles on the Boundary

Bubbles packed on the boundary edge (edge bubbles) must be constrained on the edge, and bubbles packed on the free form surface (face bubbles) must be constrained on the surface. However, because the displacements of the bubbles may have a component parallel to the normal direction of the boundary, the bubbles may stray away from the boundary unless the bubbles are explicitly constrained on the boundary. To constrain all edge bubbles on the edges and all face bubbles on the faces, bubbles are moved in the parametric space instead of in the Cartesian space [4]. However, because the displacements are computed in Cartesian space, the proposed method converts the Cartesian displacement to the parametric displacement. If an edge bubble is moving on an edge given as a parametric curve $C(s)$ and is located at $\mathbf{x} = C(s_0)$, the Cartesian displacement $\Delta\mathbf{x}$ is converted to the parametric displacement Δs, and the bubble is moved to the new location $\mathbf{x}' = C(s_0 + \Delta s)$. If a face bubble is moving on a face given as a parametric surface $S(u,v)$ and is located at $\mathbf{x} = S(u_0, v_0)$, the Cartesian displacement $\Delta\mathbf{x}$ is converted to the parametric displacements Δu and Δv, and the bubble is moved to the new location $\mathbf{x}' = S(u_0 + \Delta u, v_0 + \Delta v)$.

The parametric displacement on curve $C(s)$ is approximately computed as follows. Suppose that the bubble is located at $\mathbf{x} = C(s_0)$ as shown in Figure 3-18. The tangential vector of the curve at $\mathbf{x}$ is computed as:

$$C^s(s_0) = \begin{pmatrix} \frac{\partial C_x(s_0)}{\partial s} \\ \frac{\partial C_y(s_0)}{\partial s} \\ \frac{\partial C_z(s_0)}{\partial s} \end{pmatrix}$$

If the unconstrained Cartesian displacement is $\Delta\mathbf{x}$, the parametric displacement Δs is computed as:

$$\Delta s = \frac{d_s}{\left\| C^s(s_0) \right\|}$$

where

$$d_s = \frac{\Delta\mathbf{x}_i \cdot C^s(s_0)}{\left\| C^s(s_0) \right\|} .$$

The parametric displacements on surface $S(u,v)$ are approximately computed as the parametric displacements on two isoparametric curves. Suppose that the face bubble is located at $\mathbf{x} = S(u_0, v_0)$, and the isoparametric curves at $v = v_0$ and $u = u_0$ are $C^u(u) = S(u, v_0)$ and $C^v(v) = S(u_0, v)$ respectively. If the unconstrained Cartesian displacement is $\Delta\mathbf{x}$, the parametric displacement in u direction, Δu, is the parametric displacement on the curve $C^u(u)$ corresponding to $\Delta\mathbf{x}$, and the parametric displacement in v direction, Δv, is the parametric displacement on the curve $C^v(v)$ corresponding to $\Delta\mathbf{x}$.

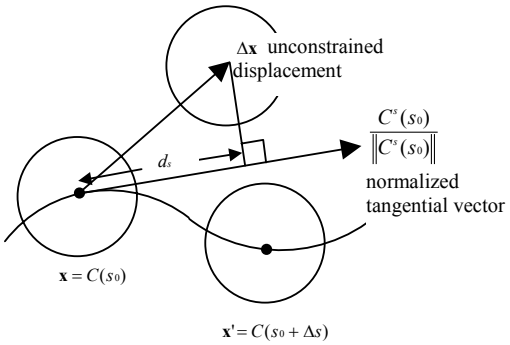


Figure 3-18 Constraining an edge bubble on an edge

3.4.4 Population Control

Another important procedure during bubble packing is a process called *adaptive population control*. The purpose of adaptive population control is to adjust the number of bubbles so that there are no significant gaps or overlaps in the final packed configuration. As mentioned above, bubbles should be closely packed to obtain high quality node locations. If the number of bubbles is too small, large gaps exist between bubbles, and the bubbles will position in a random pattern yielding a low quality mesh. If the number of bubbles is too large, it is known that the bubbles form an orthogonal pattern that is not suitable for tetrahedral meshing. To obtain the best node locations, it is important to adjust the number of bubbles during the dynamic simulation.

The optimal number of bubbles is a function of a domain geometry and a specified anisotropy, and it can be written as:

$$N_{opt} = N(\Omega, \mathbf{M})$$

where

N_{opt} : Appropriate number of bubbles
Ω : Target domain
$\mathbf{M}$: Anisotropy

If the domain is simple in terms of geometry (i.e., no sharp corners or small features) and if no anisotropy is given (i.e., a constant $\mathbf{M}$ over the domain), the optimal number of bubbles is estimated by:

$$N_{opt} = \alpha \frac{V(\Omega)}{V_b}$$

where

$V(\Omega)$: Volume of the target domain
V_b : Volume of a bubble (after transformed by $\mathbf{M}$)
α : Constant (Typically 0.9 to 1.0)

The bubble packing method can use this formula to estimate an appropriate number of bubbles in a small volume around a sample point. Although the bubble packing method makes use of the formula to make an initial guess of the bubble configuration, when the domain is complicated and a varying anisotropy is specified, this formula gives only a rough estimation of the appropriate number of bubbles.

Because the initial estimation always varies from the exact solution, the bubble packing method controls the number of bubbles adaptively through the packing process. The bubble packing method can take advantage of the computation of the inter-bubble forces to find an over-populated region and an under-populated region. If a bubble is in an over-populated region, it receives excessive repelling forces. In contrast, if a bubble is in an under-populated region, it receives more attracting forces than repelling forces. After finding over-/under-populated regions, the bubble packing method deletes some bubbles from over-populated regions and adds some bubbles in under-populated regions once every some iterations. Appropriate thresholds should be chosen to identify over-/under-populated regions.

3.5 Square Packing Method

The proposed hex-dominant meshing method is an expansion of the method called the *square packing method*, which expands the ellipsoidal bubble packing method to the quadrilateral meshing of a 2D domain [6].

The square packing method takes a 2D geometric domain Ω and directionality $\mathbf{M} = \mathbf{M(x)}$ as input and packs square cells instead of ellipsoidal bubbles in the target geometric domain to obtain good node locations for a quadrilateral mesh. The nodes are connected by a standard triangulation method such as the Delaunay triangulation method [21, 22], and some triangles are merged and converted to quadrilateral elements by the method presented by Itoh et al. [1]. Since the square packing method is for 2D domain and isotropic, directionality $\mathbf{M} = \mathbf{M(x)}$ is specified as:

$$\mathbf{u} = \mathbf{u(x)} = \begin{pmatrix} \cos\theta(\mathbf{x}) & \sin\theta(\mathbf{x}) & 0 \end{pmatrix}^T$$

$$\mathbf{v} = \mathbf{v(x)} = \begin{pmatrix} -\sin\theta(\mathbf{x}) & \cos\theta(\mathbf{x}) & 0 \end{pmatrix}^T$$

$$\mathbf{w} = \mathbf{w(x)} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T$$

$$d = d(\mathbf{x}) = d_u(\mathbf{x}) = d_v(\mathbf{x}) = d_w(\mathbf{x})$$

$$\mathbf{M(x)} = \begin{pmatrix} d\mathbf{u} & d\mathbf{v} & d\mathbf{w} \end{pmatrix}$$

where d is a given ideal edge length, and $\theta = \theta(\mathbf{x})$ is a function of a position that defines the orientation of the ideal quadrilateral at a point on the boundary of and inside the domain.

A square cell consists of five bubbles, one bubble at the center of the cell and four bubbles at the four corners of the cell as shown in Figure 3-19. The bubble at the center of the cell is the main bubble, and the bubbles at the four corners of the cells are called sub-bubbles. If the edge length of the cell is d, the diameter of the main bubble is d, and the diameter of the sub bubble is $\left(\sqrt{2}-1\right) d$.

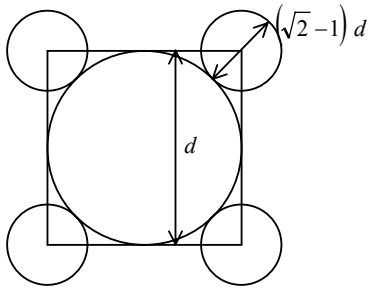


Figure 3-19 A square cell and its bubbles

For the square packing method, the equation of motion that governs the motion of the i th square cell is defined as:

$$m\ddot{\mathbf{x}}_i = \mathbf{F}_i$$

where m is a mass of the i th square cell, $\mathbf{x}_i$ is the position of the center of the i th square cell, $\mathbf{F}_i$ is total force acting on the main bubble of the i th square cell. The total force acting on the main bubble of the square cell is the sum of inter-bubble forces and damping force. Thus, the equation of motion becomes:

$$m\ddot{\mathbf{x}} + c\dot{\mathbf{x}} = \sum \mathbf{f}$$

where c is the damping coefficient, and $\mathbf{f}$ is an inter-bubble force, explained in Section 3.4.2, acting on the main bubble of the cell. (See also Section 5.4 for details.)

Similar to the ellipsoidal bubble packing, population of the cells must be adaptively adjusted during the simulation. Some cells are deleted from over-populated regions, and some cells are added to under-populated regions. The square packing method uses proximity information created during the simulation to identify over-populated regions and under-populated regions.

With this simulation, the square cells tend to converge to a structured grid pattern since there are four equilibrium positions that are orthogonal to each square cell with respect to the given directionality as shown in Figure 3-20. In Figure 3-20, square cell A has four adjacent square cells, and the main bubbles of the adjacent square cells are marked as B, C, D and E. If the configuration of the square cells is as shown in Figure 3-20, no force is acting on the adjacent bubbles, and the configuration is stable. Hence, if the appropriate number of square cells are packed in the target geometric domain, a square cell tends to come close to one of four equilibrium positions of an adjacent cell.

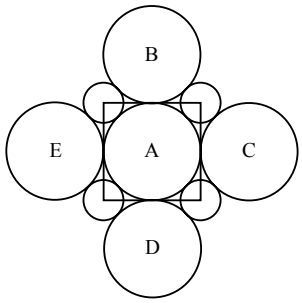
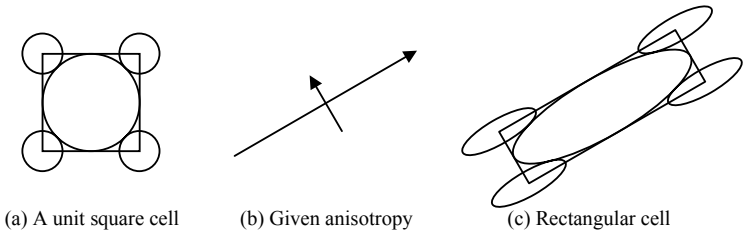


Figure 3-20 Four equilibrium positions around a square cell

Viswanath et al. expands the square packing method to anisotropic quadrilateral meshing [19]. Their method omits the restriction $\lambda = \lambda_u = \lambda_v$ of the directionality, and λ_u and λ_v can have different values. If an anisotropy is given as Figure 3-21 (b), a square cell shown in Figure 3-21 (a) is transformed into a rectangular cell shown in Figure 3-21 (c). Five bubbles of the square cell is also transformed by the anisotropy and become ellipses. Thus, when an anisotropy is given, the square cell is replaced with a rectangular cell, and inter-bubble forces must be computed with anisotropy taken into account. Nonetheless, the formulation of the equation of motion of the rectangular cell is exactly same as the equation of motion of the square cells, because $\mathbf{f}$ term of the equation of motion can take anisotropy into account as explained in Section 3.4.2.



(a) A unit square cell (b) Given anisotropy (c) Rectangular cell

Figure 3-21 Rectangular cell

Chapter 4 Anisotropic Tetrahedral Mesh Generation via Packing Ellipsoidal Bubbles

4.1 Introduction

From here on is new work. This chapter presents a new computational method for anisotropic tetrahedral meshing. The proposed method has two major advantages: (1) the proposed method can control shapes of the elements by arbitrary anisotropy function, and (2) the proposed method can avoid ill shaped elements induced from bad node locations. Many algorithms are available for tetrahedral meshing, such as the Delaunay triangulation [21, 26, 28] and the advancing front method [7, 8], but they are only applicable to isotropic graded meshes. Although many applications require anisotropic tetrahedral meshes, none of the existing method works for an arbitrarily specified anisotropy.

The proposed method extends the Bubble Mesh method [2-4] (explained in Section 3.4) to anisotropic tetrahedral meshing. The proposed method takes a geometric domain Ω and anisotropy $\mathbf{M(x)}$ as inputs and packs ellipsoids on the boundary of and inside geometric domain Ω to obtain good node locations. The nodes are then connected to create tetrahedrons. The meshing process thus consists of two major steps: (1) creating node locations by bubble packing, detailed in Section 4.2, and (2) connecting nodes with taking anisotropy into account by the advancing front method, detailed in Section 4.3.

4.2 Anisotropic Node Distribution by Packing Ellipsoids

This section describes one of the two steps of the proposed meshing procedure. The main technical issue of the first step is a node-locating algorithm called *bubble packing*. The bubble packing process consists of four sub-steps, as illustrated in Figure 4-1.

1. Place bubbles on vertices of the target domain Ω
2. Pack bubbles on edges of the target domain Ω
3. Pack bubbles on faces of the target domain Ω
4. Pack bubbles inside the target domain Ω

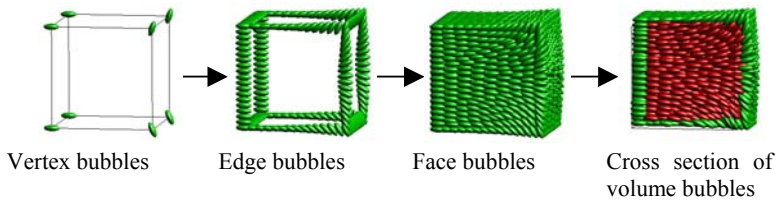


Figure 4-1 Packing ellipsoidal bubbles in the domain

During sub-processes 2 through 4, bubbles are created on the boundary of and inside target geometric domain Ω and moved to the stable positions by the physically-based particle simulation. To obtain the best node locations, the bubbles must be closely packed in the domain. Closely packed bubbles will have minimum gaps and overlaps. If an appropriate number of bubbles are closely packed in the domain with respect to the given anisotropy $\mathbf{M}(\mathbf{x})$, bubbles will move to a stable configuration in which all the proximity-based inter-bubble forces are balanced.

A pattern of the packed ellipsoids yields node locations suitable for anisotropic tetrahedral meshing because the pattern mimics ideal Volonoi polyhedra of an anisotropic tetrahedral mesh. The method thus creates well spaced nodes with respect to the given anisotropy. Since ill shaped elements are typically created where two nodes are located too closely to each other or where the density of the nodes is less than ideal, our method can avoid those ill shaped elements.

4.2.1 Computation of the Motion of the Bubbles

Like the 2D/surface bubble mesh method, the proposed method integrates a second order ordinary differential equation to compute the motions of the bubbles. The proposed method uses the same equation of motion as the 2D/surface bubble mesh method explained in Section 3.4.

4.2.2 Speed of Convergence

Bubbles packed in the domain move quickly at the beginning and rapidly decelerate as they move close to the stable positions. Figure 4-2 shows the average speed of the bubbles vs. number of iterations when spherical bubbles are packed inside the cylindrical domain. The diameter of the cylinder is 44 and the height is 35.0. The diameter of the bubbles is 4.0. As shown in Figure 4-2, the average speed of the

bubbles has two peaks. One peak appears while packing bubbles on the faces and the other peak appears while packing bubbles inside the domain. The first peak comes just after the population control function is carried out for the first time after the initial bubble configuration is made for the face bubbles. Similarly, the second peak comes just after the population control function is carried out for the first time after the initial bubble configuration is made for the interior bubbles.

Because of the nature of the second order system and the presence of numerical errors, the speed of a bubble never becomes completely zero. Since the motion of the bubbles is described by a second order differential equation, the speed of the bubbles will exponentially decrease but never reach exactly zero. Also, numerical errors accumulate over the domain, and a bubble does not decelerate less than a certain speed.

The bubbles, however, are almost stable after the speeds of the bubbles become reasonably slow. In this case, the average speed of the bubbles becomes 0.25 after 500 iterations and 0.18 after 700 iterations. The bubbles no longer move significantly after 700 iterations.

Another possible termination criterion is the quality of an output mesh. Figure 4-3 (a) plots the relation between the number of iterations and the worst radius ratio. Observe that the worst radius ratio rapidly improves with an early stage of the bubble packing, and then it stays below 7 after 420 iterations. Figure 4-3 (b) plots the relation between the number of iterations and the average radius ratio. Note that the average radius ratio gradually improves and reaches a value below 3.5 after 660 iterations. This suggests that the output mesh will have high quality if the bubble packing runs more than 700 iterations.

In reality, however, a complex geometry or complex anisotropy may induce slow convergence locally. Thus, in practice, it is best to run 1,000 to 1,500 iterations. Predicting such a slow convergence needs a sophisticated feature recognition technique, and hence is a future research topic.

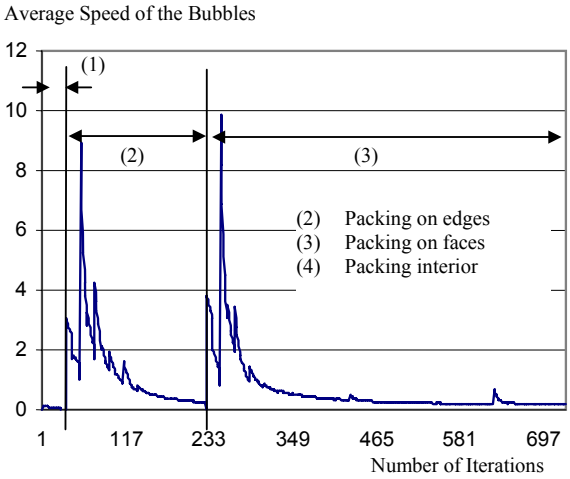
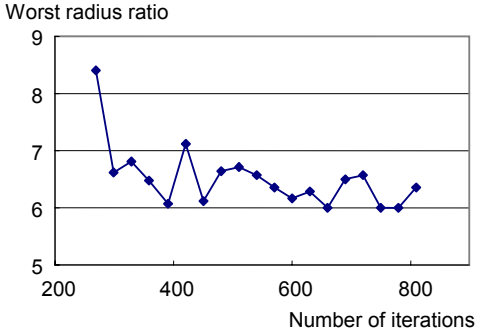
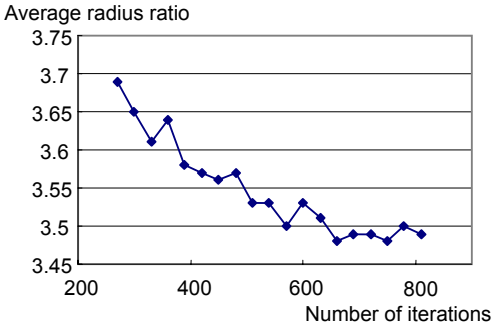


Figure 4-2 Speed of Convergence



(a) Plot of the worst radius ratio against the number of iterations



(b) Plot of the average radius ratio against the number of iterations

Figure 4-3 Plot of the quality of output tetrahedral meshes against the number of iterations

4.3 Finding Node Connectivity Taking Anisotropy into Account

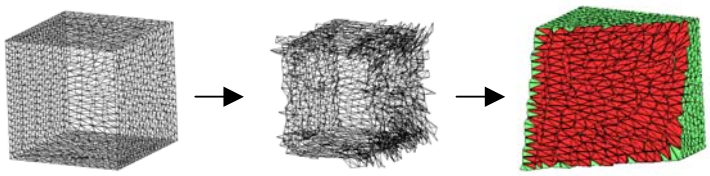
In the second step of the meshing procedure, the proposed method connects nodes to construct an anisotropic tetrahedral mesh by the advancing front method, and Barry Joe's local transformation method is applied after creating a mesh to further improve the mesh quality. Although the proposed method obtains good node locations by the bubble packing method, many tetrahedrizations are possible for a given set of nodes. However, there are only a limited number of tetrahedrizations that yield a high quality mesh

elements with respect to the given anisotropy. Thus, it is important to make sure that an appropriate tetrahedrization is chosen.

4.3.1 Advancing Front

The proposed method employs the advancing front method to connect nodes. Unlike typical advancing front methods [7, 14, 15, 29, 30], the proposed method does not create a new node during the process because the node locations are already obtained by the bubble packing method. The proposed method is rather similar to the algorithm presented by Fleischmann [8], which connects pre-created nodes and boundary triangles to form tetrahedral elements. His method skips rigorous geometry check by taking advantage of two properties of the Delaunay tetrahedrization: (1) Delaunay tetrahedrization always exists for any set of nodes, and (2) the circum-sphere test is insensitive to the orientation. However, the two properties are not always true in an anisotropic tetrahedral mesh, i.e., a tetrahedrization such that every element satisfies the modified Delaunay criterion may not exist for a given anisotropy, and the orientation and aspect ratio of a circum-ellipsoid varies among different tetrahedrons. Where the gradient of $\mathbf{M}$ is large, creating a tetrahedron that satisfies the modified Delaunay criterion will typically prevent the adjacent tetrahedrons from satisfying the modified Delaunay criterion. The proposed method thus rigorously checks the geometry of every element during the procedure to avoid intersections of elements. The detail of the procedure is explained in the following.

The proposed method first meshes the boundary into a triangular mesh with anisotropy taken into account by the modified Delaunay triangulation method [4, 16, 56] as shown in Figure 4-4 (a). This triangular mesh is the *initial front* and will become the exterior triangles of the final tetrahedral mesh.



(a) Initial front (b) After some volume is meshed (d) Tetrahedral mesh

Figure 4-4 The advancing front procedure

After creating the initial front, each triangle included in the front is connected to an appropriate node so that a well shaped tetrahedron is created. The proposed method searches for a pair of a triangle of the front and a node (a triangle-node pair in short) that satisfies two conditions: (1) the tetrahedron created by the triangle-node pair does not intersect with the front, and (2) the volume of the tetrahedron to be made by the triangle-node pair is entirely enclosed by the front. When the proposed method finds a triangle-node pair and it yields a tetrahedron satisfying the modified Delaunay criterion (explained in Section 3.3), the proposed method immediately creates a tetrahedron by connecting the triangle and the node, and the front is updated so that the front excludes the newly created tetrahedron. If no triangle-node pair yields a tetrahedron satisfying the modified Delaunay criterion, the proposed method takes the pair that yields the best tetrahedron with respect to the radius ratio (explained in Section 3.2.1) and creates a tetrahedron by connecting the triangle and the node, and the front is updated so that the front excludes the newly created tetrahedron. Thus the volume enclosed by the front shrinks after tetrahedrons are created as shown in Figure 4-4 (b). The proposed method repeats creating tetrahedrons until the volume enclosed by the front vanishes or until no triangle-node pair that satisfies the two conditions is found.

Unfortunately, some volumes may remain unmeshed at the end, typically when the remaining volume forms a shape known as a *twisted prism*, as shown in Figure 4-5, or its variation. The most primitive pattern of a twisted prism is a deformed triangular prism that satisfies three conditions: (1) its two opposite triangles are twisted so that the edges of the two opposite triangles become unparallel to each other, and (2) each quadrilateral face is subdivided into two triangles by adding a diagonal edge so that no two diagonal edges meet at a node, and (3) its shape is not convex. Any triangle-node pair that can be found in a twisted prism yields a tetrahedron that intersects either an edge of another tetrahedron or the exterior boundary of the target domain. In practice, two or more twisted prisms can be connected together and show numerous variations of unmeshable volumes. A twisted prism cannot be meshed into tetrahedrons without introducing a new node, or extra procedure is needed to break the twisted prism. If some volume remains unmeshed, the proposed method tries the following procedures.

- Undo tetrahedrons adjacent to the unmeshed volume to break a twisted prism (i.e., augment the front by the adjacent tetrahedrons) and try a different tetrahedrization. For example, if a tetrahedron has four faces t_0, t_1, t_2 and t_3, and if face t_0 is shared with the front of the unmeshed volume, the

tetrahedron is deleted, and face t_0 is deleted from the front, and faces t_1, t_2 and t_3 are added back to the front as shown in Figure 4-6. The proposed method undoes more than one tetrahedron before trying a different tetrahedrization if necessary.

- Run some more iterations of the bubble packing and mesh the domain again.

Although these extra procedures have no guarantee of termination, these extra procedures show good results for the node locations created by the bubble packing method. Finally, the volume enclosed by the front vanishes, and an anisotropic tetrahedral mesh is created as shown in Figure 4-4 (c).

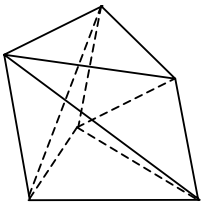
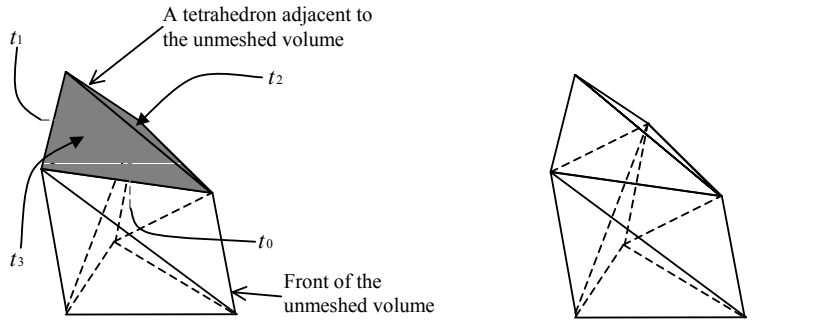


Figure 4-5 Twisted prism



(a) Front of the unmeshed volume and an adjacent tetrahedron

(b) The front after undoing the adjacent tetrahedron

Figure 4-6 Undoing a tetrahedron

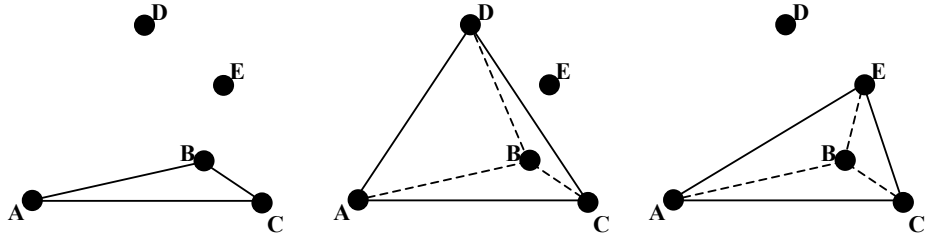
4.3.2 Discussion on the Strategy of Finding Triangle-Node Pairs

The proposed method searches triangle-node pairs based on the modified Delaunay criterion before searching triangle-node pairs based on the radius ratio, and thus the proposed method prioritizes the

modified Delaunay criterion over the radius ratio. However, the goal of the proposed method is to create well shaped tetrahedrons, i.e., tetrahedrons with smaller radius ratio. Hence, the search strategy of triangle-node pairs apparently conflicts with the goal of the proposed method. This apparently conflicting prioritization is chosen because it makes the advancing front method more likely to succeed.

The radius ratio only concerns itself with the quality of the tetrahedron and is insensitive to the node configuration around the tetrahedron, while the modified Delaunay criterion is sensitive to the nodes within the range of the circumscribed ellipsoid of the tetrahedron. As a result, if the proposed method prioritizes the radius ratio over the modified Delaunay criterion, some tetrahedrons created earlier potentially interfere with the creation of the next well shaped tetrahedron, and ill shaped tetrahedrons become inevitable. Some experiments conducted in this research indicate that the creation of an ill shaped tetrahedron is likely to yield unmeshable volume, or a twisted prism, making the advancing front method less likely to succeed. (Although many twisted prisms can be resolved by undoing some layers and trying another connection, there is no guarantee that all twisted prisms can be resolved in this way.)

For example, assume that five nodes **A**, **B**, **C**, **D** and **E** are located as shown in Figure 4-7, and triangle **ABC** is included in the front. Triangle **ABC** is an equilateral triangle, and node **D** is equidistant from nodes **A**, **B** and **C**, and the distance between node **D** and node **A** is same as a length of an edge of triangle **ABC**. Nodes **A**, **C**, **D** and **E** are co-planar, and node E is located near the center of edge **CD**, and the distance between node **A** and the center of edge **CD** is slightly shorter than the distance between nodes **A** and **E**. If the proposed method prioritizes the radius ratio over the modified Delaunay criterion, triangle **ABC** is connected to node **D** because the tetrahedron **ABCD** is an equilateral tetrahedron, and no other tetrahedron can have smaller radius ratio than tetrahedron **ABCD**. However, the creation of tetrahedron **ABCD** makes the creation of tetrahedron **BCDE** inevitable in the next step because no other node can be connected to **BCE**, and tetrahedron **BCDE** is flat and ill shaped. On the other hand, if the proposed method prioritizes the modified Delaunay criterion, triangle **ABC** is connected to node **E** and tetrahedron **ABCE** is created. Although tetrahedron **ABCE** is not as well shaped as tetrahedron **ABCD**, tetrahedron **ABED** is created in the next step, and it is not as ill shaped as tetrahedron **BCDE**. To avoid a creation of such an ill shaped tetrahedron as **BCDE**, and to make the advancing front method more likely to succeed, the proposed method prioritizes the modified Delaunay criterion over radius ratio.



(a) Node configuration and a triangle **ABC** included in the front

(b) Tetrahedron **ABCD** chosen based on the best radius ratio yielding ill shaped tetrahedron **BCED** in the next step

(c) Tetrahedron **ABCE** chosen based on the modified Delaunay criterion yielding reasonably well shaped tetrahedron **AEBD** in the next step

Figure 4-7 Comparison between a triangle-node pair based on the best radius ratio and a triangle-node pair based on the modified Delaunay criterion

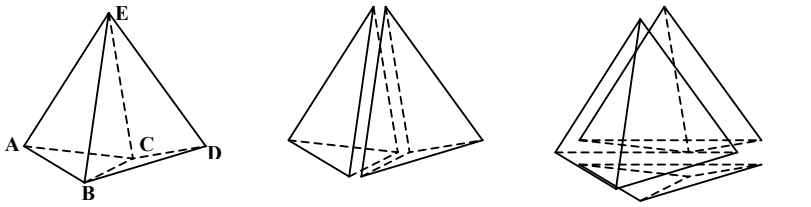
4.3.3 Quality Improvement by the Local Transformation Method

The final procedure of the meshing process improves the quality further by the local transformation method. There are two reasons why this process is needed:

- The advancing front process primarily relies on the modified Delaunay criterion, which cannot avoid creation of slivers.
- Bad elements may be created where the front meets the opposing front.

The proposed method uses Joe's local transformation method [9], which improves the mesh quality by modifying the mesh topology. This technique is based on the property of the tetrahedral mesh that there are a limited number of possible configurations of tetrahedrons for a convex volume defined by five nodes. When five nodes **A**, **B**, **C**, **D** and **E** are given, the convex volume defined by the five nodes can be subdivided into (1) two tetrahedrons **ACBE** and **BCDE** or (2) three tetrahedrons **ACBD**, **ADBE** and **ACDE** as shown in Figure 4-8. Thus, if the worst tetrahedron is included in the one of the two configurations, and if all the tetrahedrons included in the other possible configuration are better than the worst tetrahedron, the configuration that includes the worst tetrahedron is alternated to the other possible configuration. This technique is applied repeatedly, and ill shaped elements are effectively eliminated. Because this method does not alter the geometry, node locations remain unchanged.

Since the bubble packing method creates good node locations, the local transformation works effectively and eliminates virtually all ill shaped elements. Typically, all tetrahedrons of radius ratio of more than 6 are eliminated for convex domain in isotropic cases. The detailed results are presented in Section 4.5.



(a) Five nodes **A**, **B**, **C**, **D** and **E** (b) Tetrahedrons **ACBE** and **BCDE** (c) Tetrahedrons **ACBD**, **ADBE** and **ACDE**

Figure 4-8 Two possible tetrahedrization for given five nodes

4.3.4 Comparison of the Node Connecting Algorithm

Although the Delaunay triangulation method [21-23, 26, 28] is popularly used for finding node connections, the proposed method chooses the advancing front method to connect nodes because the advancing front method gives better results than the Delaunay triangulation method does when an anisotropy is given. This section explains the Delaunay triangulation method and compares the two node-connecting algorithms to justify the use of the advancing front method.

As explained in Section 2.1, the typical Delaunay triangulation method is called the point-insertion method, and the method first creates a mesh consisting of a large tetrahedron called a super tetrahedron, and the nodes are then inserted into the mesh one by one. When a node is inserted into the mesh, some tetrahedrons no longer satisfy the Delaunay criterion because their circumscribed sphere encloses the newly inserted node. Those tetrahedrons that violate the Delaunay criterion are deleted from the mesh, and the volumetric region called a cavity is created. Each triangle of the cavity is then connected to the newly inserted node so that the cavity is filled with the new tetrahedrons. The newly created tetrahedrons automatically satisfy the Delaunay criterion. This method becomes unstable when four nodes are lying on the same sphere, and thus some extra precautions, such as the one proposed by Sugihara and Inagaki [25], must be taken to avoid corruption of the process. The Delaunay triangulation method cannot avoid ill

shaped elements known as slivers, but they can be eliminated by applying some post-processes such as the local transformation method [9] as long as the nodes are well distributed.

Although the point-insertion method efficiently creates a reasonably good quality mesh in isotropic cases, the method tends to produce severely ill shaped tetrahedrons near the boundary of the domain in anisotropic cases. To expand the point-insertion method for anisotropic cases, the cavity must be created with the anisotropy taken into account. Thus, when a node is inserted, tetrahedrons that violate the modified Delaunay criterion are deleted to create a cavity. Although the anisotropy varies over the domain, the modified Delaunay criterion assumes that the anisotropy is constant over the circumscribed ellipsoid of the tetrahedron. Hence, one representative anisotropy must be chosen for each tetrahedron, and the representative anisotropy must be reasonably close to all the anisotropies within the circumscribed ellipsoid. Typically, the average of the anisotropies at four nodes of the tetrahedron, or the anisotropy at the geometric center of the tetrahedron is taken, and taking one representative anisotropy does not yield an inconsistent cavity if the anisotropy does not vary significantly over the tetrahedron. However, the anisotropy significantly varies over a tetrahedron that shares one of four nodes of the super-tetrahedron because it is typically large. As a result, the cavity becomes inconsistent if a large tetrahedron is involved. An inconsistent cavity will no longer create well shaped tetrahedrons, and the experiments performed in this research show that the tetrahedrons are typically so severely ill shaped that even the local transformation method cannot eliminate them. For example, Figure 4-9 shows a cross-section of an anisotropic mesh of a cylinder created by the anisotropic version of the point-insertion method for the same set of nodes obtained with the ellipsoidal bubble packing. The picture shows numerous long tetrahedrons around the cylinder. The anisotropy significantly varies over the long tetrahedrons, and the cavity is not created consistently if one of these long tetrahedrons is involved. As a result, significantly ill shaped elements are created near the boundary.

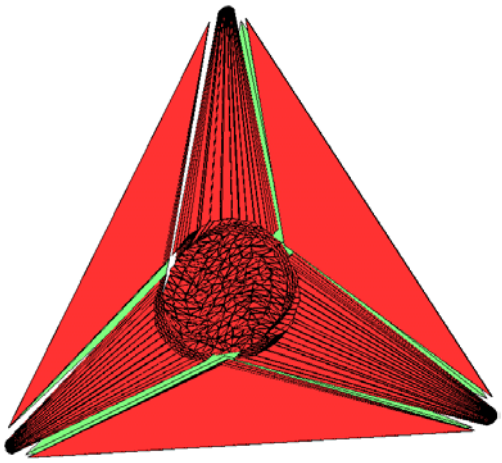


Figure 4-9 An anisotropic mesh created by the anisotropic version of the point-insertion method: Cylindrical domain is shown at the center of the mesh, and surrounding long tetrahedrons come from the super-tetrahedron

In contrast, large or long tetrahedrons like the ones created during the point-insertion method are not created during the process of the advancing front method. Although the advancing front method is not as efficient as the point-insertion method, the experiment done during this research showed that the advancing front method creates a better quality mesh than the anisotropic point-insertion method. For example, in the anisotropic case, the point-insertion method typically creates tetrahedrons of radius ratio of more than 100, while the advancing front method typically creates tetrahedrons of radius ratio of less than 10 for a convex domain. The proposed method thus uses the advancing front method for connecting nodes.

4.4 Computational Complexity

The total computational complexity of the proposed method is $O(n \log n)$, where n is the number of bubbles. The major portion of the computational time comes from the bubble packing process and the advancing front process, and the computational complexity of each of the two processes is $O(n \log n)$. The computational time for other processes, such as surface meshing and local transformation, are small compared to the bubble packing process and the advancing front process.

The computational complexity of the bubble packing process becomes $O(n^2)$ if inter-bubble forces are computed for all pairs of the bubbles exhaustively, and the proposed method applies a range-searching algorithm called kD-tree [76] to reduce the computational complexity to $O(n \log n)$. Since two too-distantly-separated bubbles do not interact with each other, the computation of the inter-bubble force between two non-interacting bubbles must be excluded from the computation of the inter-bubble forces to improve the computational efficiency. When creating a uniform isotropic mesh, the computational complexity can be reduced to $O(n)$ by applying a rectangular grid [77]. However, a rectangular grid becomes inefficient for a graded anisotropic mesh. In fact, the experiment performed in this research showed that a kD-tree performs faster than a rectangular grid for both graded and anisotropic meshes. The kD-tree is similar to the binary search tree, but the kD-tree is designed to search nodes distributed in the k-dimensional space while the binary search tree is appropriate for only one-dimensional space. The kD-tree can find neighbor bubbles of a bubble in $\log n$ steps, and inter-bubble forces are computed only between the neighbor bubbles. Since the number of neighbor bubbles of each bubble is almost constant, and there are n bubbles, and $\log n$ steps are required for each bubble to find its neighbors, total computational complexity is $O(n \log n)$. The price of building the kD-tree, and its computational complexity is also $O(n \log n)$.

Finding a triangle-node pair that yields a valid tetrahedron during the advancing front process is also a combinatorial problem, and its computational complexity easily becomes $O(n^2)$ without a sophisticated range-searching algorithm, and the proposed method again makes use of the kD-tree method to reduce the computational complexity to $O(n \log n)$. For each triangle included in the front, only limited nodes can be connected to the triangle to create a valid tetrahedron, and those candidate nodes are located within a certain range from the triangle. Thus, the proposed method makes a list of candidate nodes located within a certain range from a triangle, and only triangle-node pairs consisting of the triangle and a node within the candidate list are tested. If the nodes are well spaced, then the node to be connected to the triangle is located in the domain bound by an ellipsoid about the triangle. The size of the ellipsoid is set so that its major and minor axes are 1.5 times larger than those of the ellipsoid given by $\mathbf{M}$ at the triangle. A kD-tree efficiently finds all nodes located within the bounding box of such ellipsoid. Since the number of the

candidate nodes is almost constant for every triangle, and a candidate node list is made every time one tetrahedron is created, and $\log n$ steps are required to create a candidate node list, the total computational complexity becomes $O(m \log n)$ where m is the product of the number of tetrahedrons and the average number of candidate nodes. However, since m is nearly proportional to the number of nodes, n, $O(m \log n) = O(n \log n)$.

Some tests are performed to verify the theoretical computational complexity of the proposed method. Spherical bubbles are packed in the cylindrical domain of 20.0 diameter and 20.0 height, and nine tests are performed with different bubble radius each time. The bubble radii used in the tests are 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5 and 2.75. In each test, 800 iterations of the bubble packing are computed and the tetrahedrons are created by the advancing front method on a Pentium III 800MHz PC with 512MB RAM. The results are tabulated in Table 4-1, and the plot of the computational time of the bubble packing process vs. the number of bubbles is shown in Figure 4-10, and the plot of the computational time of the advancing front process vs. the number of tetrahedrons is shown in Figure 4-11.

The two plots indicate that the actual computational time agrees with the theoretical computational complexity of $O(n \log n)$. The plot of $y = n \log n$ becomes almost straight line for large n because the curvature of the curve, or the second derivative of the curve, is $1/n$ and dies out for large n. Since the two plots show almost straight lines, it is reasonable to assume that the actual computational time agrees with the theoretical computational complexity of $O(n \log n)$.

Table 4-1 The computational time of the bubble packing process and the advancing front process

Bubble radius	0.75	1.0	1.25	1.5	1.75	2.0	2.25	2.5	2.75
Number of bubbles	5103	2239	1256	747	505	369	281	213	165
Bubble packing	87.36sec	34.47sec	19.91sec	11.75sec	7.63sec	5.77sec	4.71sec	3.94sec	3.50sec
Number of tetrahedrons	27271	11573	6058	3520	2310	1593	1170	858	643
Advancing front	45.84sec	22.86sec	14.76sec	10.49sec	8.54sec	7.12sec	6.25sec	5.52sec	4.73sec

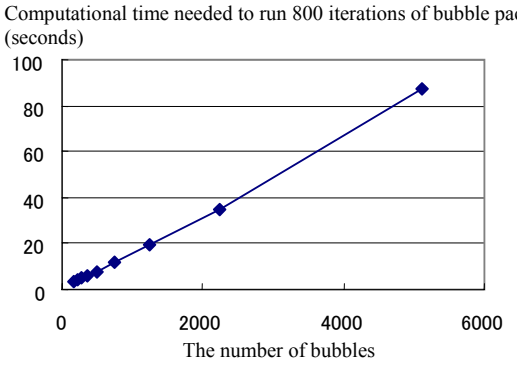


Figure 4-10 Computational time needed to run 800 iterations of bubble packing vs. the number of bubbles

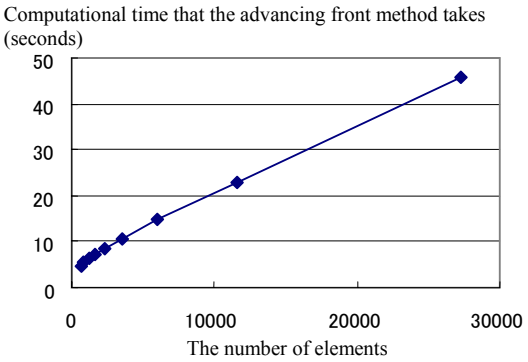


Figure 4-11 Computational time that the advancing front method takes vs. the number of elements

4.5 Results

Some experiments are conducted to test the performance of the proposed method, and five results, two isotropic meshes and three anisotropic meshes, are presented in this section. The qualities of the tetrahedrons are measured by the radius ratio explained in Section 3.2.1, and the radius ratio is computed with anisotropy taken into account in anisotropic examples as explained in Section 3.2.4. The histogram of radius ratio and the maximum radius ratio are presented for each result. The histogram of radius ratio gives an overall quality of the mesh, and if many of the elements included in the mesh are well shaped, the histogram shows a peak near radius ratio=3. The maximum radius ratio gives the quality of the worst

element of the mesh, or the *extreme quality* of the mesh, and it must be as small as possible. Pentium III 800MHz PC with 512MB RAM creates the examples.

Figure 4-12 shows an isotropic mesh of a cylinder. Figure 4-12 (a) shows packed bubbles and the cross-section of them, and Figure 4-12 (b) shows the output tetrahedral mesh and its cross-section. Since it is an isotropic example, the aspect ratios of the input anisotropy are given as $d_1 = d_2 = d_3$ all over the domain as illustrated in Figure 4-12 (c). The histogram of the radius ratio and the maximum radius ratio shown in Figure 4-12 (d) indicate that the mesh is of high quality in terms of overall quality and the extreme quality.

Figure 4-13 shows an anisotropic mesh of a cube. Figure 4-13 (a) shows the packed bubbles and the cross-section of them, and Figure 4-13 (b) shows the output tetrahedral mesh and its cross-section. The anisotropy is specified so that a tetrahedron located at (x, y) is stretched three times in the tangential direction of the sine curve, given as $y = \frac{30}{\pi}\sin\frac{\pi x}{30} + c$ (c :constant). The principal directions and aspect ratios of the input anisotropy are also illustrated in Figure 4-13 (c). The histogram of the radius ratio and the maximum radius ratio shown in Figure 4-13 (d) indicates that the mesh is of good quality in terms of both the overall quality and extreme quality.

Figure 4-14 shows an anisotropic mesh of a mechanical part. The mechanical part has a cylindrical feature on the foundation, and the anisotropy is specified so that the tetrahedrons are aligned with the cylindrical feature. The principal directions and aspect ratios of the input anisotropy are also illustrated in Figure 4-14 (c). Figure 4-14 (a) shows the packed bubbles and their cross-section, and Figure 4-14 (b) shows the output tetrahedral mesh and its cross-section. The output tetrahedral mesh shown in Figure 4-14 (b) clearly indicates that the tetrahedrons are aligned with the cylindrical feature of the mechanical part. The quality measurements are presented in Figure 4-14 (d). The histogram of the radius ratio indicates that the overall quality of the mesh is high. The maximum radius ratio of the mesh is 11.92, and the extreme quality is not as good as the ones of the previous two examples. However, the geometry of the mechanical part is more complex than the previous examples, and it is more difficult to create well shaped elements in this complex geometry than in the simple geometries of the previous two examples. The anisotropy even

makes it more difficult to make well shaped elements. The maximum radius ratio of 11.92 is reasonably good for this complex geometry.

Figure 4-15 shows an anisotropic mesh of a dinosaur. Anisotropy is specified so that (1) the tetrahedrons are aligned with the boundary surface, and (2) small tetrahedrons are created in the small features such as arms and legs. The input anisotropy is also illustrated in Figure 4-15 (c). Figure 4-15 (a) shows the packed bubbles and their cross-section, and Figure 4-15 (b) shows the output tetrahedral mesh and its cross-section. The histogram of the radius ratio and the maximum radius ratio is shown in Figure 4-15 (d). The histogram of the radius ratio indicates high over all quality. The maximum radius ratio is 12.14, and the extreme quality is not as high as the ones of the first two examples. However, the geometry of the dinosaur is much more complex than the geometry of the first two examples, and thus it is more difficult to create well shaped elements in this complex geometry than in the geometries of the first two examples. If the geometric complexity is taken into account, maximum radius ratio of 12.14 is a reasonably high quality.

Figure 4-16 shows an isotropic mesh of an airplane. Figure 4-16 (a) shows packed bubbles and Figure 4-16 (b) shows the tetrahedral mesh. In this example, some ill shaped tetrahedrons remain even after the local transformation, and these ill shaped elements are shown in Figure 4-16 (c). The geometry of the airplane has many small features such as wings and pylons, which connects a wing and an engine, and these features include sharp corners with small dihedral angles. Since the features must be preserved as much as possible, the small dihedral angles are imposed to the mesh as a geometric constraint, and the tetrahedrons near the sharp corners will have small dihedral angles, which make the tetrahedrons flat. Because it is a geometric constraint, even the local transformation method cannot remove these flat tetrahedrons. To avoid these tetrahedrons induced by thin geometry, appropriate pre-process that round-offs the sharp corners must be applied. However, the robust and efficient pre-process that can round-off the sharp corners is still a future research topic. The tetrahedrons apart from these sharp corners are all well shaped.

Table 4-2 shows the computational time required for each step of creating the five examples. Although the proposed method is not as fast as the typical commercial meshing programs, the larger computational

time is justified because the proposed method creates a high quality tetrahedral mesh, and it can take arbitrary anisotropy.

To clearly show the high quality of the meshes created by the ellipsoidal bubble packing method, Table 4-3 shows comparison between the average radius ratio of each example with the average radius ratio of a high quality reference mesh. The reference mesh is a Delaunay tetrahedral mesh created with nodes forming a hexagonal pattern. This reference was chosen because it is one of the best quality meshes that can completely fill a three-dimensional space with finite sized tetrahedrons.

To generate the reference mesh, nodes are placed in a hexagonal pattern according to the following formula:

$$x_{ijk} = \Delta x_1 + \Delta x_2 + i,$$
$$y_{ijk} = \Delta y + j\sqrt{3}/2,$$
$$z_{ijk} = k\sqrt{2/3},$$
$$\Delta x_1 = \begin{cases} 0.5 & \text{if } k \text{ is even} \\ 0 & \text{otherwise} \end{cases}$$
$$\Delta x_2 = \begin{cases} 0 & \text{if } j \text{ is even} \\ 0.5 & \text{otherwise} \end{cases}$$
$$\Delta y = \begin{cases} \sqrt{3}/6 & \text{if } k \text{ is even} \\ 0 & \text{otherwise} \end{cases}$$
$$-\infty < i, j, k < \infty \quad (i,j,k \text{ are integer numbers.})$$

The nodes are then connected to from a Delaunay tetrahedral mesh. By forming a mesh in this manner, all triangular faces parallel to the X-Y plane will be equilateral triangles. Furthermore, one in three elements will be an equilateral tetrahedron, while the other tetrahedrons will have radius ratio of 3.73; the average radius ratio is 3.49. Please note that all-equilateral tetrahedral mesh is not taken as a reference because such a mesh cannot fill entire three-dimensional space without a gap. To fill entire three-dimensional space with finite sized elements, every edge must be enclosed by elements, i.e., total dihedral angle of tetrahedrons incident to an edge must be 360 degrees. However, a dihedral angle of an equilateral tetrahedron is about 70.5 degrees, and total dihedral angle incident to an edge never becomes 360 degrees unless a non-equilateral tetrahedron is incident to the edge. Thus, this Delaunay tetrahedral mesh is one of the best quality meshes that can completely fill a three-dimensional space with finite sized tetrahedrons.

As indicated in Table 4-3, the four example meshes presented in the previous section have radius ratios that are near that of the high quality reference mesh. In the case of the cylinder shown in Figure 4-12, the ellipsoidal bubble packing method produced a mesh whose radius ratio was even better than the reference mesh. Thus, the ellipsoidal bubble packing method produces high quality tetrahedral meshes. The largest radius ratio occurred in the airplane mesh shown in Figure 4-16. This mesh had a larger radius ratio than the others because ill shaped elements were produced at the sharp edges of the input geometry. This mesh quality could be improved by pre-processing the elements near the sharp corners.

4.6 Conclusion

This chapter has presented a method for creating a high quality anisotropic tetrahedral mesh. The proposed method first creates nodes using bubble packing. Bubbles are packed inside the target geometric domain. The bubbles are then moved to the stable positions iteratively by a physically based particle simulation. Because the anisotropy is represented by a 3x3 matrix, which transforms a sphere into an ellipsoid, the proposed method packs ellipsoidal bubbles to comply with given anisotropy.

After creating nodes, the nodes are connected by the advancing front method with anisotropy taken into account. Unlike the typical advancing front method, the proposed method does not introduce new nodes during the process. The proposed method searches for a triangle-node pair that makes a high quality tetrahedron, and a tetrahedron is created by connecting a triangle and a node. Repeating this process until the front disappears, and applying local transformation method as a post process, the proposed method creates a high quality tetrahedral mesh.

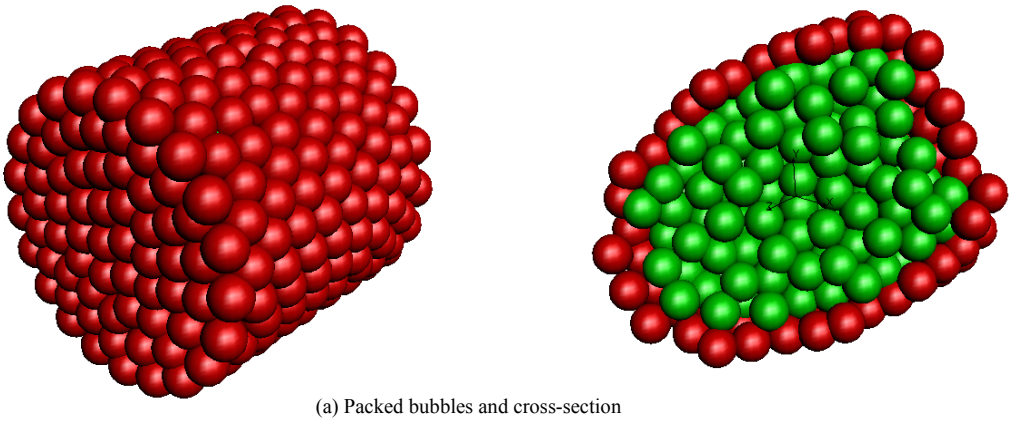
Five experimental results have been presented to demonstrate the ability of the proposed method. All examples indicate reasonably high quality, and the tetrahedrons included in the examples are aligned with the given anisotropy. The computational complexity of the proposed method is $O(n \log n)$. Although the proposed method is not as fast as the typical commercial mesh generators, the higher computational cost is justified because the proposed method creates a high quality tetrahedrons, and because the proposed method can take arbitrary anisotropy.

Table 4-2 Computational time of each step of the five examples

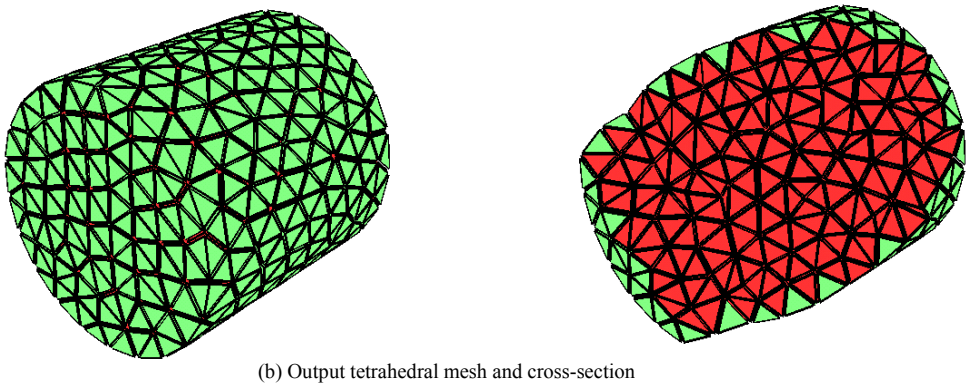
	Number of iterations	Computational time of the bubble packing process	Computational time to create surface mesh	Computational time of the advancing front process	Computational time of local transformation	Total computational time
Cylinder (Figure 4-12)	1,500	20.3 sec	2.5 sec	11.3 sec	less than 0.1 sec	34.2 sec
Cube (Figure 4-13)	1,500	42.1 sec	6.7 sec	30.5 sec	2.1 sec	81.4 sec
Mechanical part (Figure 4-14)	1,500	95.5 sec	4.3 sec	53.6 sec	10.5 sec	163.9 sec
Dinosaur (Figure 4-15)	1,500	314.7 sec	13.7 sec	109.0 sec	6.6 sec	444.0 sec
Airplane (Figure 4-16)	1,500	41.4 sec	5.3 sec	10.0 sec	11.7 sec	68.4 sec

Table 4-3 Average radius ratio of each example

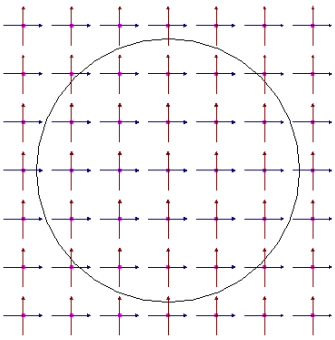
	Cylinder (Figure 4-12)	Cube (Figure 4-13)	Mechanical part (Figure 4-14)	Dinosaur (Figure 4-15)	Airplane (Figure 4-16)	Hexagonal pattern node location
Average radius ratio	3.48	3.72	3.76	3.58	5.11	3.49



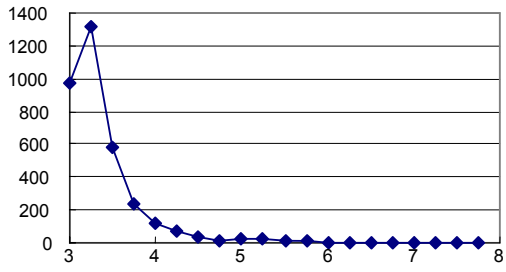
(a) Packed bubbles and cross-section



(b) Output tetrahedral mesh and cross-section



(c) Anisotropy

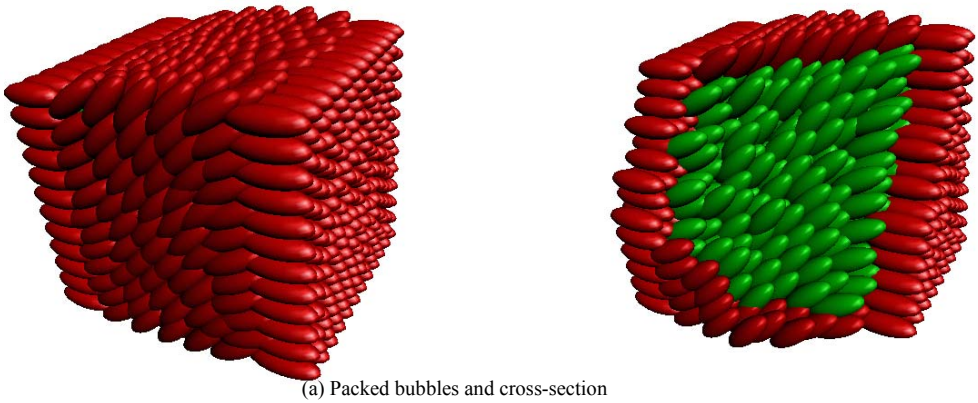


Histogram of radius ratio

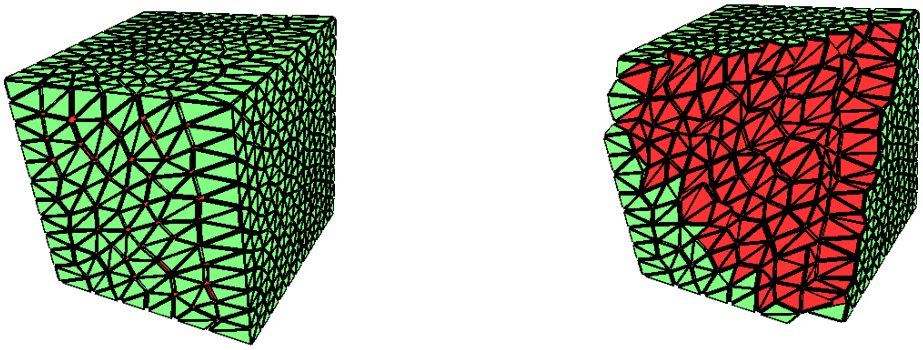
757 nodes
3,416 tetrahedrons
Maximum (worst) radius ratio=5.94

(d) Statistics and quality measurement

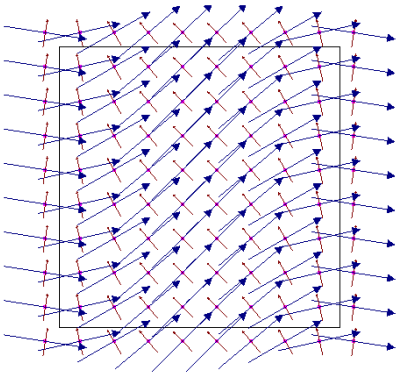
Figure 4-12 An isotropic mesh of a cylinder



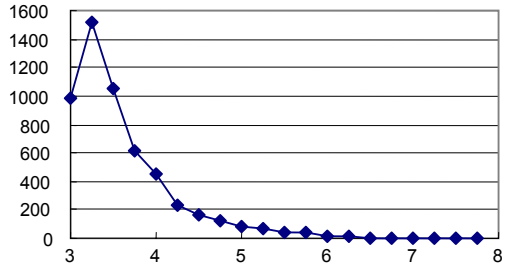
(a) Packed bubbles and cross-section



(b) Output tetrahedral mesh and cross-section



(c) Anisotropy

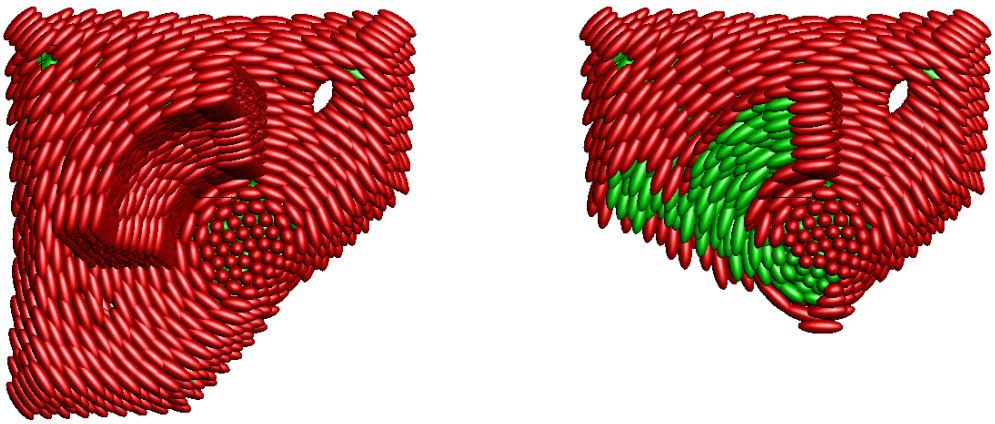


Histogram of radius ratio

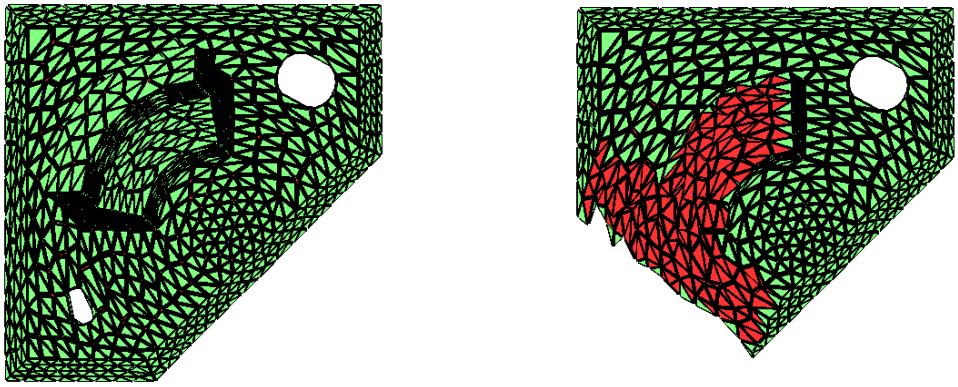
1,267 nodes
5,403 elements
Maximum (worst) radius ratio=7.71

(d) Statistics and quality measurement

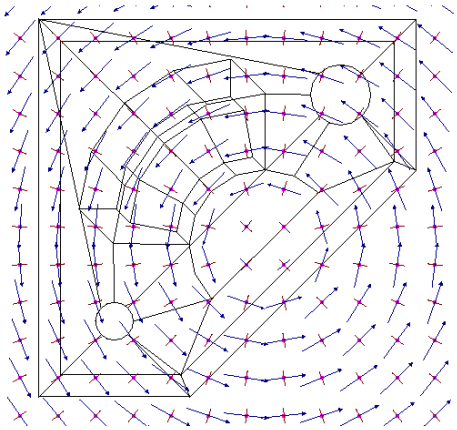
Figure 4-13 An anisotropic mesh of a cube



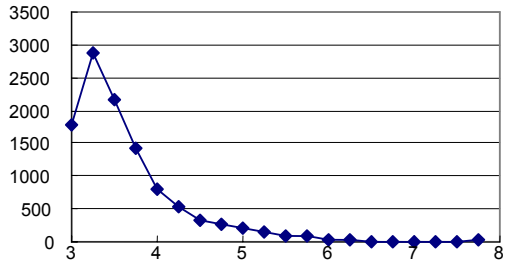
(a) Packed bubbles and cross-section



(b) Output tetrahedral mesh and cross-section



(c) Anisotropy

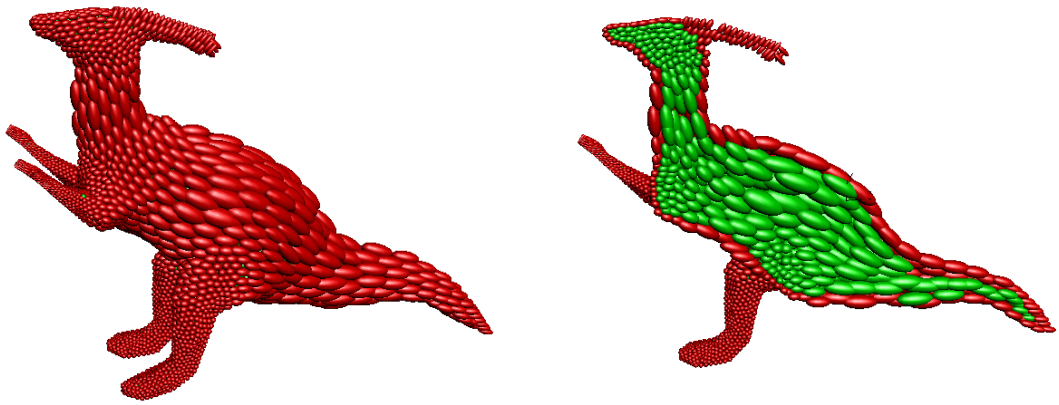


Histogram of radius ratio

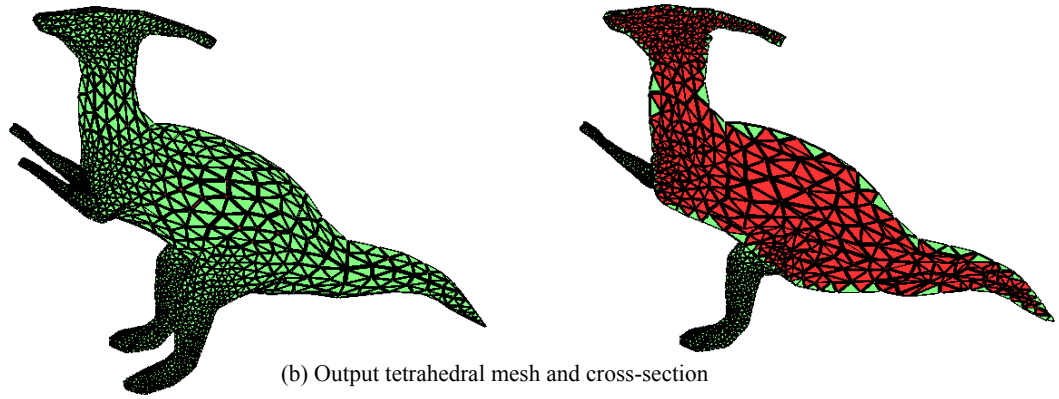
2,551 nodes
10,792 tetrahedrons
Maximum (worst) radius ratio=11.92

(d) Statistics and quality measurement

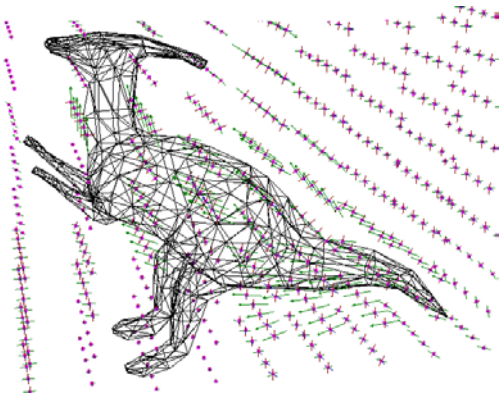
Figure 4-14 An anisotropic mesh of a mechanical part



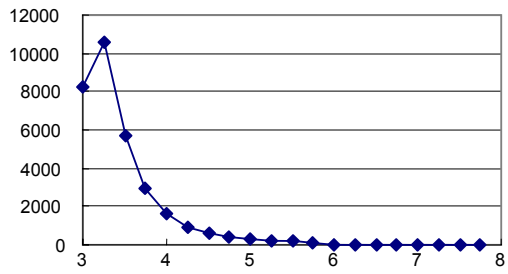
(a) Packed bubbles and cross-section



(b) Output tetrahedral mesh and cross-section



(c) Anisotropy

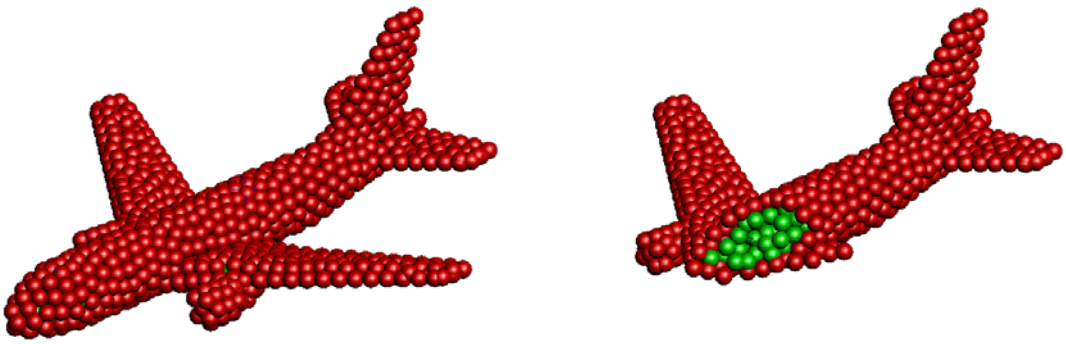


Histogram of radius ratio

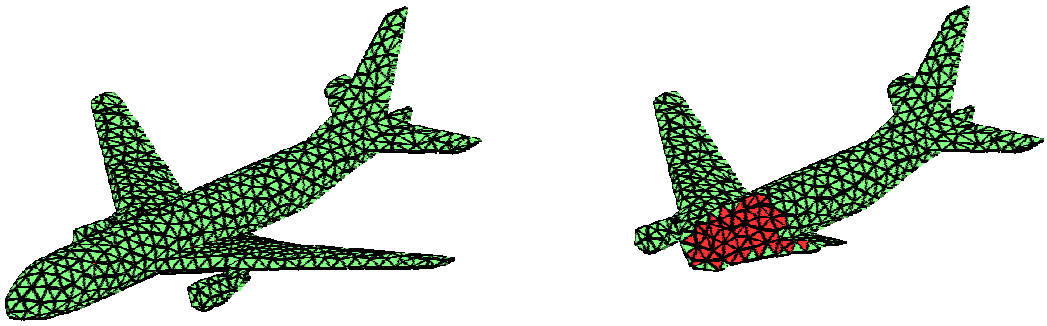
7,136 nodes
32,012 tetrahedrons
Maximum (worst) radius ratio=12.14

(d) Statistics and quality measurement

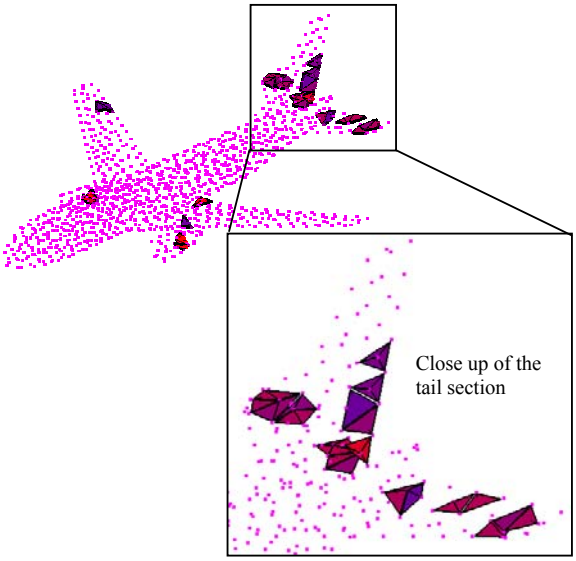
Figure 4-15 An anisotropic mesh of a dinosaur



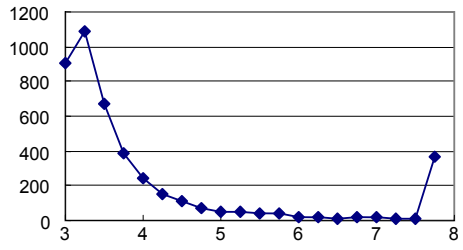
(a) Packed bubbles and cross-section



(b) Output tetrahedral mesh and cross-section



(c) Remaining ill shaped tetrahedrons



Histogram of radius ratio

1,300 nodes
4,296 tetrahedrons
Maximum (worst) Radius Ratio=105.1

(d) Statistics and quality measurement

Figure 4-16 An isotropic mesh of an airplane

Chapter 5 Hex-dominant Mesh Generation via Packing Rectangular Solid Cells

5.1 Introduction

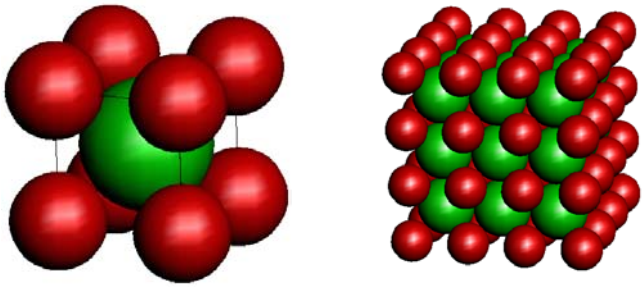
This chapter presents a new automatic hex-dominant mesh generation technique of an arbitrary 3D geometric domain. Although several hex-dominant meshing techniques are presented [36, 38, 67], they typically have some limitations; the percentage of hexahedral elements is low, or the result depends heavily on the input tetrahedral mesh; element directionality is not controlled, or the grading of element size is not precisely controlled. The proposed method overcomes these limitations by: (1) packing rectangular cells of various sizes; (2) avoiding ill-shaped elements induced by nodes located too closely together; and (3) controlling the directionality of the output hex-dominant mesh by aligning the packed rectangular solid cells.

A hex-dominant mesh combines the good aspects of a tetrahedral mesh and an all-hex mesh. A tetrahedral mesh is widely used in industry, and it has three major advantages: (1) it can be created automatically, (2) several automatic tetrahedral mesh generation techniques give reasonably high quality, and (3) it is relatively easy to control the grading of element size precisely. However, a tetrahedral mesh generally needs more elements than an all-hex mesh to gain same accuracy in the finite element analysis. Although an all-hex mesh provides a more accurate solution with fewer elements than a tetrahedral mesh, an all-hex mesh is more difficult to create, and no available technique can automatically create a high quality all-hex mesh. It is also difficult to control the grading of element size precisely in all-hex mesh. On the other hand, a hex-dominant mesh needs fewer elements than a tetrahedral mesh to attain the same accuracy in the finite element analysis, and the grading of the element size in a hex-dominant mesh can be controlled more easily than in an all-hex mesh.

The proposed method creates a hex-dominant mesh consisting of hexahedrons, prisms and tetrahedrons; and hexahedrons and prisms yield more accurate solutions than tetrahedrons in non-linear finite element analyses, since these two types of elements have higher order terms in the shape functions than a tetrahedron [12]. Among these three types of elements, a hexahedron has the highest order term, so an all-hex mesh, consisting exclusively of hexahedrons, is in great demand in industry. Creating an all-hex mesh

of an arbitrary domain is, however, a challenging problem, and no perfect solution is presented. Instead, the hex-dominant mesh is an alternative to an all-hex mesh, and it generally gives a better solution than a tetrahedral mesh [78]. Section 5.9 also shows some results of finite element analyses, which indicate that a hex-dominant mesh outperforms a tetrahedral mesh. A method that converts a hex-dominant mesh to an all-hex mesh is also presented in Chapter 6, and the conversion method requires a high quality hex-dominant mesh to create a high quality all-hex mesh. Therefore, it is important to develop a high quality hex-dominant meshing technique.

To create a hex-dominant mesh of well-shaped elements, the proposed method obtains node locations by the physical simulation of body centered cubic (BCC) structured particles. The BCC structure is seen in the crystal pattern of some natural substances, such as NaCl, illustrated in Figure 5-1 (a). If BCC structured cells are packed tightly in the domain, the cells will form a crystal pattern as shown in Figure 5-1 (b), and the centers of the cells will provide good node locations for a hex-dominant mesh.



(a) A BCC structured cell (b) A crystal pattern of BCC structure

Figure 5-1 BCC structure, which can be seen in some natural substances such as a molecule of NaCl: Closely packed BCC structured cells yield a structured grid pattern

The proposed method takes a geometric domain, desired directionality and edge length as input and creates a hex-dominant mesh in three steps. First, rectangular solid cells are packed on the boundary of and inside the target geometric domain, and the nodes are created at the centers of the cells. Next, a tetrahedral mesh is created by connecting the nodes obtained in the previous step. Last, some tetrahedrons are merged to create as many hexahedrons and prisms as possible.

The original idea of this approach is presented by Shimada et. al [6], and Itoh et al. [1]. Their method packs square cells inside a two-dimensional geometric domain to obtain good node locations for a quad-dominant mesh. The nodes are then connected by the Delaunay triangulation method, and a triangular mesh is created. Some pairs of triangles are then merged and converted to quadrilaterals. This hex-dominant mesh generation technique is an expansion of their method to the hex-dominant mesh generation.

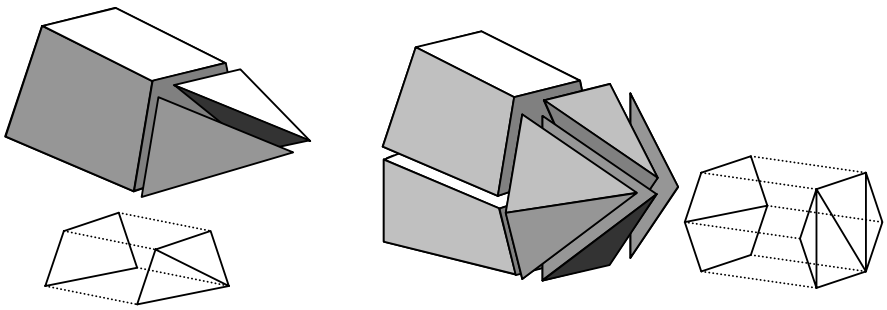
Although this hex-dominant mesh generation technique can create an anisotropic hex-dominant mesh, the isotropic hex-dominant mesh generation technique is discussed first to facilitate the explanation. The technique is later expanded to the anisotropic hex-dominant mesh generation in Section 5.11.

5.2 Interface Conformity Conditions of the Hex-Dominant Mesh

In general mesh must satisfy interface-conformity conditions: (1) every interface between elements is shared by only two elements, and (2) no interface between two elements is exposed to the exterior of the target domain or to the third element. A two-dimensional mesh easily satisfies interface conformity because there is only one type of interface --an edge-- and interface conformity is satisfied if every edge is shared by two elements. However, such conditions make hex-dominant mesh very difficult to create because there are two types of interface, a triangular face and a quadrilateral face. If the interface-conformity conditions are strictly enforced, no triangular face of a tetrahedron, a prism or a pyramid can be directly connected to a quadrilateral face of a hexahedron, a prism or a pyramid, because a triangle can hide only half of a quadrilateral; the other half of the quadrilateral is exposed either to the exterior of the target domain or to the third element, and the condition (2) is violated. Thus the interface-conformity conditions do not permit connection of a tetrahedron directly to a hexahedron, because a tetrahedron consists only of triangular faces, and a hexahedron consists exclusively of quadrilateral faces.

Interface-conformity conditions significantly limit the type of geometry that can be meshed into a hex-dominant mesh. Typically, only a limited volume of the target geometric domain can be meshed into hexahedrons. Although the remaining volume can be meshed into tetrahedrons, regions filled with hexahedrons and regions filled with tetrahedrons cannot be connected to each other because of these conditions.

One solution to the interface-conformity problem of hex-dominant mesh is to relax interface-conformity conditions and allow transition from a hexahedron or a prism to two tetrahedrons through a quadrilateral face, as shown in Figure 5-2 (a). Such a quadrilateral face, connected to two tetrahedrons, is called a non-conforming quadrilateral. Allowing non-conforming quadrilaterals eases the difficulty of creating a hex-dominant mesh. However, non-conforming quadrilaterals yield considerable error in the finite element analysis because of the difference in the geometry of a quadrilateral and the geometry of a triangle, small gaps or overlaps exists around a non-conforming quadrilateral. The geometry of a quadrilateral is a bi-linear surface, while the counterpart of a triangle is a plane. Thus, unless four nodes on the non-conforming quadrilateral are co-planer, gaps or overlaps are inevitable around the non-conforming quadrilateral. For example, the cross section of the non-conforming quadrilateral, shown in Figure 5-3, clearly visualizes the gaps.



(a) Valid transition (b) Invalid transition

Figure 5-2 Valid transition and invalid transition between hexahedrons and tetrahedrons

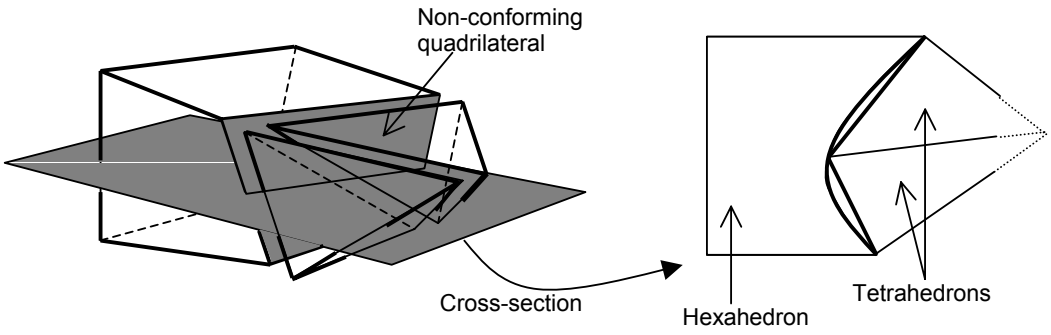


Figure 5-3 Gaps at the non-conforming quadrilateral

Several methods that reduce errors induced by non-conforming quadrilaterals are presented. Dewhirst et al. present a method that reduces the error by applying multi point constraints (MPCs) [79, 80]. Their method is applicable when tetrahedrons included in the hex-dominant mesh are 10-node tetrahedrons, which have nodes at the four corners and at the centers of the six edges of each tetrahedron. Their method also constrains a node at the center of an edge lying on the diagonal of a non-conforming quadrilateral by the function of the four nodes of the non-conforming quadrilateral. The reductions of the errors are reported in their paper. If tetrahedrons included in the hex-dominant mesh are 4-node tetrahedrons, they can be converted to 10-node tetrahedrons simply by adding nodes at the centers of the tetrahedrons' edges. Dohrmann et. al present a method that allows connection between dissimilar surfaces of the three-dimensional meshes [81]. Although the primary purpose of their method is to increase the accuracy of the contact problem, their method can also be applied to reduce errors of non-conforming quadrilaterals.

Another solution to the interface-conformity problem of the hex-dominant mesh is to introduce pyramids. A pyramid consists of one quadrilateral face and four triangular faces, and it acts as a bridge between a quadrilateral face of a hexahedron or a prism and four tetrahedrons by connecting a quadrilateral face of the pyramid to a quadrilateral face of a hexahedron or of a prism and connecting the four triangular faces of the pyramid to four tetrahedrons. Owen et. al [10] present a method that places pyramids on non-conforming quadrilaterals, resolving the non-conformity. Their method takes a hex-dominant mesh that includes non-conforming quadrilaterals as input and converts it to a strictly conformed hex-dominant mesh. This method places a pyramid on every non-conforming quadrilateral, and tetrahedrons connected to the non-conforming quadrilateral are transformed so that they conform the triangular faces of the pyramid. Since pyramids eliminate all non-conforming quadrilaterals, errors induced by gaps and overlaps around the non-conforming quadrilaterals are also eliminated. However, this method is applicable only when the finite element solver accepts pyramids.

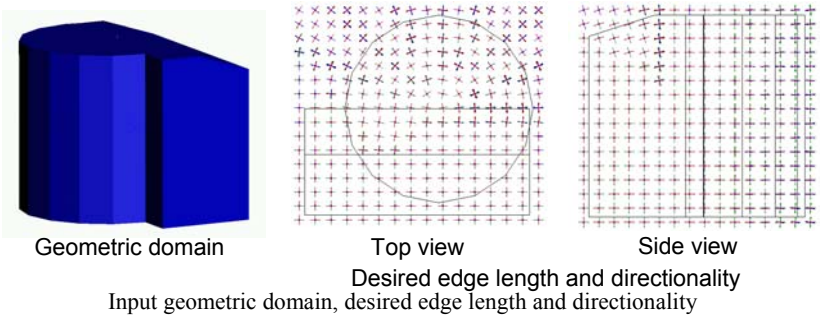
It should be emphasized that relaxed interface-conformity conditions do not allow arbitrary transition from a hexahedron or a prism to tetrahedrons through a quadrilateral face, and a non-conforming quadrilateral must be connected to two tetrahedrons. The hex-dominant mesh is invalid if a quadrilateral face is connected to more than two tetrahedrons. For example, if two hexahedrons are connected to four tetrahedrons, as shown in Figure 5-2 (b), the connection is invalid because a quadrilateral face of the top

hexahedron is connected to all four tetrahedrons, as is a quadrilateral face of the bottom hexahedron. If such an invalid transition exists, neither Dewhirst et. al's error reduction technique, nor Owen et. al's pyramid placing technique can be applied. Although Dohrmann's error reduction technique can still be applied, the effectiveness of the technique is reduced.

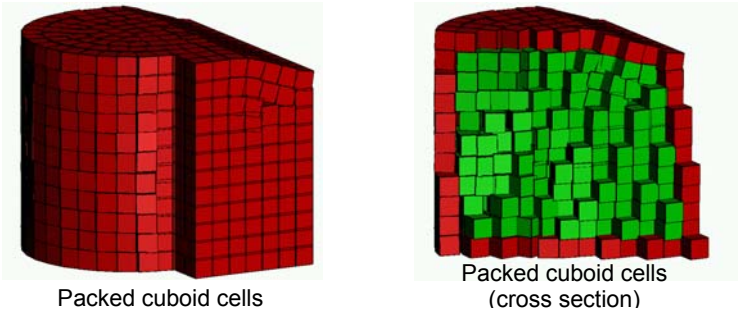
In summary, once a hex-dominant mesh with non-conforming quadrilaterals is created, (1) error reduction techniques such those of Dewhirst et. al's or Dohrmann's can be applied when the solver does not accept pyramids, or (2) Owen et al.'s method can eliminate all non-conforming quadrilaterals by introducing pyramids if the solver accepts pyramids. Thus, the method presented in this paper focuses on creating a hex-dominant mesh with non-conforming quadrilaterals.

5.3 Overview of the Proposed Method

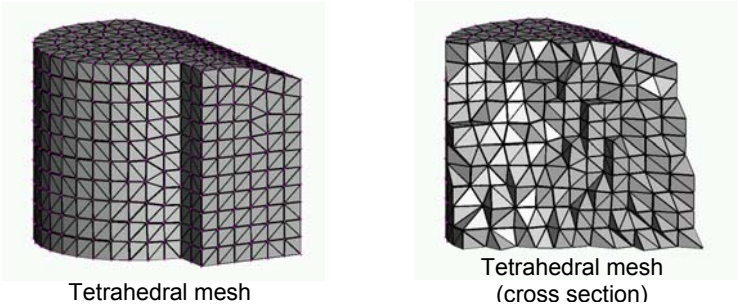
The proposed method takes a 3D geometric domain Ω, a desired directionality and edge length $\mathbf{M} = \mathbf{M}(\mathbf{x})$ (See Section 3.1 for the representation of a desired anisotropy, directionality and edge length) as input and creates a hex-dominant mesh in three steps, as illustrated in Figure 5-4. In the first step, rectangular solid cells are packed on the boundary of and inside domain Ω. Since the cells form a crystal pattern, shown in Figure 5-1 (b), the centers of the cells provide good node locations for a hex-dominant mesh. However, hexahedrons are not easy to create directly from the set of nodes obtained in the second step (See Section 5.6 for details). Instead of creating hexahedrons directly, the proposed method creates a tetrahedral mesh in the second step by the advancing front method explained in Section 4.3 and the local transformation method [9]. The tetrahedral mesh is then transformed to a hex-dominant mesh by merging tetrahedrons in the third step.



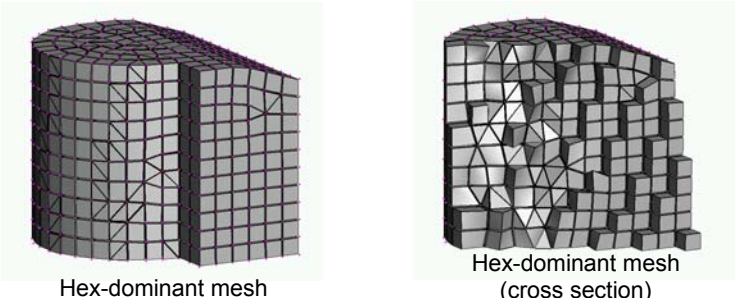
Input geometric domain, desired edge length and directionality



(a) Step 1 Packing rectangular solid cells to obtain good node locations of a hex-dominant mesh



(b) Step 2 Creating a tetrahedral mesh



(c) Step 3 Converting a tetrahedral mesh into a hex-dominant mesh

Figure 5-4 Overview of the proposed method

Although the matrix function $\mathbf{M}(\mathbf{x})$, which is given as an input, has the potential to represent anisotropy as well as directionality and edge length, it is limited for now to only represent directionality and edge length. Recall the representation of anisotropy, directionality and edge length presented in Section 3.1. The anisotropy is expressed by the aspect ratio of the lengths of three column vectors of $\mathbf{M}(\mathbf{x})$. Hence, here the lengths of three column vectors of $\mathbf{M}(\mathbf{x})$ are constrained to be equal. The constraint is later removed to expand the technique for the anisotropic hex-dominant mesh generation in Section 5.11.

Although the method assumes that the directionality is given as an input, in a special case, where a boundary-aligned hex-dominant mesh is required, directionality is computed based on the normal vector of the boundary surface and the tangential vector of the boundary edges. Since hexahedrons must be aligned with the boundary in this special case, principal vectors are constrained by two conditions as follows:

- On the boundary edges, one of three principal vectors must be parallel to the tangential direction of the edge.
- On the boundary surfaces, one of three principal vectors must be parallel to the normal direction of the surface.

To satisfy these two conditions, the first principal vector and the second principal vector at point $\mathbf{x}$, $\mathbf{u(x)}$ and $\mathbf{v(x)}$, are defined as follows:

(1) $\mathbf{u(x)}$ is equal to the unit normal vector at the point on the boundary surface that is closest to $\mathbf{x}$.

(2) Let $\mathbf{x}_e$ be a point on a boundary edge, and $\mathbf{t}_e(\mathbf{x}_e)$ be a unit tangential vector of the boundary edge at $\mathbf{x}_e$. Point $\mathbf{x}_{e0}$, $\mathbf{x}_{e0} \in \mathbf{x}_e$, is defined such that it satisfies $\cos(3\pi/4) < \mathbf{t}_e(\mathbf{x}_{e0}) \cdot \mathbf{u}(\mathbf{x}) < \cos(\pi/4)$, i.e., $\mathbf{t}_e(\mathbf{x}_{e0})$ always makes more than 45 degree and less than 135 degree angle with $\mathbf{u(x)}$. Now assume that point $\mathbf{x}_v$, $\mathbf{x}_v \in \mathbf{x}_{e0}$, minimizes the distance between $\mathbf{x}_v$ and $\mathbf{x}$, and that vector $\mathbf{t}_v$ is defined as $\mathbf{t}_v = (\mathbf{u}(\mathbf{x}) \times \mathbf{t}_e(\mathbf{x}_v)) \times \mathbf{u}(\mathbf{x})$. $\mathbf{v(x)}$ is then computed as $\mathbf{v}(\mathbf{x}) = \mathbf{t}_v / \|\mathbf{t}_v\|$.

The third principal vector $\mathbf{w(x)}$ is simply computed by taking the cross product of $\mathbf{u(x)}$ and $\mathbf{v(x)}$. Although there is apparent ambiguity in the choice of the first principal vector on the medial surface of the boundary because the distances from more than one surface to the point on the medial surface are equal, the

ambiguity only affects the region near the medial surface of the volume; it does not much affect the overall outcome of the whole meshing process.

5.4 Packing Rectangular Solid Cells on the Boundary of and Inside the Domain

In the first step of the proposed method, rectangular solid cells are packed on the boundary of and inside the geometric domain. During the process, rectangular solid cells are created in the domain and moved to stable positions by physically-based particle simulation. If an appropriate number of cells (with respect to the given directionality and edge length, $\mathbf{M}(\mathbf{x})$) are closely packed in the domain, they form a crystal pattern as shown in Figure 5-1 (b), and the centers of the cells will become good node locations for a hex-dominant mesh.

To run the simulation, an equation of motion, governing the motion of the cells, must be derived. The equation of motion is written as:

$$m\ddot{\mathbf{x}} = \sum \mathbf{f} ,$$

where m is a mass of the cell, $\mathbf{x}$ is a position of the cell, and $\sum \mathbf{f}$ is total force acting on a cell. A cell receives two types of forces; a force based on the proximity of the cells called inter-bubble force and a damping force.

The inter-bubble force is computed based on the proximity between spheres, or bubbles, creating a body-centered cubic (BCC) structure located at the center and at the eight corners of rectangular solid cells. A bubble at the center of the cell is the main bubble; bubbles at the eight corners of the cell are sub-bubbles. A sub-bubble produces forces against main bubbles, but never produces a force against other sub-bubbles; also, sub-bubbles never receive a force. This interaction is illustrated in Figure 5-5. If the length of an edge of a rectangular solid cell is d, the main bubble has a diameter of d, and thus the main bubble touches each face at its centers. A sub-bubble has a diameter of $\sqrt{3d^2} - d$, and thus it touches the main bubble at one point, also illustrated in Figure 5-6.

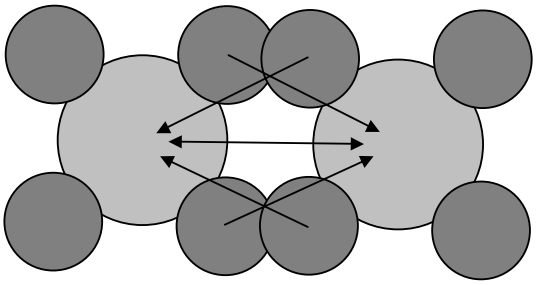


Figure 5-5 Interaction between two adjacent rectangular solid cells

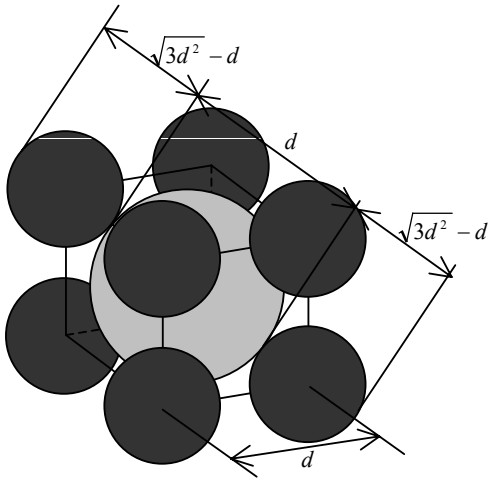


Figure 5-6 Diameters of the bubbles

An inter-bubble force acts between two adjacent bubbles, and the inter-bubble force acting on a rectangular solid cell is computed by summing up inter-bubble forces acting on the main bubble of the cell. If main bubble *A,* located at $\mathbf{x}_A$, and bubble *B*, located at $\mathbf{x}_B$, are adjacent to each other, and if r_A and r_B are the radii of *A* and *B*, the magnitude of inter-bubble force acting between two bubbles is calculated by a cubic function as:

$$f(w) = \begin{cases} k(1.25w^3 - 2.375w^2 + 1.125) & \text{if } 0 \le w \le 1.5 \\ 0 & \text{otherwise} \end{cases},$$

where $w = \dfrac{\|\mathbf{x}_B - \mathbf{x}_A\|}{r_A + r_B}$, $\|\cdot\|$ is the Euclidian norm, and k is a spring constant. If $f(w)$ is positive, two bubbles repel each other, or if it is negative, two bubbles attract each other. The direction of the inter-bubble force acting on *A* is given as a unit vector:

$$\mathbf{f}_{BA} = \frac{\mathbf{x}_A - \mathbf{x}_B}{\left\|\mathbf{x}_A - \mathbf{x}_B\right\|}.$$

As a result, the inter-bubble force acting on A, by the interaction between A and B, is calculated as:

$$\mathbf{F}_{BA} = f(w)\mathbf{f}_{BA}.$$

Since more than one bubble can be adjacent to bubble A, the total inter-bubble force acting on A is computed as follows:

$$\mathbf{F} = \sum_i \mathbf{F}_{iA},$$

where i is ith bubble that is adjacent to A.

By defining a mass of a rectangular solid cell m and damping force $-c\dot{\mathbf{x}}$, the equation of motion of the cell is written as:

$$m\ddot{\mathbf{x}} = \mathbf{F} - c\dot{\mathbf{x}},$$

where $\mathbf{x}$ is the position of the main bubble, which is equal to the position of the rectangular solid cell. The equation is solved by an iterative numerical integration scheme such as Euler's method or the 4th order Runge-Kutta method. In each iteration, the orientations of the rectangular solid cells are updated based on the vector field generated in the first step of the process.

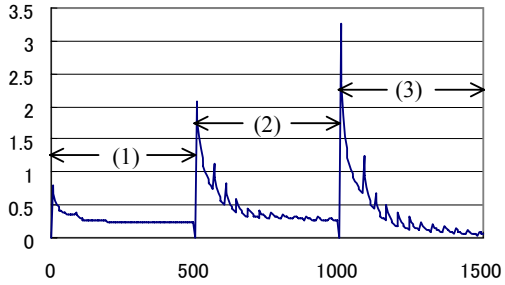
It should be noted that the locations of the cells give good node locations for a hex-dominant mesh only if the appropriate number of cells are packed in the domain, and thus some cells should be added or deleted in the domain during the simulation in order to adjust the number of cells. Since the precise number of cells is often impossible to compute when the input domain is complex, the number of cells must be adjusted adaptively during the simulation based on the population density of the cells. In the current implementation, the program finds regions where cells are too sparse or too crowded, based on the distances between adjacent cells (the cells that give non-zero $f(w)$), and adds some cells to sparse regions and deletes some cells from crowded regions.

Because this simulation mimics the formation of a BCC crystal pattern, shown in Figure 5-1 (b), the centers of the rectangular solid cells tend to form a structured orthogonal grid pattern, which is appropriate

for a hex-dominant mesh. Although the pattern will not be well structured in a region where $\mathbf{M}(\mathbf{x})$ changes drastically, those cases are usually local, and most node locations are well structured.

5.5 Convergence of the Rectangular Solid Cells

The experiments performed in this research show that the rectangular solid cells packed in the target geometric domain converge to a stable configuration in less than 1,500 iterations in general. The cells packed on the edges of the geometric domain usually converge to a stable configuration in less than 500 iterations, and another 500 iterations or less are required to stabilize the cells packed on the faces of the geometric domain, and another 500 iterations or less are required for the cells packed interior of the domain to reach stability. The plot of the average speed of the cells against the number of iterations is shown in Figure 5-7 to show the convergence. The average speed is measured while packing cells on the boundary of and inside the geometric domain of a mechanical part with a cylindrical feature shown in Figure 5-15 (a). The cells move fast at the start of packing of the cells on the edges, and at the start of packing of the cells on the faces, and at the start of packing of the cells interior of the domain, and the average speed of the cells quickly drops after some iterations.



(1) Packing cells on the edges of the domain
(2) Packing cells on the faces of the domain
(3) Packing cells interior of the domain

Figure 5-7 Plot of the average speed of the rectangular solid cells against the number of iterations

Although the average speed of the cells does not drop to zero, the cells do not move significantly after the average speed drops to a certain level. The system of the cells is a second order system because the motions of the cells are governed by the second order ordinary differential equation; and the speed of the

bubbles decays exponentially, but small vibrations of the cells remain indefinitely. As a result, certain amount of average speed of the cells is observed even after the cells stop showing significant movement. However, since the vibration is small, the configuration of the cells is stable, and it gives good node locations for a hex-dominant mesh.

Since the number of iterations required for stabilizing the cells is nearly constant, the computational complexity of the packing process depends on the computational complexity of an iteration of the motions of the cells. The complexity of the computation of an iteration of the motions of the cells is $O(n \log n)$ where n is the number of the cells as discussed in Section 4.4

5.6 Generating a Tetrahedral Mesh

After the nodes are created by the rectangular solid cell packing method, the nodes are connected to form a tetrahedral mesh. The purpose of creating a tetrahedral mesh, instead of creating a hex-dominant mesh directly, is to avoid potential numerical instability caused by the computation of intersections between the bi-linear surfaces that are included in hexahedrons and prisms. No two elements should intersect during the mesh generation process because any intersection between elements will make the mesh invalid. (Recall that the advancing front technique requires intersection-checking.) However, checking the intersection between elements that include quadrilateral faces makes the computation potentially unstable. The geometry of a quadrilateral face is a bi-linear surface, and a non-linear equation solving scheme, such as Newton's method, is needed to find an intersection between bi-linear surfaces. Such a non-linear equation solving scheme may become unstable, making the result of the intersection check inaccurate. In the worst case, two elements may intersect, and this makes the entire mesh invalid.

On the other hand, a tetrahedron has only triangular faces. Since the geometries of the triangular faces are planar, an intersection between tetrahedrons can be detected easily by solving linear equations. Hence, creating tetrahedrons is easier than creating hexahedrons and prisms in terms of robustness. In addition, if no two tetrahedrons are intersecting, merging some tetrahedrons to create hexahedrons and prisms will not create a new intersection. It is thus reasonable to create a hex-dominant mesh in two steps: (1) creating a tetrahedral mesh, and (2) merging some tetrahedrons to create hexahedrons and prisms.

A tetrahedral mesh is created by the advancing front method, presented in Section 4.3. After the tetrahedral mesh is created, the mesh quality is improved by a method called local transformation [9].

Another possible option for creating a tetrahedral mesh is to apply the Delaunay triangulation method [23, 26]. The Delaunay triangulation method is widely used to create a tetrahedral mesh. However, edges in the tetrahedral mesh created by the Delaunay triangulation method often intersect with the original boundary unless the geometry is convex, and those intersections must be removed in a boundary recovery post-process, which often adds new nodes. Since the nodes created in the rectangular solid cell packing are well located, adding a new node can worsen the quality of the output hex-dominant mesh. The advancing front method is thus suitable for the work presented in this paper.

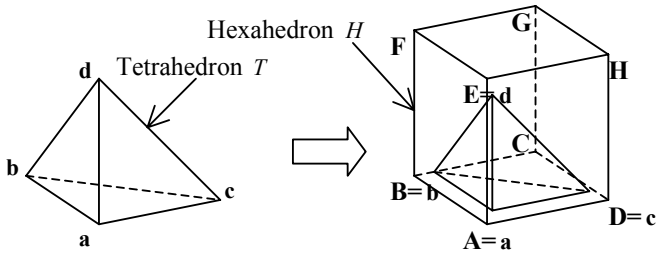
5.7 Converting a Tetrahedral Mesh to a Hex-Dominant Mesh

After the tetrahedral mesh is created, the proposed method merges some tetrahedrons to create hexahedrons and prisms. This method first creates a list of node combinations that form hexahedrons; the combinations are then sorted by the shape quality of the hexahedrons so that the node combinations yielding the superior quality hexahedrons are listed first. Tetrahedrons are merged and converted to hexahedrons individually based on the list. After exhausting all the possible hexahedrons, the proposed method creates a list of node combinations that create prisms. The combinations are then sorted by the quality of the prisms so that the combination that yields the best quality prism comes first and the worst quality last. Some tetrahedrons are then merged and converted to prisms based on the list. The search for possible combinations relies on the topological information of the tetrahedral mesh; a topological data structure is needed to perform the search efficiently.

During this process, the quality of a hexahedron is measured by minimum scaled Jacobian. (See Section 3.2.2 for the definition of scaled Jacobian and minimum scaled Jacobian.) Scaled Jacobian of element e at node $\mathbf{n}$ is denoted as $Js_e(\mathbf{n})$, and minimum of $Js_e(\mathbf{n}_i)$, where $\mathbf{n}_i$ is ith node of element e, measures the quality of element e and is called minimum scaled Jacobian.

5.7.1 Finding Node Combinations that Create Hexahedrons

The proposed method makes a list of possible node combinations that can each be converted to a hexahedron. Each entry of the list stores eight nodes that will become eight nodes of a hexahedron. To facilitate the explanation, a hexahedron to be created is denoted as H, and the eight corners of hexahedron H are denoted as **A** through **H**, and **ABCD**, **ADHE**, **BAEF**, **CBFG**, **DCGH** and **FEHG** are the six quadrilateral faces of hexahedron H (see Figure 5-8). Nodes in the tetrahedral mesh are assigned to **A** through **H** to make a node combination.



(a) A tetrahedron in the tetrahedral mesh

(b) Assigning the four nodes of the tetrahedron

Figure 5-8 Finding node combinations based on Pattern 1

The proposed method searches node combinations based on the two possible tetrahedron configurations: (Pattern 1) three edges of a tetrahedron become three edges of a hexahedron that share a common node, as shown in Figure 5-8 (b), and (Pattern 2) six tetrahedrons share an edge that will become a diagonal of the hexahedron, as shown in Figure 5-9 (b).

The proposed method searches node combinations of Pattern 1 based on the assumption that four nodes of a tetrahedron become **A**, **B**, **D** and **E** of hexahedron H, and nodes to be assigned for **C**, **F**, **G** and **H** will be found by the edges of the tetrahedral mesh. Let T be a tetrahedron, and $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ are the four nodes of tetrahedron T, satisfying $J_{ST}(\mathbf{a}) \geq J_{ST}(\mathbf{b}), J_{ST}(\mathbf{c}), J_{ST}(\mathbf{d})$ and $(\mathbf{c}-\mathbf{a})\times(\mathbf{b}-\mathbf{a})\cdot(\mathbf{d}-\mathbf{a})>0$, as shown in Figure 5-8 (a). The proposed method assumes that three edges of T, $\mathbf{a}-\mathbf{b}$, $\mathbf{a}-\mathbf{c}$ and $\mathbf{a}-\mathbf{d}$, will become three edges of hexahedron H, as shown in Figure 5-8 (b). If node $\mathbf{a}$ is assigned to **A**, nodes $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ can be assigned to **B**, **D** and **E**, and edges $\mathbf{a}-\mathbf{b}$, $\mathbf{a}-\mathbf{c}$ and $\mathbf{a}-\mathbf{d}$ will become **AB**, **AD** and **AE**. Although ($\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$) can also be assigned to (**D**, **E**, **B)** or (**E**, **B**, **D)**, the result will be equivalent. The candidate

nodes for **F** are then limited to the nodes directly connected to **b** and **d**; the candidate nodes for **H** are limited to the nodes directly connected to **c** and **d**; and the candidate nodes for **C** are limited to the nodes directly connected to **b** and **c**. One of the candidates for **F** is denoted as **p**, for **C** is denoted as **q**, and for **H** is denoted as **r**. For each choice of **p**, **q** and **r**, the candidates for **G** are the nodes directly connected to nodes **p**, **q** and **r**, and one of the candidates for **G** is denoted as **s**. To avoid degeneracy, it is important to assure that no two nodes of **a**, **b**, **c**, **d**, **p**, **q**, **r**, and **s** are identical. The proposed method checks every possible choice of **p**, **q**, **r**, and **s**, and if node combination (**A**, **B**, **C**, **D**, **E**, **F**, **G**, **H**)=(**a**, **b**, **q**, **c**, **d**, **p**, **s**, **r**) makes a minimum scaled Jacobian of H positive, the node combination is added to the node combination list. It must be noted that a scaled Jacobian $Js_T(\mathbf{a})$ is inherited to $Js_H(\mathbf{A})$, and since node **a** is chosen so that it satisfies $Js_T(\mathbf{a}) \geq Js_T(\mathbf{b}), Js_T(\mathbf{c}), Js_T(\mathbf{d})$, the largest possible scaled Jacobian of tetrahedron T, $Js_T(\mathbf{a})$, is inherited to hexahedron H. The proposed method searches node combinations of Pattern 1 for every tetrahedron, and all node combinations found by the search are added to the node combination list.

Although the program searches node combinations exhaustively for each tetrahedron, the computational time to search combinations for a tetrahedron is nearly constant because the proposed method only visits nodes less than three edges away from the tetrahedron. The number of nodes that are less than three edges away from the tetrahedron is almost constant. The computational complexity for whole tetrahedral mesh is thus an order of n, where n is number of tetrahedrons included in the tetrahedral mesh.

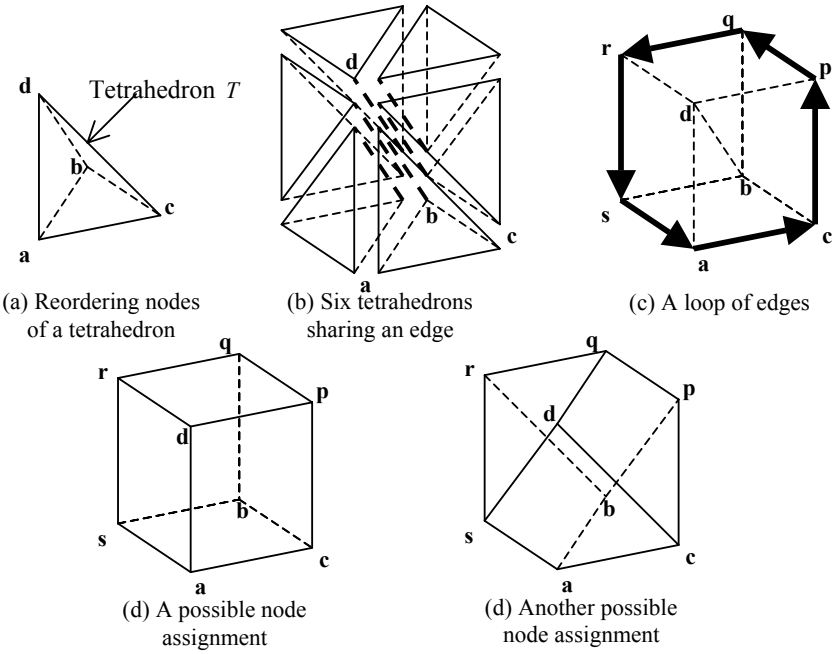


Figure 5-9 Finding node combinations based on Pattern 2

The proposed method searches node combinations of Pattern 2 based on the assumption that the longest edge of a tetrahedron becomes a diagonal of a hexahedron, and the assumption will narrow the possible node combinations to two, and the combination yielding a better minimum scaled Jacobian will be added to the node combination list. Let T be a tetrahedron, and **a**, **b**, **c** and **d** be nodes of tetrahedron T, satisfying $\|\mathbf{b}-\mathbf{d}\| \geq \|\mathbf{a}-\mathbf{b}\|, \|\mathbf{a}-\mathbf{c}\|, \|\mathbf{a}-\mathbf{d}\|, \|\mathbf{b}-\mathbf{c}\|, \|\mathbf{c}-\mathbf{d}\|$ and $(\mathbf{c}-\mathbf{a})\times(\mathbf{b}-\mathbf{a})\cdot(\mathbf{d}-\mathbf{a})>0$, as shown in Figure 5-9 (a). The proposed method assumes that edge **b - d** will become a diagonal of the hexahedron, because the diagonal of a well-shaped hexahedron is always longer than any edges of the hexahedron and any diagonals of the faces of the hexahedron. If the number of tetrahedrons sharing edge **b - d** is not six, the method moves on to the next T and begins again. If the number of tetrahedrons sharing edge **b - d** is six, the proposed method then makes a list of six tetrahedrons sharing edge **b - d**, as shown in Figure 5-9 (b). The edges included in the six tetrahedrons and not connected to nodes **b** and **d** make a loop as shown in Figure 5-9 (c), and four nodes as well as nodes **a** and **c** are included in the loop and denoted as **p**, **q**, **r** and **s**, and the order of the nodes in the loop is (**a**, **c**, **p**, **q**, **r**, **s**). At this point, there are still two possible node combinations. One is (**A**, **B**, **C**, **D**, **E**, **F**, **G**, **H**)=(**a**, **s**, **b**, **c**, **d**, **r**, **q**, **p**) as shown in Figure 5-9 (d), and

the other is (**A**, **B**, **C**, **D**, **E**, **F**, **G**, **H**)=(**a** , **b** , **p** , **c** , **s** , **r** , **q** , **d**) as shown in Figure 5-9 (e). Both combinations are topologically correct, and hence the geometric quality must be taken into account to choose one of the two possible combinations. The proposed method computes a minimum scaled Jacobian of the hexahedrons created by both node combinations, and the one that yields a larger minimum scaled Jacobian is added to the node combination list. In the case shown in Figure 5-9, the combination shown in Figure 5-9 (d) clearly yields better quality, and thus it will be taken. The proposed method searches node combinations of Pattern 2 for every tetrahedron, and all the node combinations found by the search are added in the node combination list.

5.7.2 Creating Hexahedrons Based on the Node Combination List

After the node combination list is created, the entries of the list are sorted by the minimum scaled Jacobian of hexahedrons to be created so that the entry of the largest minimum scaled Jacobian comes first, and the entry of the smallest minimum Jacobian last. The proposed method then attempts to create hexahedrons based on the sorted node combination list.

For each node combination consisting of eight nodes, denoted as (**A**, **B**, **C**, **D**, **E**, **F**, **G**, **H**), the proposed method attempts to create hexahedron H , which consists of the eight nodes, by merging tetrahedrons that each consist of four of the eight nodes. The node combination, however, is discarded if it does not satisfy the following two conditions: (1) creating hexahedron H does not induce gaps or overlaps, and (2) quadrilateral faces of H are compatible with quadrilateral faces of adjacent hexahedrons.

Condition (1) is not satisfied when some tetrahedrons that were using four of the eight nodes no longer exist because they already became a part of another hexahedron. To enforce condition (1), the proposed method creates a polyhedron by merging tetrahedrons, each consisting of four of the eight nodes, as shown in Figure 5-10. If the number of triangles included in the polyhedron is not 12, or if every quadrilateral face of hexahedron H does not correspond to a unique pair of adjacent triangles of the polyhedron, the node combination is discarded. Since no geometric computation is carried out to enforce Condition (1), it is not necessary to check intersections between bi-linear surfaces, which potentially makes the computation unstable.

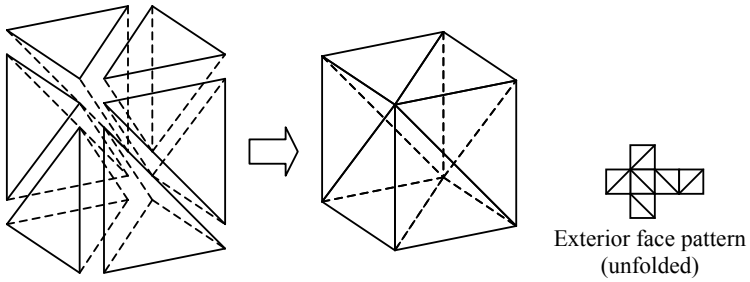


Figure 5-10 A set of tetrahedrons to be merged to form a hex, and a polyhedron made of the exterior of the set of the tetrahedrons

Condition (2) is not satisfied when a quadrilateral face of hexahedron H shares three of four nodes of a quadrilateral face of a hexahedron already-created, as shown in Figure 5-11. To enforce condition (2), the proposed method checks six quadrilateral faces of hexahedron H, **ABCD**, **ADHE**, **BAEF**, **CBFG**, **DCGH** and **FEHG**. If a quadrilateral face of hexahedron H shares three of four nodes of a quadrilateral face of the adjacent hexahedron, the node combination is discarded.

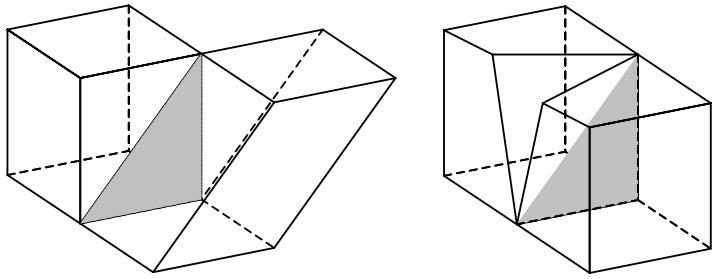


Figure 5-11 Examples of two quadrilaterals sharing three nodes, which violate the interface-conformity conditions

The two conditions also prevent the creation of an invalid transition between a hexahedron and tetrahedrons, such as shown in Figure 5-2 (b), because a quadrilateral face of hexahedron H is connected either to two triangles, to a quadrilateral, or to the exterior of the domain, if the two conditions are satisfied. If the two conditions are satisfied, the proposed method deletes the tetrahedrons that each consist of four of the eight nodes of the node combination, and hexahedron H is added to the mesh.

It must be noted that if an ill-shaped tetrahedron --known as a sliver-- consists of four nodes of a node combination, as shown in Figure 5-12, the sliver will be deleted when a hexahedron is created from the node combination. Although slivers are mostly eliminated by local transformation [9], some slivers may

remain even after the local transformation, and a remaining sliver typically consists of four nodes that will become the quadrilateral face of a hexahedron or a prism. The conversion process thus eliminates many of the remaining slivers, and the quality of the mesh is improved.

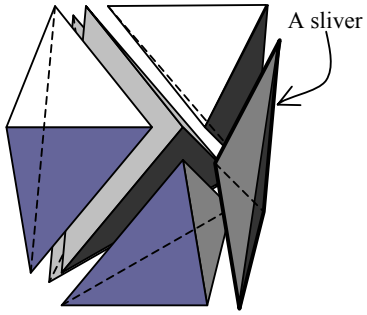
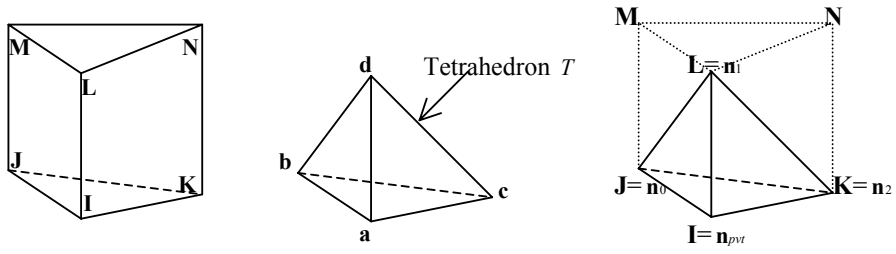


Figure 5-12 A sliver (very flat tetrahedron) included in a node combination: The sliver will be deleted when the tetrahedrons are merged and converted to a hexahedron

5.7.3 Finding Node Combinations for Prisms

After creating hexahedrons, the proposed method makes a list of node combinations that can each be a prism; each entry contains six nodes capable of becoming nodes of a prism. To elucidate: the prism to be created is denoted as P, and nodes of P are denoted as (**I**, **J**, **K**, **L**, **M**, **N**). **IJK** and **MLN** are the two triangles of the prism, and **ILMJ**, **IKNL** and **IMNK** are the three quadrilaterals of the prism, as shown in Figure 5-13 (a). Since some tetrahedrons are already converted to hexahedrons, the mesh that was a tetrahedral mesh is no longer a tetrahedral mesh, but it is already a hex-dominant mesh with no prism.



(a) A prism to be created

(b) A tetrahedron in the tetrahedral mesh

(c) Node assignments for **I**, **J**, **K** and **L**

Figure 5-13 Finding node combinations for a prism

The proposed method assumes that four nodes of a tetrahedron will become **I**, **J**, **K** and **L**, and candidate nodes to be assigned to **M** and **N** are searched by the edges in the mesh. Let T be a remaining tetrahedron, and $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ be four nodes of tetrahedron T, as shown in Figure 5-13 (b). And node $\mathbf{n}_{pvt}$ is one of $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$, and $\mathbf{n}_0$, $\mathbf{n}_1$ and $\mathbf{n}_2$ are the other three nodes of the tetrahedron, satisfying $(\mathbf{n}_1 - \mathbf{n}_0) \times (\mathbf{n}_2 - \mathbf{n}_0) \cdot (\mathbf{n}_{pvt} - \mathbf{n}_0) > 0$. There are four possible choices of $\mathbf{n}_{pvt}$, and for each choice of $\mathbf{n}_{pvt}$ there are three different choices of $\mathbf{n}_0$, $\mathbf{n}_1$ and $\mathbf{n}_2$. For example, if $\mathbf{n}_{pvt} = \mathbf{a}$, ($\mathbf{n}_0$, $\mathbf{n}_1$, $\mathbf{n}_2$) can be ($\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$), or ($\mathbf{c}$, $\mathbf{d}$, $\mathbf{b}$), or ($\mathbf{d}$, $\mathbf{b}$, $\mathbf{c}$). For each choice of $\mathbf{n}_{pvt}$, $\mathbf{n}_0$, $\mathbf{n}_1$ and $\mathbf{n}_2$, the proposed method assumes **I**= $\mathbf{n}_{pvt}$, **J**= $\mathbf{n}_0$, **K**= $\mathbf{n}_1$, and **L**= $\mathbf{n}_2$ as shown in Figure 5-13 (c), and nodes to be assigned to **M** and **N** are searched by the edges connected to $\mathbf{n}_0$, $\mathbf{n}_1$ and $\mathbf{n}_2$. The candidate nodes for **M** are limited to the nodes directly connected to $\mathbf{n}_0$ and $\mathbf{n}_1$, and the candidate nodes for **N** are the nodes directly connected to $\mathbf{n}_1$ and $\mathbf{n}_2$. One of the candidate nodes for **M** is denoted as $\mathbf{p}$, and one of the candidate nodes for **N** is denoted as $\mathbf{q}$. For every pair of $\mathbf{p}$ and $\mathbf{q}$ that is directly connected to each other, the proposed method computes the minimum scaled Jacobian of the prism made by node combination (**I**, **J**, **K**, **L**, **M**, **N**)=($\mathbf{n}_{pvt}$, $\mathbf{n}_0$, $\mathbf{n}_1$, $\mathbf{n}_2$, $\mathbf{p}$, $\mathbf{q}$). If the minimum scaled Jacobian is positive, the node combination is added to the node combination list. The proposed method searches node combinations for every remaining tetrahedron, and all node combinations found by the search are added to the node combination list.

5.7.4 Creating Prisms Based on the Node Combination List

After the node combination list is created, the entries of the list are sorted by the minimum scaled Jacobian of prisms to be created so that the entry of the largest minimum scaled Jacobian comes first, and the entry of the smallest minimum Jacobian last. The proposed method then attempts to create prisms based on the sorted node combination list.

For each node combination, consisting of six nodes denoted as (**I**, **J**, **K**, **L**, **M**, **N**), the proposed method attempts to create prism P, which consists of the six nodes, by merging tetrahedrons each consisting of four of the six nodes. The node combination, however, is discarded if it does not satisfy the following two

conditions: (1) the creation of prism P does not induce gaps or overlaps, and (2) the quadrilateral faces of P are compatible with quadrilateral faces of adjacent hexahedrons and prisms.

Condition (1) is not satisfied when some tetrahedrons that were using four of the six nodes no longer exist because they are already part of another hexahedron or prism. To enforce condition (1), the proposed method creates a polyhedron by merging the tetrahedrons that each consist of four of the six nodes as shown in Figure 5-14. If the number of triangles included in the polyhedron is not 8, or if every quadrilateral face of prism P does not correspond to a unique pair of adjacent triangles of the polyhedron, the node combination is discarded.

Condition (2) is not satisfied when a quadrilateral face of prism P shares three of four nodes of a quadrilateral face of hexahedron or prism that has already been created, as shown in Figure 5-11. To enforce condition (2), the proposed method checks three quadrilateral faces of prism P, **JILM**, **KJMN** and **IKNL**. If a quadrilateral face of prism P shares three of four nodes of a quadrilateral face of the adjacent hexahedron or a prism, the node combination is discarded.

If the two conditions are satisfied, the proposed method deletes the tetrahedrons consisting of four of the six nodes each of the node combination, and prism P is added to the mesh.

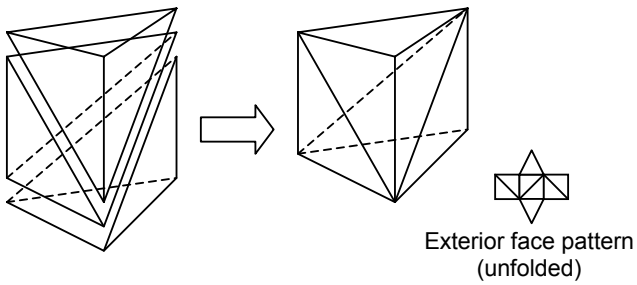


Figure 5-14 A set of tetrahedrons to be merged to form a prism, and a polyhedron made of the exterior of the set of the tetrahedrons

5.8 Results

This section presents some results of applying the proposed method. The percentages of hexahedrons, prisms and tetrahedrons are presented for each example in terms of the number of elements, and of volumes. The quality of the hexahedrons and prisms are presented by the histogram of the minimum scaled Jacobian,

and the quality of the tetrahedrons is presented by the histogram of radius-ratio, which is the radius of circumscribed sphere divided by the radius of the inscribed sphere. For an equilateral tetrahedron, which is a perfectly well-shaped tetrahedron, the radius-ratio becomes 3.0, and it grows as the shape of the element worsens.

In each radius ratio histogram, a dashed line indicates radius ratio of 3.88, which is an average radius ratio of tetrahedrons included in the pattern shown in Figure 5-8. Since the rectangular solid cell packing tends to create a structured grid pattern, tetrahedrons are likely to form the patterns shown in Figure 5-8 and Figure 5-9. The average radius ratio of tetrahedrons included in the pattern shown in Figure 5-9 is 4.18. Thus, if the remaining tetrahedrons have radius ratio that are less than 3.88, it indicates that they are reasonably high quality elements that can be created with the rectangular solid cell packing. Since the radius ratio histograms show a peak left of 3.88, the remaining tetrahedrons are of reasonably high quality.

Figure 5-15 (d) and Figure 5-15 (e) show a hex-dominant mesh of a mechanical part with a cylindrical feature, and Figure 5-15 (a) shows the input geometric domain and the directionality. The directionality shown in Figure 5-15 (a) is computed from the boundary of the geometry by the method presented in Section 5.3. The packing of rectangular solid cells is shown in Figure 5-15 (b) and Figure 5-15 (c). The picture clearly shows that hexahedrons on the boundary are aligned with the given directionality. The histogram of the radius-ratio of tetrahedrons, the scaled Jacobian of hexahedrons and the scaled Jacobian of prisms are shown in Figure 5-15 (f), Figure 5-15 (g), and Figure 5-15 (h). These histograms indicate that tetrahedrons are well-shaped because most of them have radius-ratios of between 3.0 and 4.0. Hexahedrons and prisms are also well-shaped because the peaks of the histograms of hexahedrons and prisms are located near the maximum value that the minimum scaled Jacobian can take, which are 1.0 for a hexahedron and .866 for a prism as discussed in Section 3.2.2.

Figure 5-16 shows a hex-dominant mesh of the same geometry as that of the previous example with different directionality. For this example, the directionality is specified so that the hexahedrons are aligned with the cylindrical feature of the mechanical part, but the other portion of the mechanical part is not taken into account by the directionality. Figure 5-16 (a) shows the given directionality, and Figure 5-16 (b) shows the packing of rectangular solid cells. Figure 5-16 (c) and Figure 5-16 (d) show the output hex-

dominant mesh. The histogram of the radius-ratio of tetrahedrons, the scaled Jacobian of hexahedrons and prisms are shown in Figure 5-16 (e), Figure 5-16 (f) and Figure 5-16 (g). These histograms indicate that elements in the hex-dominant mesh are well shaped.

Figure 5-17 shows a graded hex-dominant mesh of the same geometry as the geometry of the previous example. In this example, the desired edge length, specified at the bottom of the cylindrical feature, is a half of the length specified at the top of the cylindrical feature. The specified directionality is identical to the Figure 5-15 (a). As can be seen from Figure 5-17 (a) and Figure 5-17 (b), the packed rectangular solid cells at the bottom of the cylindrical feature are small, so as the hexahedrons at the bottom of the cylindrical feature. The quality of the elements is shown in Figure 5-17 (e), Figure 5-17 (f) and Figure 5-17 (g), and it is reasonably high.

Figure 5-18 (d) and Figure 5-18 (e) show a hex-dominant mesh of an L-bracket, and Figure 5-18 (a) shows the input geometric domain and directionality. The packing of rectangular solid cells is shown in Figure 5-18 (b) and Figure 5-18 (c). The histogram of radius-ratio of tetrahedrons, the scaled Jacobian of hexahedrons and the scaled Jacobian of prisms are shown in Figure 5-18 (d), Figure 5-18 (e) and Figure 5-18 (f). The picture shows that hexahedrons are aligned with the given directionality, and hexahedrons, prisms and tetrahedrons are well-shaped.

Table 5-1 summarizes the statistics of the ratios of hexahedrons, prisms and tetrahedrons in the meshes in terms of the number of elements and volumes. The hex-dominant mesh of the mechanical part with a cylindrical feature, meshed with the boundary-aligned directionality, has hexahedrons filling almost three quarters of the entire volume, although the number of hexahedrons is less than 40% of the total number of elements. The volume ratio and the ratio of the number of elements are so different because a hexahedrons is created by merging five to six tetrahedrons, and an average hexahedron has about five to six times larger volume than an average tetrahedron.

The hex-dominant mesh of the mechanical part with a cylindrical feature, meshed with the boundary-unaligned directionality, indicates less volume ratio of hexahedrons than the part meshed with boundary-aligned directionality. The reason the volume ratio of hexahedrons lowers with boundary-unaligned directionality is because many non-hex elements are created near the boundary. If the given directionality

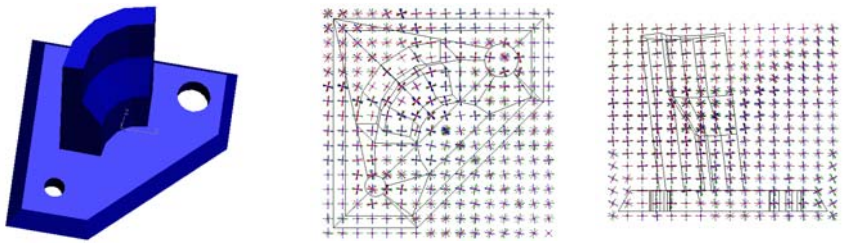
near the boundary is not aligned with the boundary, i.e., if none of three principal vectors is parallel to the normal of the boundary surface, hexahedrons gap near the boundary, and this gap is filled with non-hex elements. As a result the volume ratio of non-hex elements increases. However, if the boundary-unaligned directionality is given, creating fewer hexahedrons is the right solution for the input.

The graded hex-dominant mesh of the mechanical part also indicates lower volume ratio of hexahedrons than the uniform hex-dominant mesh of the mechanical part. When element size varies over the domain, some non-hex elements must be packed between a large hexahedron and a small hexahedron; as a result the percentage of hexahedrons decreases. The possible solution to this problem is to specify that the gradient of the element size in the critical region is small. If the gradient of the element size is small in the critical region, many hexahedrons are created in the critical region, and the result of the finite element analysis in and near the critical region is expected to be more accurate even though the overall hexahedron ratio is low.

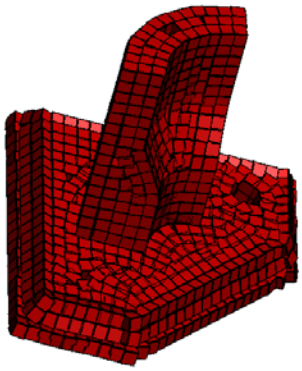
The hex-dominant mesh of the L-bracket has hexahedrons filling more than three quarters of the total volume, and the number of hexahedrons is 40% of the total number of elements. The hex-dominant mesh of the L-bracket is created with the boundary-aligned directionality; the result confirms that the boundary-aligned directionality increases the total number of hexahedrons.

Table 5-1 Statistics of the example hex-dominant meshes

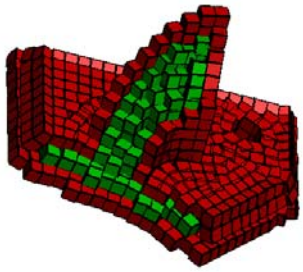
	Volume				Number of elements			
	HEX	PRISM	TET	TOTAL	HEX	PRISM	TET	TOTAL
Mechanical part with a cylindrical feature meshed with the boundary-aligned directionality (Figure 5-15)	2,486.7 (75.6%)	482.7 (14.5%)	362.4 (10.9%)	3,331.8	794 (39%)	238 (11%)	996 (49%)	2,028
Mechanical part with a cylindrical feature meshed with the boundary-unaligned directionality (Figure 5-16)	2,163.9 (64.9%)	459.9 (13.8%)	710.9 (21.3%)	3,334.7	630 (29%)	265 (12%)	1,258 (58%)	2,153
Mechanical part with a cylindrical feature meshed with the boundary-aligned directionality and graded element size (Figure 5-17)	2,145.6 (64.4%)	424.8 (12.8%)	760.8 (22.8%)	3331.3	2,428 (29%)	762 (9%)	4,976 (60%)	8,166
L-bracket (Figure 5-18)	569.1 (76.3%)	60.8 (8.2%)	115.8 (15.5%)	745.79	712 (40%)	165 (9%)	902 (50%)	1,779



(a) Input geometric domain and directionality



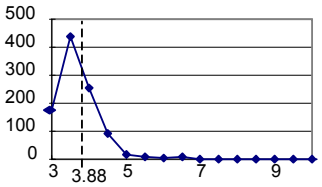
(b) Packing of rectangular solid cells



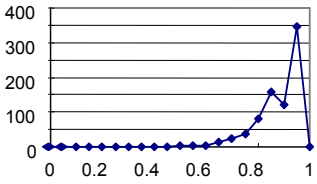
(c) Packing of rectangular solid cells (cross-section)

(d) Hex-dominant mesh

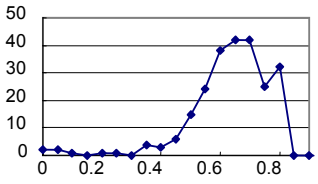
(e) Hex-dominant mesh (cross-section)



(f) Histogram of radius-ratio of tetrahedrons

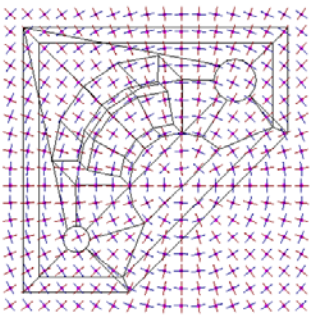


(g) Histogram of scaled Jacobian of hexahedrons

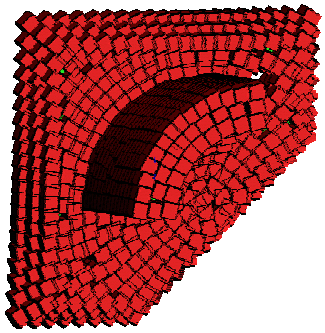


(h) Histogram of scaled Jacobian of prisms

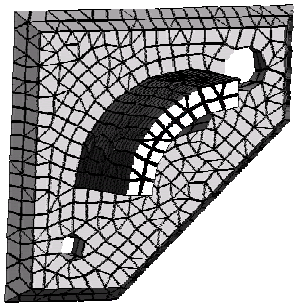
Figure 5-15 A hex-dominant mesh of a mechanical part with a cylindrical feature (meshed with boundary-aligned directionality)



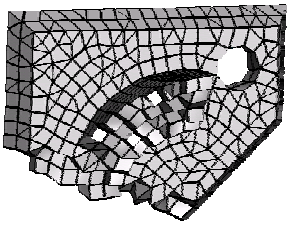
(a) Input directionality



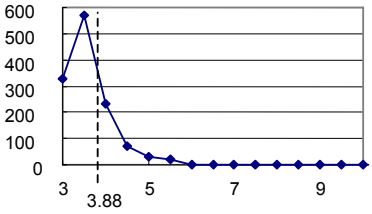
(b) Packed rectangular solid cells



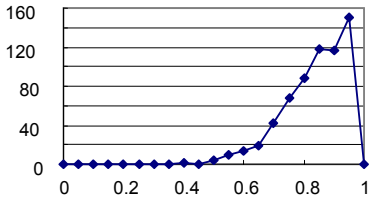
(c) Hex-dominant mesh



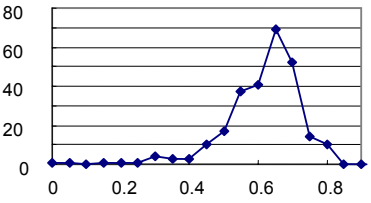
(d) Hex-dominant mesh (cross-section)



(e) Histogram of radius ratio of tetrahedrons

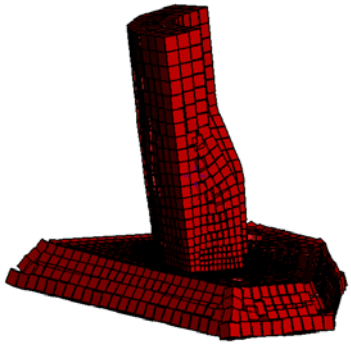


(f) Histogram of scaled Jacobian of hexahedrons

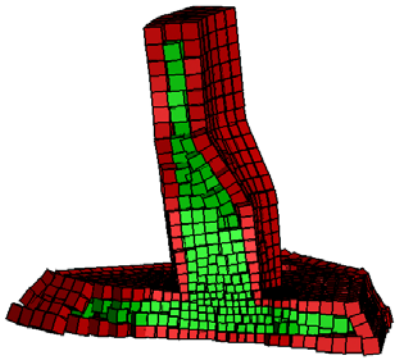


(g) Histogram of scaled Jacobian of prisms

Figure 5-16 A hex-dominant mesh of a mechanical part with a cylindrical feature (meshed with boundary-unaligned directionality)



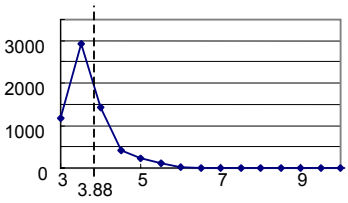
(a) Packed rectangular solid cells



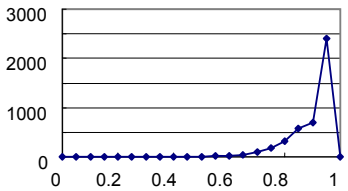
(b) Packed rectangular solid cells (cross-section)

(c) Graded hex-dominant mesh

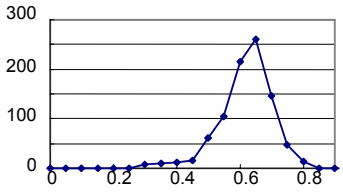
(d) Graded hex-dominant mesh (cross-section)



(e) Histogram of radius ratio of tetrahedrons

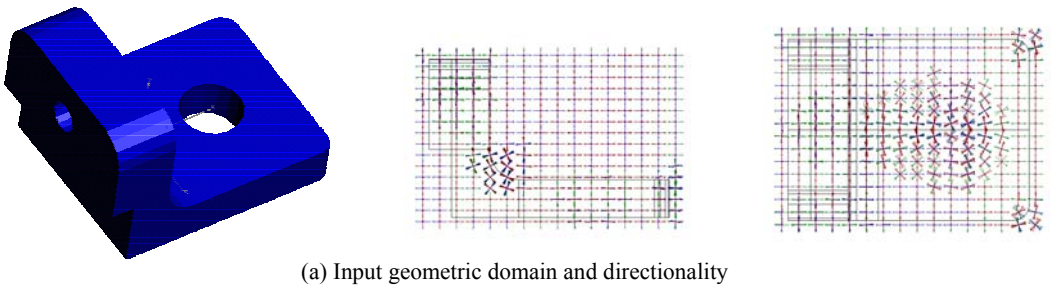


(f) Histogram of scaled Jacobian of hexahedrons

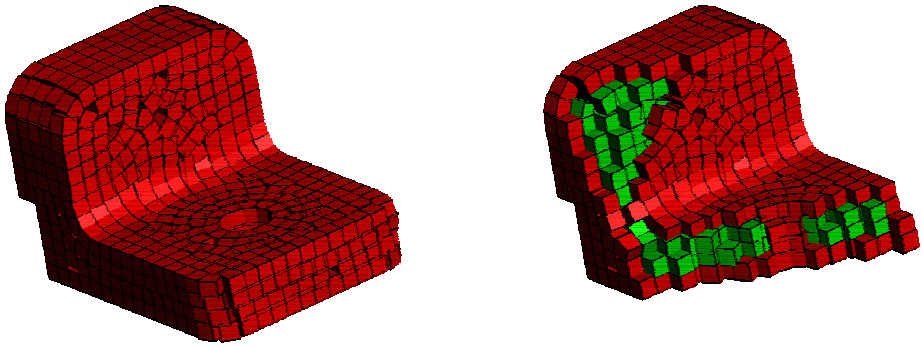


(g) Histogram of scaled Jacobian of prisms

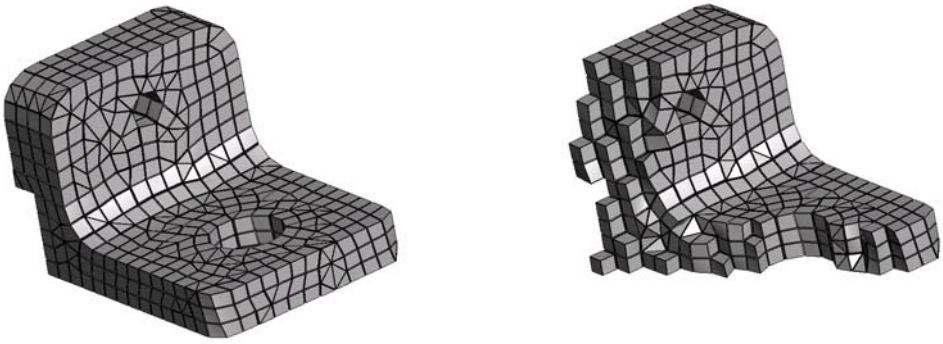
Figure 5-17 A graded hex-dominant mesh of a mechanical part with a cylindrical feature



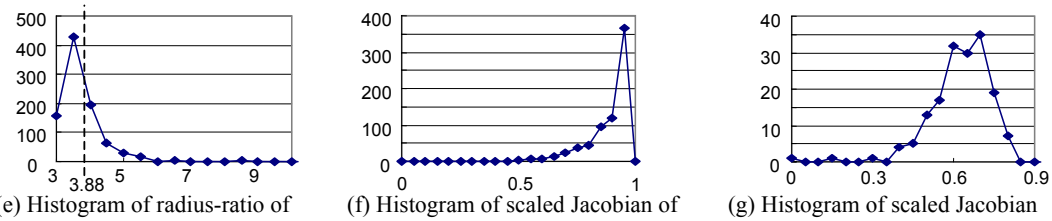
(a) Input geometric domain and directionality



(a) Packing of rectangular solid cells (b) Packing of rectangular solid cells (cross-section)



(c) Hex-dominant mesh (d) Hex-dominant mesh (cross-section)



(e) Histogram of radius-ratio of tetrahedrons (f) Histogram of scaled Jacobian of hexahedrons (g) Histogram of scaled Jacobian of prisms

Figure 5-18 A hex-dominant mesh of a L-bracket

5.9 Experimental Finite Element Analysis

Some structural finite element analyses are performed to show the performance of hex-dominant meshes. The experiments are conducted by using ANSYS [82], using the geometry of a mechanical part with a cylindrical feature, shown in Figure 5-15 (a). Five different tetrahedral meshes and three different hex-dominant meshes of various resolutions are fed to ANSYS, and ANSYS solves a structural analysis with the following loading conditions: (1) the bottom face of the mechanical part cantilevered; and (2) the force applied to the top surface of the cylindrical feature (to the centripetal direction of the cylindrical feature). The tetrahedral meshes are created by the method presented in Chapter 4, and none of the tetrahedrons in the meshes have radius ratio higher than 7.0; the mesh quality is thus very high. Hex-dominant meshes are created by the method presented in this paper, and since ANSYS can accept pyramids, Owen et al.'s method [10] is applied, and pyramids are placed on the non-conforming quadrilaterals to make the mesh conform fully.

To compare the accuracy of the results, the total displacement of the four corner nodes of the top surface of the cylindrical feature, obtained with tetrahedral meshes and hex-dominant meshes, is tabulated in Table 5-2 and plotted in Figure 5-19. The results show that the hex-dominant meshes outperform the tetrahedral meshes. The plots in Figure 5-19 clearly show that total displacement converges to near 2.1 with finer mesh sizes. To facilitate the comparison, let 2.1 be the acceptably accurate solution to the total displacement of the four nodes, and take it as a reference value to compute percent error of the solution. The tetrahedral mesh requires 89,726 elements (18,017 nodes) to obtain a solution with 1% error, while the hex-dominant mesh needs only 21,185 elements (9,966 nodes) to obtain a solution with 0.5% error. The tetrahedral mesh with 17,249 elements (3,951 nodes) shows 7% error, and it is almost same as the error obtained with the hex-dominant mesh with 3,608 elements (834 nodes). Again, the hex-dominant mesh needs less than one fourth the number of elements than the tetrahedral mesh to achieve the same accuracy.

A hex-dominant mesh needs fewer nodes than a tetrahedral mesh to obtain a solution of comparable accuracy (i.e., a finite element analysis takes less time with a hex-dominant mesh than with a tetrahedral mesh). Table 5-1 also shows the number of nodes of each mesh and the required computation time. These calculations were performed in ANSYS on a Pentium III 800 MHz PC with 512MB RAM. Figure 5-20 shows the computation time of each mesh as a function of the number of nodes. The number of nodes of a

hex-dominant mesh required to obtain 0.5% error is about a half of the number of nodes of a tetrahedral mesh required to obtain 1% error. Furthermore, ANSYS solved the hex-dominant mesh in less than half the time it required for the tetrahedral mesh. The dominant portion of computational time for a finite element analysis comes from solving the matrix system. For an n node model, the number of computational steps needed to solve the matrix system generally grows at a rate between n^2 and n^3. Thus, this reduction in the number of nodes significantly reduces the computational time required for solving a finite element analyses.

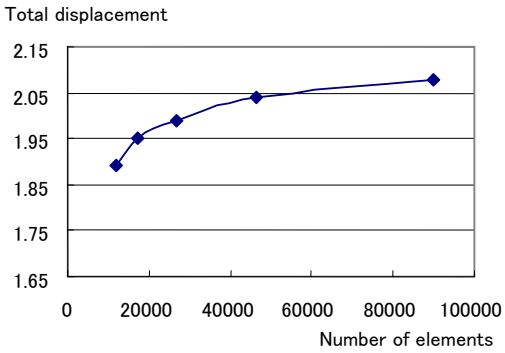
Figure 5-21 (a) and Figure 5-21 (b) plot distributions of the first principal stress, obtained with the hex-dominant mesh with 21,185 elements (9,966 nodes), and the tetrahedral mesh with 89,726 elements (18,017 nodes) respectively. In both plots, stress is concentrated at the bottom of the cylindrical feature, and there is another dim peak at the middle of the cylindrical feature. Although the plot obtained with the hex-dominant mesh has some jaggy contours, it captures well the contour around the most important region -- the region near the stress concentration.

Figure 5-21 (c) plots the distribution of the 1st principal stress obtained with a graded hex-dominant mesh with 16,146 elements. Although the plot has some jaggy contours, in general it agrees with the plot obtained with the tetrahedral mesh with 89,726 elements. The graded hex-dominant mesh tends to include more non-hex elements than a uniform hex-dominant mesh because some non-hex elements must fill the gaps between a large hexahedron and a small hexahedron, and the non-hex elements yield jagged contours. The future development of the error reduction technique for these non-hex elements may reduce those jagged contours, and a graded hex-dominant mesh will give more accurate results with fewer elements if such a technique is developed.

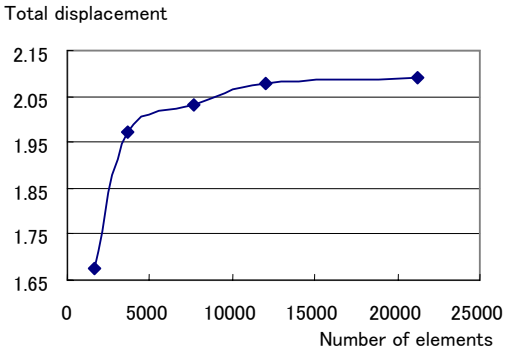
These experimental results show that the hex-dominant mesh with fewer elements gives a more accurate solution than the tetrahedral mesh. Thus, the hex-dominant mesh contributes to the reduction in the computational time of the finite element analysis.

Table 5-2 Number of elements and total displacements of the four corner nodes of the top surface of the cylindrical feature

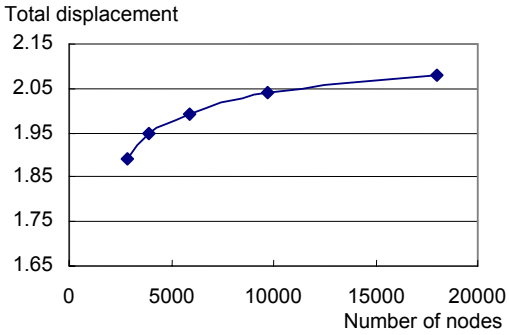
Tetrahedral mesh	Number of elements	12,063	17,249	26,986	46,240	89,726
	Number of nodes	2,869	3,951	5,939	9,736	18,017
	Displacement	1.89	1.95	1.99	2.04	2.08
	Computational time for the analyses	25 sec	31 sec	63 sec	275 sec	964 sec
	% error against 2.1	10%	7%	5%	3%	1%
Hex-dominant mesh	Number of elements	1,677	3,608	7,692	11,996	21,185
	Number of nodes	834	2,176	3,743	6,347	9,966
	Displacement	1.68	1.97	2.03	2.08	2.09
	Computational time for the analyses	8 sec	14 sec	45 sec	113 sec	453 sec
	% error against 2.1	20%	6%	3%	1%	0.5%
Graded hex-dominant mesh	Number of elements			16,146		
	Number of nodes			6,899		
	Displacement			2.04		
	Computational time for the analyses			262 sec		
	% error against 2.1			3%		



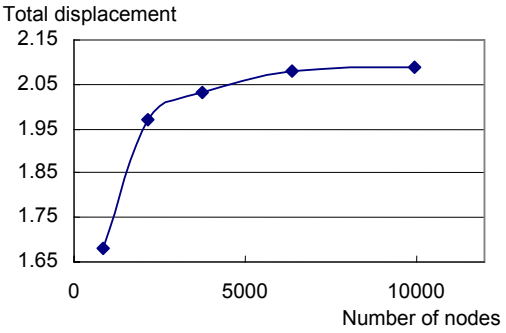
(a) Total displacements obtained with tetrahedral meshes against the number of elements



(b) Total displacements obtained with hex-dominant meshes against the number of elements



(c) Total displacement obtained with tetrahedral meshes against the number of nodes



(d) Total displacement obtained with hex-dominant meshes against the number of nodes

Figure 5-19 Plot of the total displacement of the four corner nodes of the top surface of the cylindrical feature

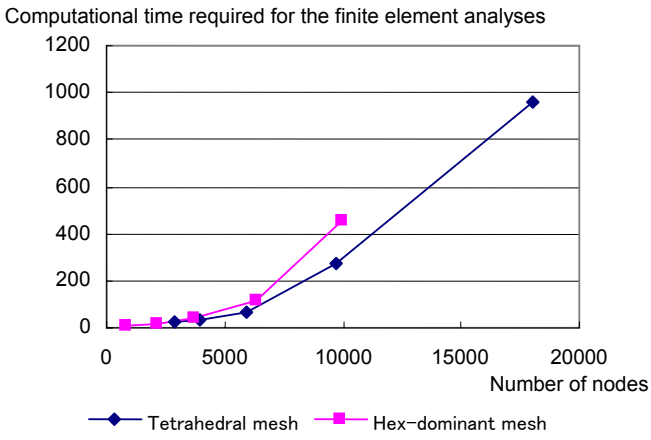
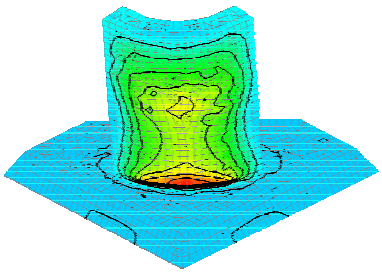
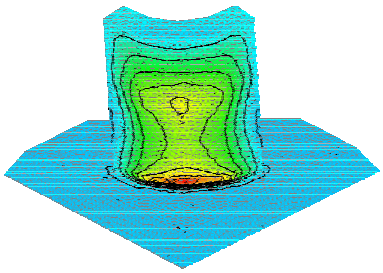


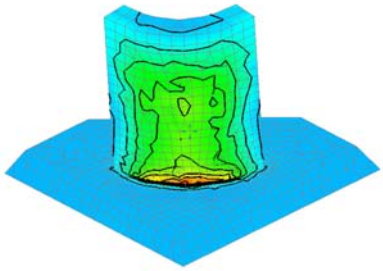
Figure 5-20 Plot of the computational time required for the finite element analyses against the number of nodes



(a) Stress distribution obtained by the hex-dominant mesh with 21,185 elements



(b) Stress distribution obtained by the dense tetrahedral mesh with 89,726 elements



(c) Stress distribution obtained by the graded hex-dominant mesh with 16,146 elements: Edge lengths of the elements at the bottom of the cylindrical feature are half the edge lengths of the elements at the top of the cylindrical feature.

Figure 5-21 Stress distribution obtained in the experimental finite element analysis

5.10 Industrial Applications

To demonstrate the capability of the proposed methods in realistic industrial applications, this section presents four hex-dominant meshes of mechanical parts from industrial applications: (1) a crank, (2) a connecting rod, (3) a sheet metal part, and (4) a pipe system. The mechanical parts of real industrial applications typically have small features or high curvature regions, which make it difficult to create a hex-dominant mesh.

The proposed method creates hex-dominant meshes for the four geometries with the geometry-based directionality created by the method explained in Section 5.3. The hex-dominant meshes are shown in Figure 5-22, Figure 5-23, Figure 5-24 and Figure 5-25. Their qualities are also shown in the figures with histogram of radius ratio of tetrahedrons, histogram of minimum scaled Jacobian of hexahedrons, and histogram of minimum scaled Jacobian of prisms. Table 5-3 shows the ratios of hexahedrons, prisms and tetrahedrons in terms of volume and number of elements in each mesh. The qualities of the meshes

indicate a pattern similar to that of the examples presented in Section 5.8, and the qualities are reasonably high. However, hexahedron ratios show a considerable variation among the four examples.

Among the four hex-dominant meshes, the hex-dominant mesh of the sheet metal part shown in Figure 5-24 shows a significantly high hexahedron ratio; 77% of the volume is filled with hexahedrons. Since the sheet metal part does not have small features, and many of the edges included in the part are intersecting nearly perpendicularly, the geometry of the sheet metal part is relatively easy for hex-dominant meshing compared to the other examples.

The hex-dominant mesh of the crank shown in Figure 5-22 shows the second highest hexahedron ratio among the four hex-dominant meshes. It shows such a high hexahedron ratio because the crank does not have small features, while the hex-dominant meshes of the connecting rod and the pipe system shown in Figure 5-24 and Figure 5-25.

The hex-dominant meshes of the connecting rod and the pipe system shown in Figure 5-24 and Figure 5-25 indicate lower hexahedron ratios than the other examples. The connecting rod shown in Figure 5-24 has small features such as fillets around the holes. Furthermore, the thickness of the beam between the two holes of the rod varies, and the beam is thin at the middle and thick at the both ends. The thin and thick portions of the cross-section of the beam are smoothly connected together by fillets. The pipe-system shown in Figure 5-25 has cylindrical features sticking out of the exterior faces of the pipes. Three branches of the pipe are connected smoothly to the foundation by fillets. Intersections of the pipe branches also make complex surfaces. Such complex geometry and features make hex-dominant meshing difficult. And thus the ratio of hexahedrons becomes smaller. Nonetheless, in both examples, the proposed method fills more than 70% of the volume with hexahedrons and prisms. The proposed method is thus capable of dealing with such complex geometries.

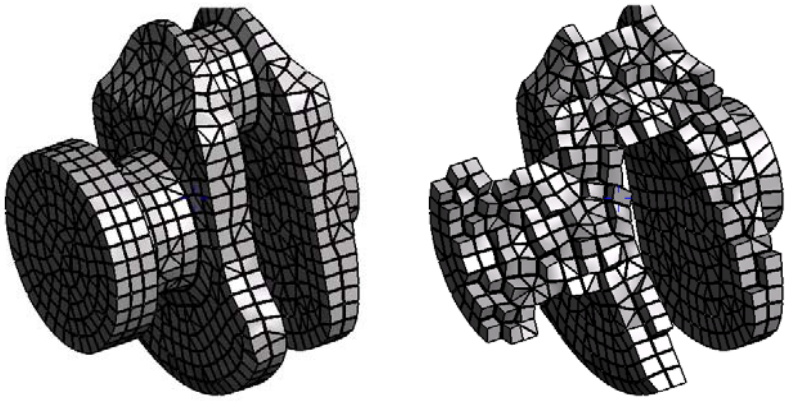
If even higher hexahedron ratio is desired, some kind of feature suppression technique such as Ribelles et al.'s [83] must be applied. However, such feature suppression techniques are typically unstable, or computationally expensive.

An alternative solution to the feature problem is to control the mesh size so that smaller elements are created in and near the features. However, currently available feature identification techniques are

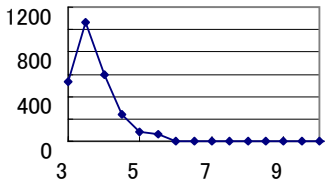
typically unstable. The future development of such feature suppression techniques and feature identification techniques will increase the applicability of the proposed method.

Table 5-3 Volume ratios and number of element ratios of hex-dominant meshes of crank, connecting rod, sheet metal part, and pipe-system

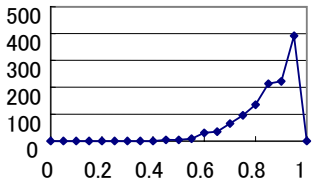
	Volume				Number of elements			
	Hex	Prism	Tet	Total	Hex	Prism	Tet	Total
Crank	372,599 (67.4%)	68,552 (12.4%)	111,531 (20.2%)	552,682	1,252 (29%)	490 (11%)	2455 (58%)	4,197
Connecting rod	13,954 (59.5%)	3,839 (16.4%)	5,651 (24.1%)	23,444	533 (23%)	301 (13%)	1,390 (62%)	2,224
Sheet metal part	52,989 (77.1%)	7,077 (10.3%)	8,630 (12.5%)	68,696	391 (44%)	102 (11%)	376 (43%)	869
Pipe system	132,671 (58.3%)	37,075 (16.3%)	57,821 (25.4%)	227,567	757 (22%)	444 (13%)	2193 (64%)	3,394



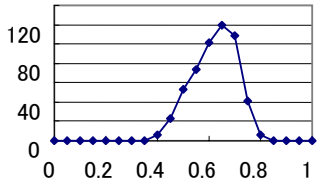
(a) Hex-dominant mesh and its cross-section



(b) Histogram of radius ratios of tetrahedrons

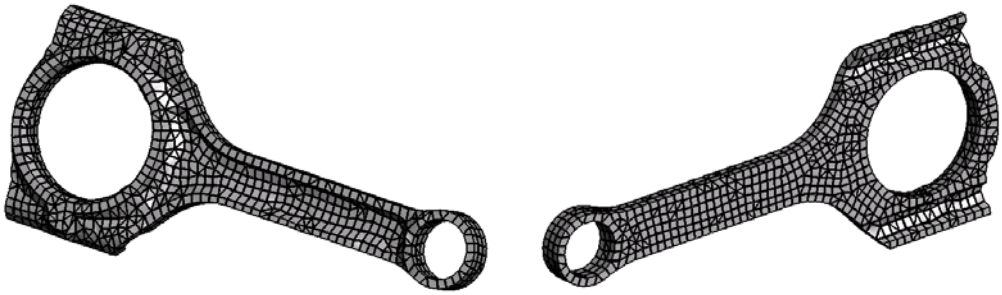


(c) Histogram of minimum scaled Jacobian of hexahedrons

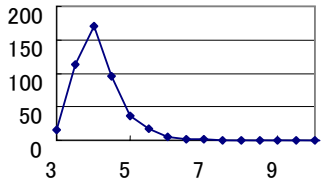


(d) Histogram of minimum scaled Jacobian of prisms

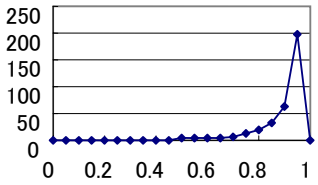
Figure 5-22 A hex-dominant mesh of a crank



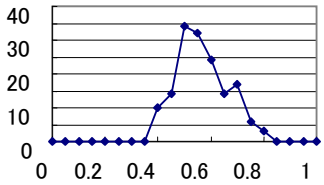
(a) Hex-dominant mesh from two different view points



(b) Histogram of radius ratios of tetrahedrons

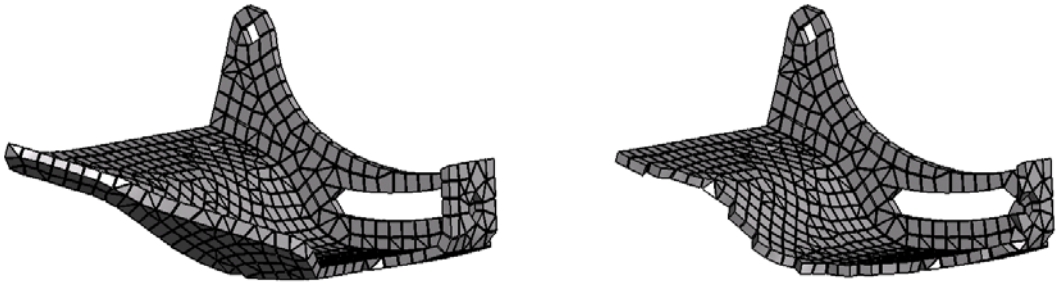


(c) Histogram of minimum scaled Jacobian of hexahedrons

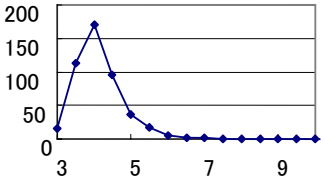


(d) Histogram of minimum scaled Jacobian of prisms

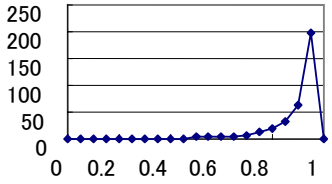
Figure 5-23 A hex-dominant mesh of a connecting rod



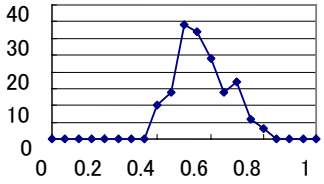
(a) Hex-dominant mesh and its cross-section



(b) Histogram of radius ratios of tetrahedrons

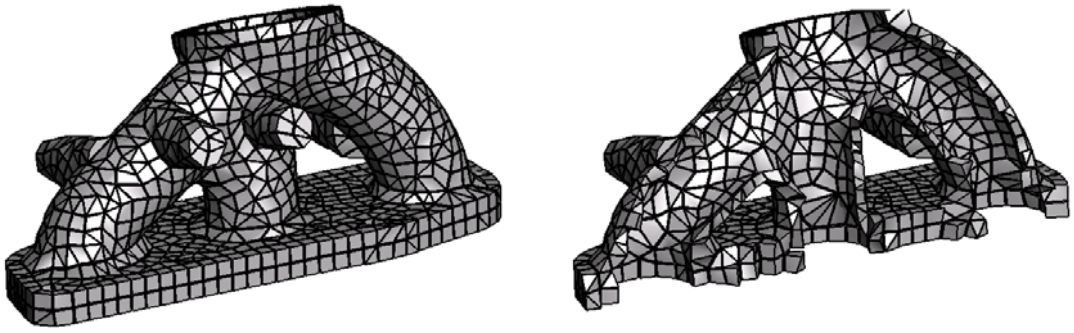


(c) Histogram of minimum scaled Jacobian of hexahedrons

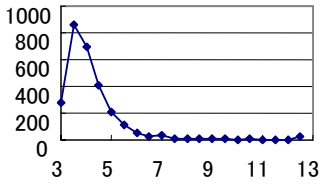


(d) Histogram of minimum scaled Jacobian of prisms

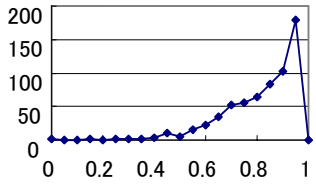
Figure 5-24 A hex-dominant mesh of a sheet metal part



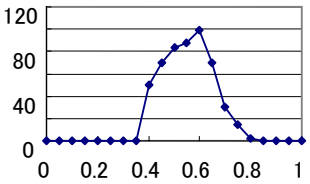
(a) Hex-dominant mesh and its cross-section



(b) Histogram of radius ratios of tetrahedrons



(c) Histogram of minimum scaled Jacobian of hexahedrons



(d) Histogram of minimum scaled Jacobian of prisms

Figure 5-25 A hex-dominant mesh of a pipe-system

5.11 Expansion to Anisotropic Hex-Dominant Mesh Generation

The method presented in this chapter can be expanded to anisotropic hex-dominant mesh generation. For the isotropic version of the method, the lengths of three column vectors of matrix $\mathbf{M}(\mathbf{x})$, d_1, d_2 and d_3, are constrained as $d_1 = d_2 = d_3$. (Recall the representation of anisotropy, directionality and element size explained in Section 3.1.) The constraint is removed to expand the method for the anisotropic hex-dominant mesh generation, i.e., d_1, d_2 and d_3 are allowed to have different values. Then, matrix $\mathbf{M}(\mathbf{x})$ transforms a unit cube to a rectangular solid of aspect ratio $d_1 : d_2 : d_3$, and the main bubble of the unit cube is transformed to an ellipsoid at the center of the rectangular solid, and the eight sub-bubbles of the unit cube are transformed to eight ellipsoids at the eight corners of the rectangular solid cell as illustrated in Figure 5-26. Nonetheless, no change is necessary for the meshing process to cope with the anisotropy because: (1) the $\mathbf{f}$ term of the governing equation of the cells can take anisotropy into account as explained in Section 3.4.2, and (2) a tetrahedral mesh to be converted to a hex-dominant mesh can be created by the advancing front method with anisotropy taken into account as explained in Section 4.3, and (3) qualities of hexahedrons and prisms are measured with anisotropy taken into account as explained in Section 3.2.4.

Thus, by simply removing the constraint, $d_1 = d_2 = d_3$, the method can create an anisotropic hex-dominant mesh.

The result, however, heavily depends on the shape of the target domain and the anisotropy. If none of three principal vectors of **M** is parallel to the normal vector on the boundary of the domain, the ratio of hexahedrons becomes small because many non-hex elements are created near the boundary as already discussed in Section 5.8.

Here are presented two examples, (1) one with boundary-aligned anisotropy and (2) the other with boundary-unaligned anisotropy. The geometric domain of example (1) is enclosed by two curved surfaces, $y = \frac{30}{\pi}\sin\frac{\pi x}{30} + 5$, $y = \frac{30}{\pi}\sin\frac{\pi x}{30} - 5$, and four planes $z = -5$, $z = 5$, $x = -15$, and $x = 15$. The geometric domain of example (2) is a cube enclosed by six planes $y = 12.5$, $y = -12.5$, $x + z = 17.68$, $x + z = -17.68$, $x - z = 17.68$, $x - z = -17.68$. For both two geometric domains, anisotropic hex-dominant meshes are created with following anisotropy:

$$\mathbf{u} = \left(\frac{1}{\sqrt{1+\cos^2\frac{\pi x}{30}}} \quad \frac{\cos\frac{\pi x}{30}}{\sqrt{1+\cos^2\frac{\pi x}{30}}} \quad 0 \right)^T$$

$$\mathbf{v} = \left(\frac{-\sin\frac{\pi x}{30}}{\sqrt{1+\sin^2\frac{\pi x}{30}}} \quad \frac{1}{\sqrt{1+\sin^2\frac{\pi x}{30}}} \quad 0 \right)^T$$

$$\mathbf{w} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T$$

$$d_1 = 3.0$$

$$d_2 = d_3 = 1.0$$

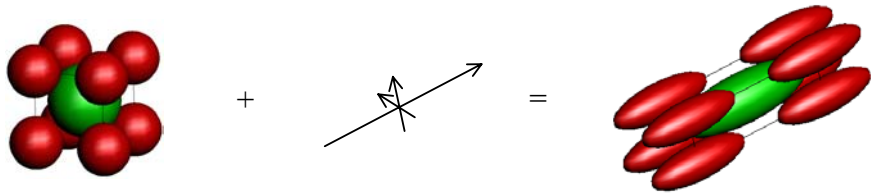
The first principal vector **u** is always parallel to the surface $y = \frac{30}{\pi}\sin\frac{\pi x}{30} + c$, where c is an arbitrary constant, and the second principal vector **v** is always perpendicular to the curve $y = \frac{30}{\pi}\sin\frac{\pi x}{30}$. Both **u** and **v** are parallel to the plane $z = c$, and **u** is parallel to the plane $x = c$ at $x = \pm 15$. Thus, at least one of the principal vectors is always parallel to the normal vector on the boundary in example (1). In contrast, none of the principal vectors is parallel to the normal vector on the boundary in example (2). The packed

rectangular solid cells, output hex-dominant meshes, and quality measurements are shown in Figure 5-27 and Figure 5-28, and both meshes show reasonably good qualities.

However, two meshes show significant difference in the ratio of hexahedrons in terms of both the number of elements and the volume. The hex-dominant mesh of example (1) has larger percentage of hexahedrons than the hex-dominant mesh of example (2). The statistics of the output hex-dominant meshes are shown in Table 5-4, and it shows that the number of hexahedrons is 34% of total number of elements in example (1), while it is only 20% in example (2). The total volume of hexahedrons is 72.8% of total volume of the geometric domain in example (1), while it is only 55.0% in example (2).

The results indicate that if the anisotropy does not take the boundary into account, fewer hexahedrons will be created. However, if the user needs boundary-unaligned anisotropy, creation of many non-hex elements near the boundary is unavoidable as explained in Section 5.8. If the user needs more hexahedrons, the input anisotropy must be modified so one of the principal vectors on the boundary becomes always parallel to the normal vector. However, future research is required to develop such a technique.

Another possible expansion of this technique is the use of parallelepiped cells. Consider the example shown in Figure 5-27. If the shape of a cell is specified so that two faces are parallel to YZ plane, two faces are parallel to XY plane, and two faces are parallel to vector $\mathbf{u}$ and perpendicular to XY plane, the percentage of hexahedrons can potentially be increased. The proposed method can be adapted to these parallelepiped cells by relaxing a constraint on $\mathbf{M}(\mathbf{x})$ and allowing the three column vectors to be non-orthogonal. However, finding an appropriate parallelepiped shape for a general domain is not a trivial problem, and it is one of the future research topics.

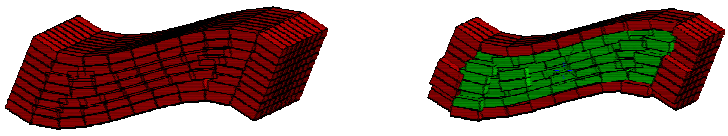


(a) A unit cube aligned with X-, Y- and Z-axis

(b) Anisotropy

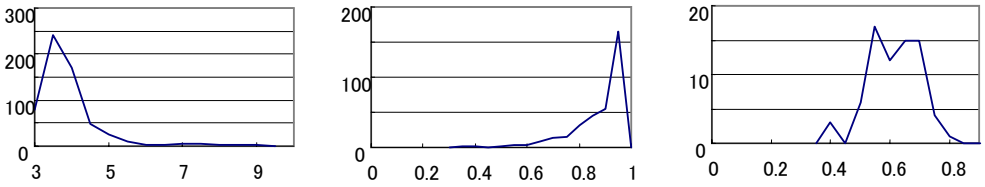
(c) A rectangular solid

Figure 5-26 Transformation of a unit cube to a rectangular solid by an anisotropy



(a) Packed rectangular solid cells and cross-section

(b) Hex-dominant mesh and cross-section



(c) Histogram of radius ratio of tetrahedrons

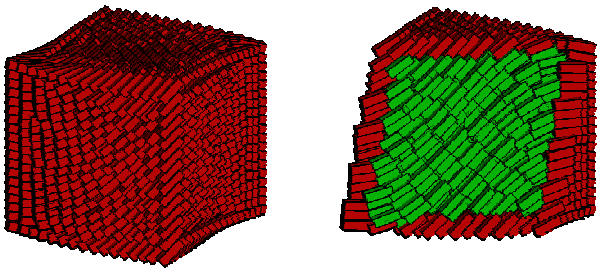
(d) Histogram of scaled Jacobian of hexahedrons

(e) Histogram of scaled Jacobian of prisms

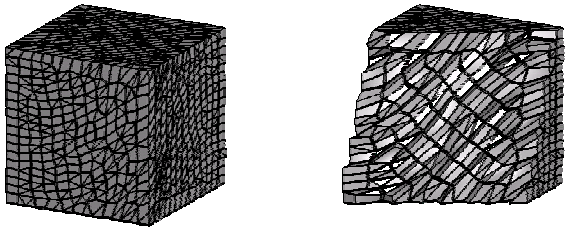
Figure 5-27 Example (1) of an anisotropic hex-dominant mesh

Table 5-4 Statistics of the examples of anisotropic hex-dominant meshes

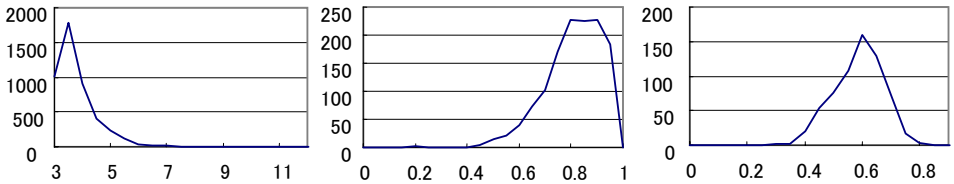
	The number of elements and ratio				Volume and ratio			
	HEX	PRISM	TET	Total	HEX	PRISM	TET	Total
An anisotropic hex-dominant mesh with boundary-aligned anisotropy (Figure 5-27)	346 (34%)	73 (7%)	589 (58%)	1008	2184.8 (72.8%)	235.8 (7.9%)	579.6 (19.3%)	3000.2
An anisotropic hex-dominant mesh with boundary-unaligned anisotropy (Figure 5-28)	1295 (20%)	636 (9%)	4528 (70%)	6459	8598.3 (55.0%)	2243.7 (14.4%)	4783.0 (30.6%)	15625.0



(a) Packed rectangular solid cells and cross-section



(b) Hex-dominant mesh and cross-section



(c) Histogram of radius ratio of tetrahedrons

(d) Histogram of scaled Jacobian of hexahedrons

(e) Histogram of scaled Jacobian of prisms

Figure 5-28 Example (2) of an anisotropic hex-dominant mesh

5.12 Conclusion

This chapter has presented a new method for creating a hex-dominant mesh consisting of hexahedrons, prisms and tetrahedrons. The process is fully automatic, and the output hex-dominant mesh show high quality. A hex-dominant mesh combines good aspects of a tetrahedral mesh and an all-hex mesh: the grading of element size can be controlled precisely as in a tetrahedral mesh, and a hex-dominant mesh gives a more accurate solution with fewer elements than a tetrahedral mesh.

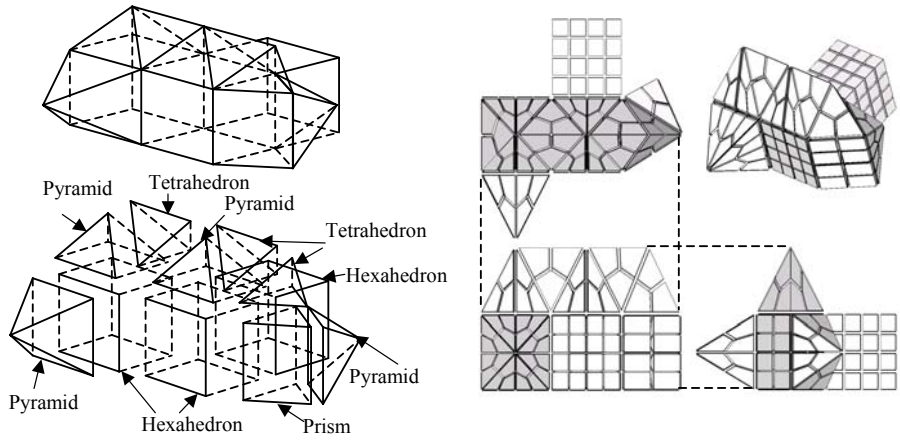
The method takes as input a 3D geometric domain, desired edge length and mesh directionality, and creates a hex-dominant mesh in three steps: (1) packing rectangular solid cells on the boundary and the interior of the domain to obtain good node locations for a hex-dominant mesh, (2) creating a tetrahedral mesh based on the nodes created in the second step, and (3) converting a tetrahedral mesh to a hex-dominant mesh. The proposed method avoids ill-shaped elements induced by nodes located too close together and creates well-shaped elements in an output hex-dominant mesh. Several experimental results show that the proposed method creates hex-dominant mesh of well-shaped elements.

Although a hex-dominant mesh has some non-conforming quadrilaterals, which induce errors in the finite element analysis, several solutions are available; (1) reducing errors by MPCs as presented by Dewhirst et al. [79, 80]; (2) reducing errors by the technique allowing connection between dissimilar surface meshes as presented by Dohrmann et al. [81]; or (3) eliminating non-conforming quadrilaterals by introducing pyramids as presented by Owen et. al [10]. However, these techniques that cope with non-conforming quadrilaterals are not yet developed for some analyses such as fluid mechanics simulations. The future expansion of those techniques to more general analyses will increase the applicability and value of the proposed hex-dominant mesh generation techniques.

Chapter 6 Modular Templates for Hex-dominant to All-hex Mesh Conversion

6.1 Introduction

This chapter proposes new mesh conversion templates, called HEXHOOP, that fully-automate a conversion from a hex-dominant mesh to an all-hex mesh. A hex-dominant mesh is a three-dimensional mesh consisting of four types of elements, hexahedrons, prisms, pyramids and tetrahedrons, as illustrated in Figure 6-1 (a). An all-hex mesh is a mesh consisting of exclusively hexahedral elements. Figure 6-1 (b) shows an example of an all-hex mesh converted from a hex-dominant mesh shown in Figure 6-1 (a) by using our HEXHOOP templates.



(a) an example of a hex-dominant mesh (b) an all-hex mesh

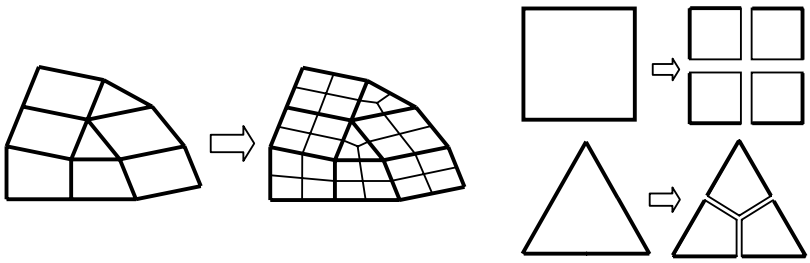
Figure 6-1 Converting a hex-dominant mesh to an all-hex mesh

Such conversion templates make all-hex meshing possible for a complicated geometry for which the existing direct all-hex meshing methods are inadequate. Although it would be ideal if an all-hex mesh could be generated for an arbitrary three-dimensional shape without going through a hex-dominant mesh, the direct all-hex meshing problem is known to be highly challenging, and none of the exiting methods always succeeds to create a high-quality all-hex mesh for a complex three-dimensional geometry. Although there exists a trivial solution, creating a tetrahedral mesh first and subdividing each tetrahedron into four smaller hexahedrons, such an all-hex mesh is topologically so irregular that this method is not used in practice. Creating a quality hex-dominant mesh, on the other hand, is an easier problem to solve,

either by hand or by an automated algorithm [36, 38, 67]. The HEXHOOP templates take as input any type of hex-dominant mesh and convert it to an all-hex mesh automatically.

To highlight the difficulty in developing conversion templates for all-hex meshing, a much easier two-dimensional problem of converting a quad-dominant mesh to an all-quad mesh, as illustrated in Figure 6-2 (a), is examined. The solution to this problem is well known—only two types of templates are needed as shown in Figure 6-2 (b). With these two types of templates, it is guaranteed that any quad-dominant mesh is successfully converted to an all-quad mesh.

During this all-quad mesh conversion, it is important to satisfy the topological validity, or interface-conformity requirement, between adjacent mesh elements. To satisfy the interface-conformity requirement, every interior edge of an output all-quad mesh must be shared by exactly two elements. By using the two templates shown in Figure 6-2 (b) it is easy to satisfy such conformity requirement in the all-quad mesh conversion because all the edges of an input quad-dominant mesh are always split into two segments.



(a) Conversion of a quad-dominant mesh to an all-quad mesh

(b) Two all-quad conversion templates

Figure 6-2 Templates for converting a quad-dominant mesh to an all-quad mesh

A similar interface-comformity requirement exists in the all-hex mesh conversion problem. The common method for converting a hex-dominant mesh to an all-hex mesh is to subdivide, or dice, a non-hex element into a set of smaller hexahedrons. A hexahedron in the original mesh is also subdivided into a set of smaller hexahedrons. In a final all-hex mesh, all the interfaces between adjacent hexahedrons must be quadrilaterals, and each of the quadrilaterals must be shared by exactly two hexahedrons in order to satisfy the interface-conformity requirement.

Despite the apparent similarity between the problem statements, this all-hex mesh conversion problem is significantly more challenging than the all-quad mesh conversion problem for the following two reasons:

- An input hex-dominant mesh consists of four different types of elements, hexahedrons, prisms, pyramids and tetrahedrons, as opposed to only two types of elements in a quad-dominant mesh.
- A hex-dominant mesh has two types of interfaces, triangles and quadrilaterals, which makes it more difficult to maintain the topological and geometric conformity at the interfaces, compared with the all-quad mesh conversion problem, in which there is only one type of interfaces, a line segment.

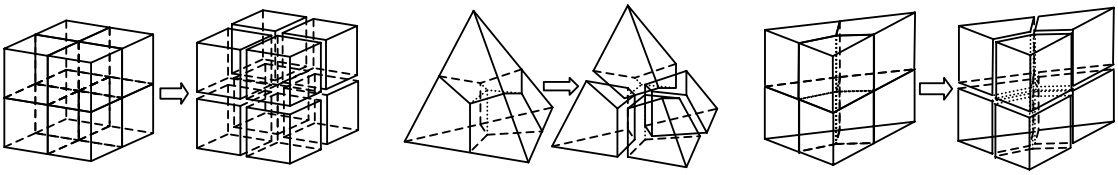
Among the four types of elements in a hex-dominant mesh, hexahedrons, tetrahedrons, and prisms have the following well-known, simple conversion templates:

- A hexahedron can be split into eight smaller hexahedrons by adding a node at the center of the volume, six nodes at the centers of six quadrilateral faces, twelve nodes at the centers of twelve edges of the original hexahedron as shown in Figure 6-3 (a).
- A tetrahedron can be split into four smaller hexahedrons by adding a node at the center of the volume, four nodes at the centers of four triangular faces, and six nodes at the centers of six edges of the original tetrahedron as shown in Figure 6-3 (b).
- A prism can be split into six smaller hexahedrons by adding a node at the center of the volume, five nodes at the centers of two triangular faces and three quadrilateral faces, and nine nodes at the centers of nine edges of the original prism as shown in Figure 6-3 (c).

Note that all of these three templates apply aforementioned all-quad templates, shown in Figure 6-2 (b), —splitting a triangular face of the original element into three smaller quadrilaterals, and a quadrilateral face into four smaller quadrilaterals.

One problem is that there is no such template known for a pyramid; if it exists, such a template should subdivide four triangular faces of the pyramid into three smaller quadrilateral faces and the bottom quadrilateral face into four smaller quadrilaterals as shown in Figure 6-4 (a). This subdivision pattern of the bottom quadrilateral face and its variations are referred to as *rectangular patterns*. Given this boundary mesh consisting of 16 quadrilaterals, finding a valid internal structure that dices the pyramid into a set of

smaller hexahedrons is difficult, and there is no valid solution published for this open problem as Schneiders discusses in his paper [45] and in one of his web pages [84]. This problem is referred to as "Schneiders' Open Problem". Although Carboner proposes a solution to the problem [85], his solution is not valid because some interior faces are not shared by two hexes.

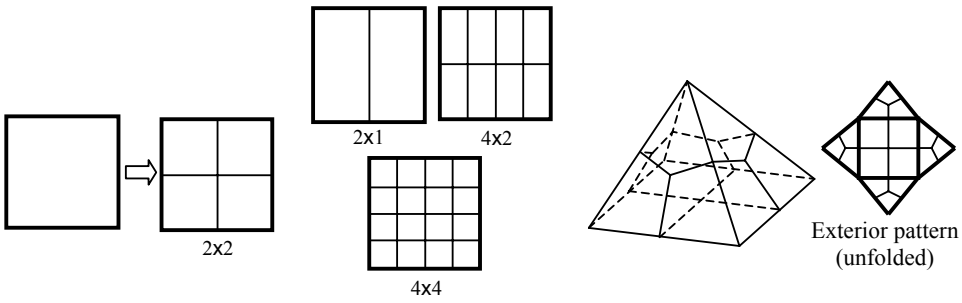


(a) A subdivision template for a hexahedron

(b) A subdivision template for a tetrahedron

(c) A subdivision template for a prism

Figure 6-3 Known subdivision templates for a hexahedron, a tetrahedron and a prism



(a) basic rectangular pattern

(b) variations of rectangular pattern

(c) Schneiders' Open Problem: a valid dicing of this pyramid has not been published.

Figure 6-4 Rectangular pattern and Schneiders' Open Problem

One known simple template for a pyramid first splits the pyramid into two tetrahedrons, and then applies the known tetrahedron-template to each tetrahedron. With this template the bottom quadrilateral face of a pyramid is split into a pattern shown in Figure 6-5 (a). This pattern and its variations are referred to as *triangular patterns*. However, if a bottom face of a pyramid is subdivided into a triangular pattern, a hexahedron or a prism adjacent to the pyramid must have a triangular pattern on one face in order to maintain the interface conformity. This brings up another unsolved problem of finding conversion templates for a hexahedron and a prism that have both rectangular patterns and triangular patterns mixed on the exterior surface of the hexahedron and the prism. Mitchell presents a partial solution to this problem as

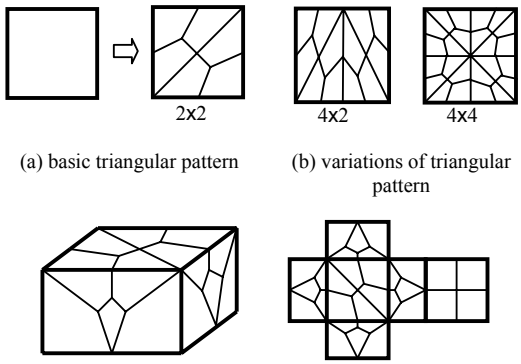
shown in Figure 6-5 (c) [86], but this template has an irregular subdivision pattern on the side faces of a hex, which limits its application and practical value.

In summary, there are two approaches to the all-hex conversion template, but no complete solution to these two approaches has been published:

(1) to find templates for a pyramid with a rectangular subdivision pattern on the bottom quadrilateral face, as Schneiders pointed out in [45] and [84], and

(2) to find a template for a hexahedron and a prism that have mixed subdivision patterns, rectangular and triangular, as Mitchell attempted [86].

This chapter proposes a family of new mesh conversion templates, called HEXHOOP, that provide solutions to both unsolved problems. Unlike previously published templates, HEXHOOP is not a single specific template; it is a systematic method for constructing a family of modular sub-templates that can be assembled to form all-hex conversion templates for hexahedrons, pyramids, and prisms. This new template design uses two types of modular sub-templates, called a *core* and a *cap*. One core is defined for a hexahedron or a prism, and a core specifies the subdivision patterns of two opposite faces of the input hexahedron or prism. Four caps are defined for a hexahedron, and three caps are defined for a prism to specify the subdivision patterns of the other faces. The advantage of the HEXHOOP method is that two subdivision patterns, rectangular and triangular, can be mixed and matched freely on the exterior surfaces of a hexahedron, prism, and a pyramid.



(a) basic triangular pattern (b) variations of triangular pattern

(c) Mitchell's Geode template: the side face has an irregular pattern.

Figure 6-5 Triangular pattern and Mitchell's Geode template

6.2 Modular Approach to All-Hex Templates

The goal of the proposed method is to develop a system of all-hex templates for hexahedrons, prisms, and pyramids. As discussed in Section 6.1 the difficulty is that there exist two face subdivision patterns, rectangular pattern and triangular pattern, mixed on the exterior faces of a hexahedron, prism, and pyramid. Sections 6.2 and 6.3 primarily discuss templates for hexahedrons in detail to explain the new modular design approach of HEXHOOP. Templates for prisms and pyramids are discussed later in Sections 6.4 and 6.5.

Because a hexahedron-template has six exterior faces, and each of the faces has either a rectangular or triangular pattern, there are ten different hexahedron-templates. The number of exterior faces with a triangular pattern ranges from 0 to 6, and in the cases where the number of such exterior faces is 2, 3, and 4, there exists two topologically different ways to choose the exterior faces as illustrated in Figure 6-6. Among the ten cases there are two cases for which there exists a known, simple solution: a hexahedron-template with six rectangular patterned faces (Figure 6-6 (b)), and a hexahedron-template with two triangular patterned faces, one on the top and one on the bottom (Figure 6-6 (e)). Solutions to the other eight cases are not trivial, there has been no published solution to creating a valid template for all of the cases.

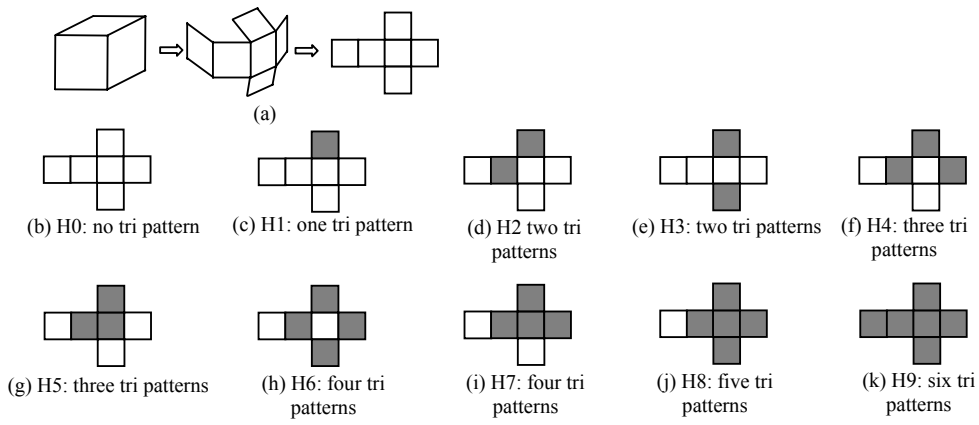


Figure 6-6 Ten possible combinations of triangular patterns and rectangular patterns on a template (shaded faces are triangular patterned faces)

To tackle this challenging all-hex template problem, a new modular approach is proposed, and it provides a systematic method of constructing a family of modular sub-templates that can be assembled to form all-hex conversion templates. The new template design uses two types of modular sub-templates, called a *core* and a *cap*. For a hexahedron-template one core specifies the subdivision patterns of two opposite faces and then four caps specifies the subdivision patterns of the other four faces, as shown in Figure 6-7.

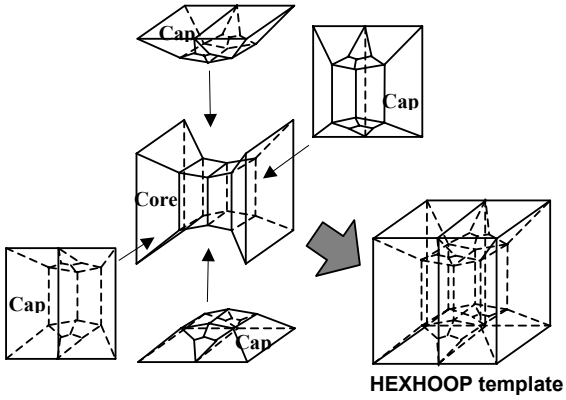


Figure 6-7 HEXHOOP's modular approach to a all-hex template

The design of the cap is key to the modular template design. Each cap has two faces, front and back, with an irregular subdivision pattern. Thus it exhibits the same problem as Mitchell's Geode template—all the irregular faces must be matched and shared with the next cap. Mitchell addressed this problem by assuming that all the Geodes are laid out in a closed shell-like volume, which is a strong restriction on the applicability of the template. In the HEXHOOP approach this problem is solved by arranging four caps to form a hoop, or a ring. In this way, all the irregular faces are matched and shared by adjacent caps without exposing them to the exterior of the template. This is the single most important feature of the modular template design. Therefore the template is named "HEXHOOP", meaning a hoop of hexahedrons.

Each cap has one exterior face to be subdivided into either a rectangular or triangular pattern. The construction process for each type of cap is as follows: First, one end face of an all-hex mesh consisting of four elements is subdivided into a triangular pattern as shown in Figure 6-8 (a). This triangular pattern marches through a mesh and reaches the other end as shown in Figure 6-8 (b). Next, the corners of the two ends are joined as shown in Figure 6-8 (b). By joining two corners a volumetric region surrounded by inside faces of the mesh is created as shown in Figure 6-8 (c). This region is called a *pipe*. Some hexahedrons (in this case two hexahedrons) are added to fill the pipe. Finally, an appropriate deformation scheme is applied so that neither gaps nor overlaps remain when the cap is assembled with a core and other caps. The final shape of the cap is shown in Figure 6-8 (d). This all-hex sub-template is called a cap of the HEXHOOP template. Because the top faces form a triangular pattern, a diced tetrahedron, pyramid, or prism can be connected without losing the mesh conformity. In a similar way one end face can be subdivided into a rectangular pattern. In this way, a cap with a rectangular pattern on its external face is obtained. To distinguish two different types of caps, a cap with a triangular pattern is referred to as a *triangular cap* and a cap with a rectangular pattern is referred to as a *rectangular cap*.

An important property of this cap design is that both rectangular and triangular caps share the identical boundary face subdivision patterns except on the top face. It is especially convenient to have a simple rectangular pattern on both types of caps on the bottom face. A volumetric region surrounded by four 'hooped' caps is therefore a region surrounded by four rectangular patterns. A sub-template for this volume is called a *core*, and the core is easily constructed by sweeping a quad mesh shown in Figure 6-4 (a), Figure 6-4 (b), Figure 6-5 (a) or Figure 6-5 (b) four times.

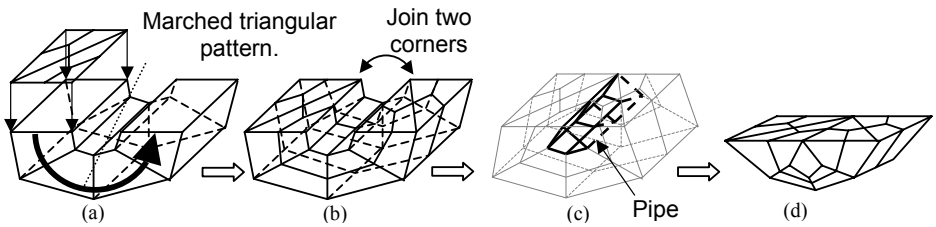


Figure 6-8 Construction of a triangular cap

6.3 Construction of HEXHOOP Templates for a Hex Element

6.3.1 Construction of a Standard Core

HEXHOOP's core has two *wing faces* and four *slots* as shown in Figure 6-9. Four slots are labeled as slot 0, slot 1, slot 2 and slot 3. The two wing faces and three vertical, cross-sectional faces are all rectangles. In order to dice such a core into a set of smaller hexahedrons two sets of parallel edges, vertical and horizontal, are subdivided into a same number of line segments. This will subdivide the two wing faces and three cross-sectional faces into a structured rectangular grid pattern. The number of sub-edges in horizontal direction is denoted as n_1, and the number of sub-edges in vertical direction is denoted as n_2 as shown in Figure 6-10. Then the number of subdivision of slot 0 and slot 1 is n_1, and slot 2 and slot 3 n_2. This type of core is called a $n_1 \times n_2$ *rectangular core*. An example of 2×4 rectangular core is shown in Figure 6-10 (a).

Another type of core is a *triangular core*, which has a triangular subdivision pattern on the two wing faces. Figure 6-10 (b) shows an example of such a core. Because this core has 4×4 triangular pattern on wing faces the core is called 4×4 *triangular core*. By choosing a different triangular subdivision pattern, shown in Figure 6-10 (a) and Figure 6-10 (b), and by applying it to the wing faces, different types of triangular cores such as a 4×2 triangular core and a 2×2 triangular core are created.

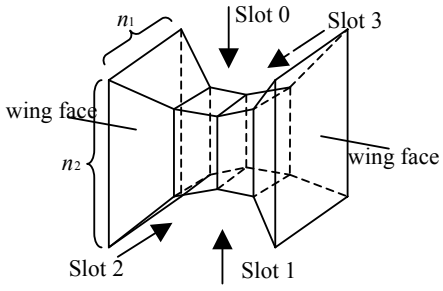
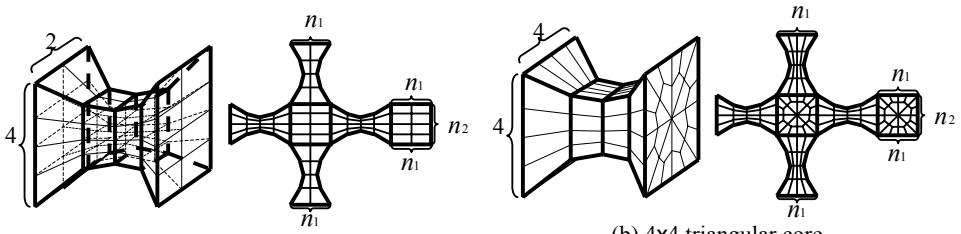


Figure 6-9 A core has two wing faces and four slots.



(a) 2x4 rectangular core (b) 4x4 triangular core

Figure 6-10 Rectangular core

6.3.2 Construction of a Cap

A cap has three types of faces: T-faces, B-faces, and F-faces, as shown in Figure 6-11. T-faces become a part of the exterior faces of a hexahedron. B-faces are connected to a core, and F-faces are connected to an adjacent cap. As illustrated in Figure 6-12, the characteristic of a cap is fully described by three factors:

- the number of subdivision, n_f, of the two edges shared by a T-face and a B-face
- the number of subdivision, n_s, of the four edges shared by a T-face and a F-face
- the type of the subdivision pattern of two T-faces, rectangular or triangular

A cap is either a $n_s \times n_f$ triangular cap or a $n_s \times n_f$ rectangular cap.

Any two caps with the same n_f can be connected each other with F-faces without losing the interface-conformity. Any cap whose n_s is the same as n_1 of the core can be connected to slot 0 or slot 1 of a core. Similarly any cap whose n_s is the same as n_2 of the core can be connected to slot 2 or slot 3. If T-faces

have a triangular pattern, a triangular face of a pyramid, tetrahedron or prism can be attached to these T-faces. On the other hand if T-faces have a rectangular pattern, a quadrilateral face of a hexahedron or a prism can be attached to these T-faces.

There are five types of caps that are practically most useful, illustrated in Figure 6-13. Since all caps shown in Figure 6-13 have $n_f = 4$, any of the caps shown in Figure 6-13 can be connected to each other without losing the mesh conformity.

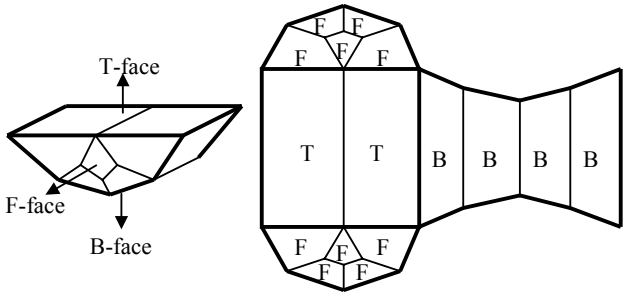
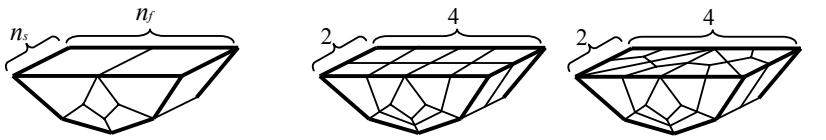


Figure 6-11 Faces of a Cap



Example 1: $n_s = 2$, $n_f = 4$ (2x4 rectangular cap)

Example 2: $n_s = 2$, $n_f = 4$ (2x4 triangular cap)

Figure 6-12 Subdivision of a Cap

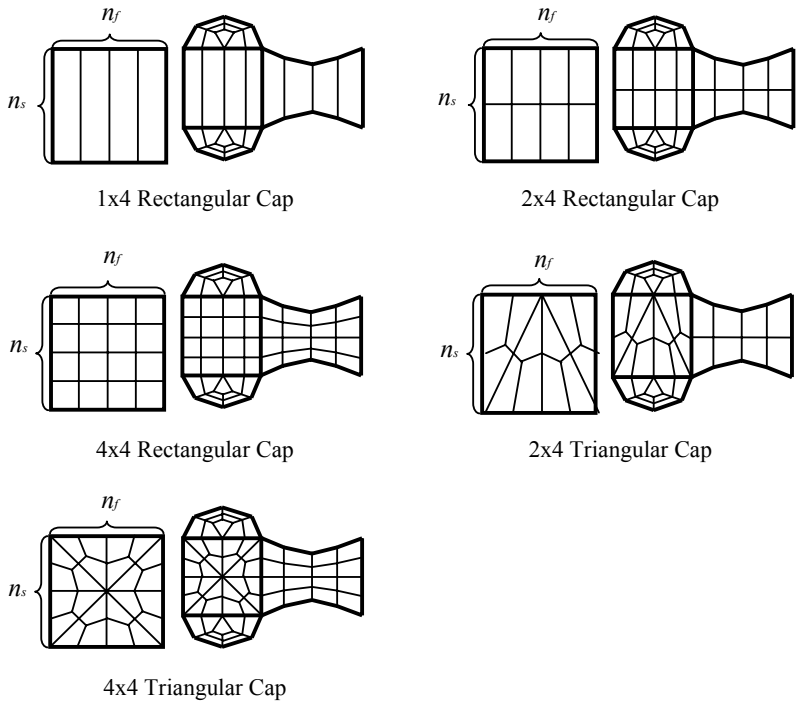


Figure 6-13 Five practically most useful caps

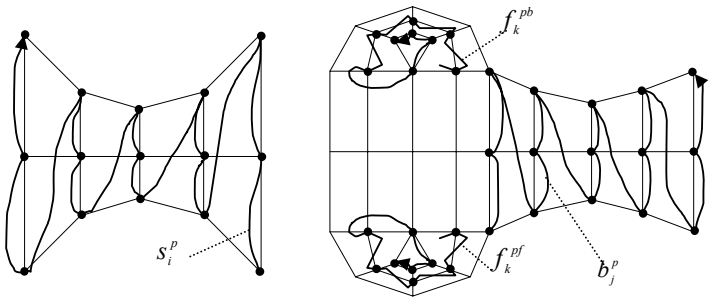
6.3.3 Assembly of a HEXHOOP Template

One core and four caps are assembled together to construct a complete HEXHOOP template for a hexahedron. In order to assemble a core and caps following node tables are necessary:

- nodes on each slot of the core
- nodes on B-face of a cap
- nodes on F-faces of a cap

The node tables of slot p, $0 \le p \le 3$, are denoted as s_i^p, nodes on the B-faces as b_j^p, nodes on the front F-faces as f_k^{pf} and nodes on the back F-faces as f_k^{pb}. n_1 is the number of subdivision of slot 0 and slot 1, and n_2 is the number of subdivision of slot 2 and slot 3. And for n_s of cap p is denoted as n_s^p and n_f of cap p as n_f^p.

All the nodes listed in the node tables must be ordered consistently according to the order illustrated in Figure 6-14. The length of table s_i^p is $5(n_1+1)$ for $p=0,1$ and $5(n_2+1)$ for $p=2,3$. The length of table b_j^p is $5(n_s^p+1)$. The length of f_k^{cf} and f_k^{cb} is $5\left(\frac{n_f^p}{2}-1\right)+4$.



Slot p of a core Cap p

Figure 6-14 Orders of nodes in node tables

Because cap p is connected to slot p, the following conditions must be satisfied:

Condition 1: $n_s^0 = n_s^1 = n_1, \quad n_s^2 = n_s^3 = n_2$

If the above conditions are not satisfied, the length of table s_i^p does not match the length of table b_j^p, and thus the cap and the slot cannot be connected.

Also, in order to form a hoop of caps, the following condition must be satisfied:

Condition 2: $n_f^0 = n_f^1 = n_f^2 = n_f^3$

If the above condition is not satisfied, the length of tables f_k^{pf} does not match corresponding f_k^{pb} (and vice-versa), and thus caps cannot be connected.

Now sufficient information to assemble a HEXHOOP template is prepared. After each cap is formed and oriented properly so that it fits its matching slot, nodes are joined together as shown in Figure 6-15.

It must be stressed again that the interior pattern of T-faces has nothing to do with this assembly process. The combination of caps can therefore be chosen arbitrarily as long as Conditions 1 and 2 are both satisfied.

This section has explained how to generate two types of standard cores, triangular cores and rectangular cores, and two types of caps, triangular caps and rectangular caps. By assembling these two types of cores and caps various all-hex templates for a hexahedron are obtained. This covers all the ten triangular and rectangular pattern combinations listed in Figure 6-6 except that shown in Figure 6-6 (g). This combination of three triangular patterned faces and three rectangular patterned faces require a non-standard core that has one triangular wing face and one rectangular wing face. Section 6.4.1 discusses how to construct such a core.

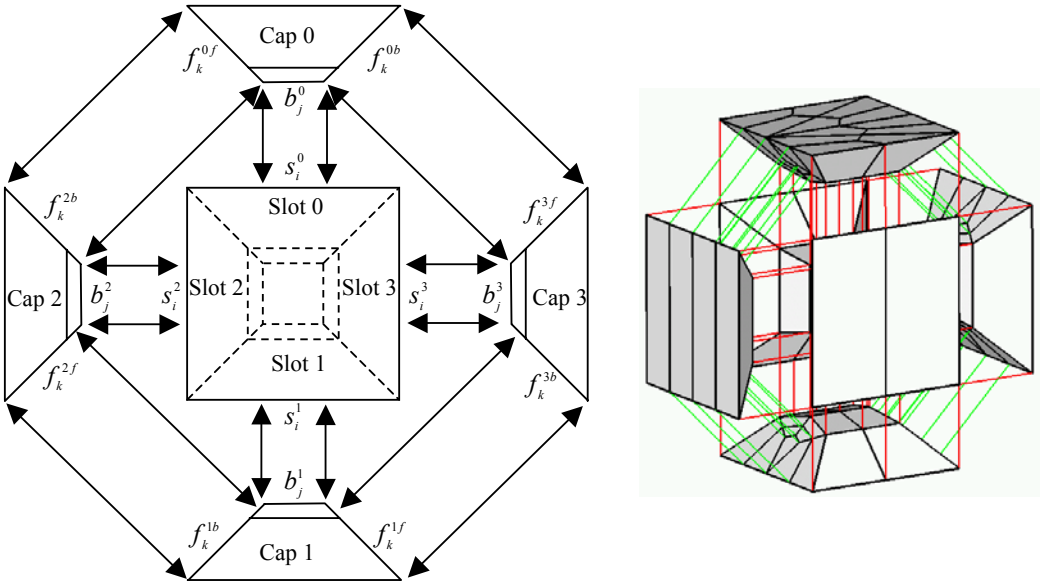


Figure 6-15 Joining nodes of a core and a cap

6.4 Variations of a Core

This section discusses two variations of a core. These non-standard cores are required in: (1) covering all the ten combinations of triangular and rectangular patterned faces shown in Figure 6-6, and (2) generating HEXHOOP templates for a prism.

6.4.1 Double Hoop Core

As pointed out earlier, a combination of three triangular patterned faces and three rectangular patterned faces, shown in Figure 6-6 (g), requires a non-standard core with one triangular patterned face and one

rectangular patterned face. Figure 6-16 shows an example of such a template with three 4×4 triangular patterns and three 4×4 rectangular patterns.

In order to make this new core with one triangular wing face and one rectangular wing face, first a HEXHOOP template with a 4×4 rectangular core, a 4×4 triangular cap and three 4×4 rectangular caps is assembled. If the assembled template is rotated 90 degree to the left, the HEXHOOP template as shown in Figure 6-17 (a) is obtained. Only one face of a template has a 4×4 triangular pattern and all other faces have a 4×4 rectangular pattern. Now, the center of the template is narrowed as shown in Figure 6-17 (b). The mesh then becomes a 4×4 core with only one wing face 4×4 triangular pattern. This core is called a 4×4 *double hoop core* because two hoops of caps exist in this template as shown in Figure 6-18.

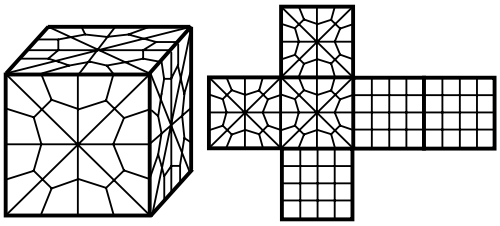


Figure 6-16 A combination of three 4x4 triangular patterns and three 4x4 rectangular patterns require a non-standard core with one triangular patterned face and one rectangular patterned face.

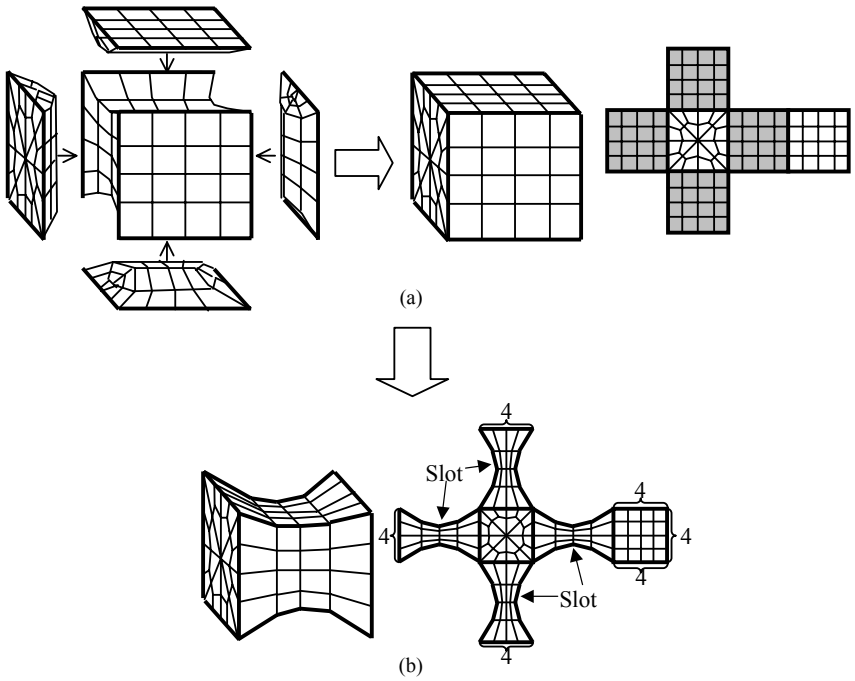


Figure 6-17 4x4 double hoop core

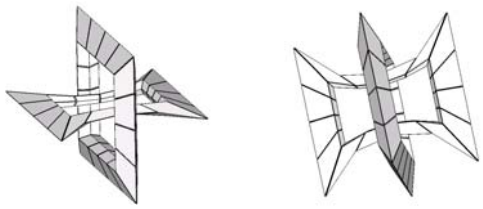


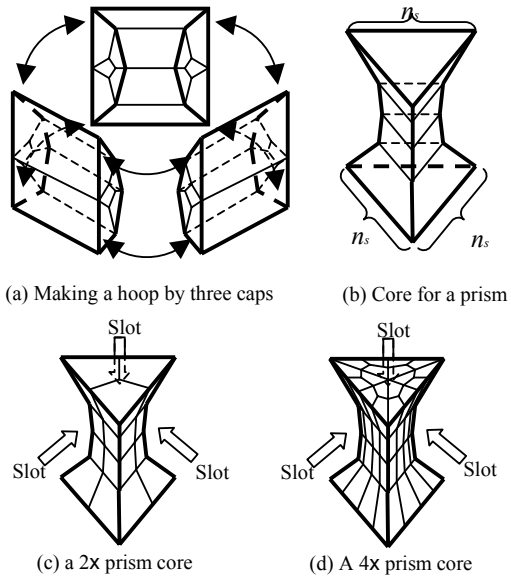
Figure 6-18 Two hoops of caps exist in a double hoop core (showing only pipes)

6.4.2 Core for a Prism Element

Up to this point HEXHOOP templates for a hexahedron have been discussed. HEXHOOP templates for a prism element can be developed in a similar way. Two differences, also illustrated in Figure 6-19, are: (1) a core's two wing faces for a prism are triangles, and (2) a hoop of caps for a prism consists of only three caps.

Two wing faces can be subdivided in various ways. Figure 6-19 (c) and Figure 6-19 (d) show two such examples. By sweeping the two wing faces' subdivision pattern throughout the core, we obtain an all-hex

mesh. Three $4 \times n$ caps can be attached to the core's three slots. Like the hexahedron templates, the subdivision pattern of each cap's T-faces can be either triangular or rectangular. The core shown in Figure 6-19 (c) is called a $2 \times$ *prism core* and the core shown in Figure 6-19 (d) is called a $4 \times$ *prism core*.



(a) Making a hoop by three caps (b) Core for a prism

(c) a 2x prism core (d) A 4x prism core

Figure 6-19 Core for a prism element

6.5 Two Solutions to Schneiders' Open Problem with HEXHOOP

This section provides two different solutions to Schneiders' Open Problem [84] using the HEXHOOP templates, one with a hexahedron-template and the other with a prism-template.

6.5.1 A Solution with a Hexahedron-Templates

In this first solution to Schneiders' Open Problem, a template shown in Figure 6-20 (a) is created first. This is a HEXHOOP template that combines a 2×1 rectangular core, two 1×4 rectangular caps, a 2×4 rectangular cap, and a 2×4 triangular cap.

Consider such a template whose bounding box is (-1.0,-1.0,-1.0) to (1.0,1.0,1.0). The template is then deformed by following coordinate transformation:

$$s_x = \frac{|2-y|}{2}$$

$$s_z = \frac{s_x \cdot |x+2|}{3}$$

$$d_y = \frac{1-|x| \cdot s_x}{5}$$

$$x' = x \cdot s_x$$

$$y' = \frac{y+1}{4} + d_y$$

$$z' = z \cdot s_z$$

This deformation transforms the HEXHOOP template to the shape shown in Figure 6-20 (b).

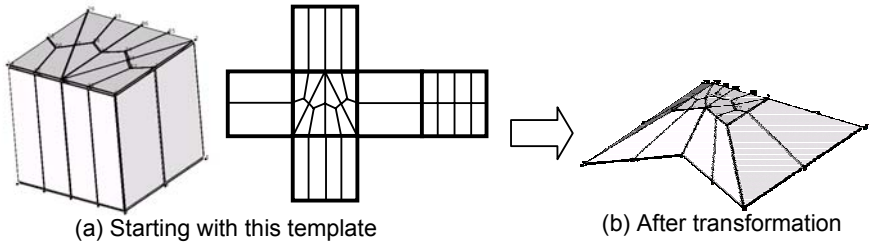


Figure 6-20 A HEXHOOP template for a hex

At this point, the bottom face has a 2x4 rectangular pattern as shown in Figure 6-21 (a); now we attach a diced prism from the bottom as shown in Figure 6-21 (b). This gives a mesh shown in Figure 6-21 (c)

After the diced prism is attached to the bottom, the bottom face becomes 2x2 rectangular pattern; this is the pattern that Schneiders' Open Problem demands. Next, we attach two more diced prisms as shown in Figure 6-22 (a) from the front and from the back.

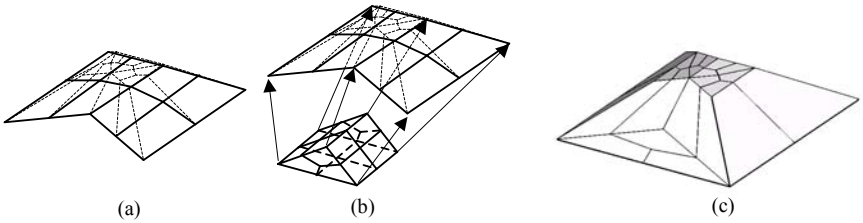


Figure 6-21 Bottom face of the deformed template

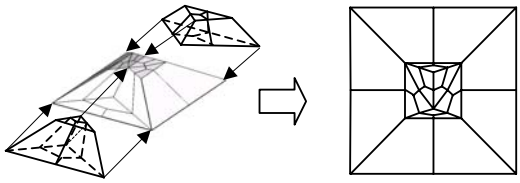


Figure 6-22 Attaching two diced prisms from the front and from the back

At this point, the top portion of the mesh shows the pattern illustrated in Figure 6-22 (b). The inner square has a triangular pattern. Thus, obviously we can attach six diced tetrahedrons on the inner square. A tetrahedral mesh as shown in Figure 6-23 is created, diced it into hexahedrons, and attached to the inner square of the current mesh.

Now, the mesh shows the pattern illustrated in Figure 6-24. Finally, adding four diced prisms from the four sides as shown in Figure 6-25 gives a solution to Schneiders' Open Problem. Because each interior face is always shared by two hexahedral elements during the above procedure, the mesh is valid.

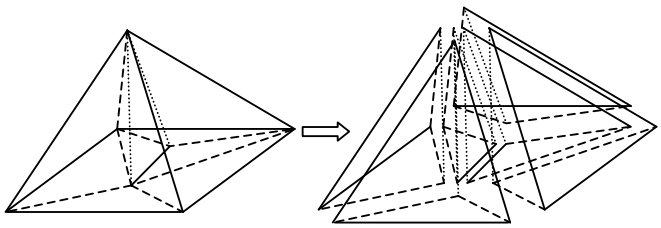


Figure 6-23 Tet mesh which is attached to the center square

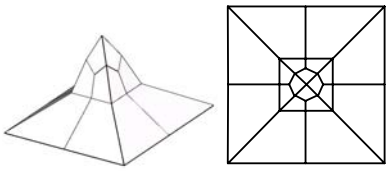


Figure 6-24 After a diced tet mesh is attached

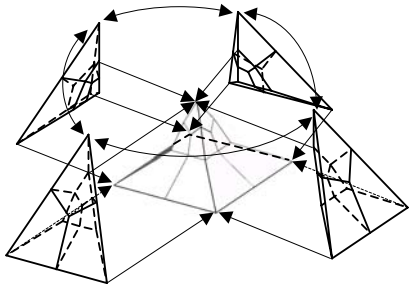


Figure 6-25 A solution to Rob Schneiders' Open Problem

6.5.2 A Solution with a Prism-Templates

There exists another solution based on a prism HEXHOOP. A $2\times$ prism core, a 2×4 rectangular cap and two 2×4 triangular caps are first combined to make a prism-template as shown in Figure 6-26 (a), and the template is oriented and scaled to the size shown in Figure 6-26 (b)

This template is then transformed with the following coordinate transformation:

$$xx = x(1-0.5y)$$
$$yy = \frac{y}{4}$$
$$zz = z$$
$$x' = xx$$
$$y' = yy + \frac{|1-xx|}{5}$$
$$z' = zz(\frac{|xx|}{4} + 0.75)$$

This gives a mesh shown in Figure 6-27.

Next, a diced prism is attached to the mesh from the bottom as shown in Figure 6-27. After attaching the diced prism, the bottom face becomes a 2×2 rectangular pattern, which Schneiders' Open Problem demands.

Now the exterior surface of the top portion looks like Figure 6-28 (a). The topology of the exterior surface is exactly same as a quadrilateral mesh derived from a triangular mesh shown in Figure 6-28 (b).

Now a point and a triangular mesh shown in Figure 6-28 (b) are connected together to create a tetrahedral mesh as shown in Figure 6-28 (c).

The tetrahedral mesh is then diced, and nodes are moved so that they fit the mesh shown in Figure 6-28 (a). Finally, by merging the diced tet mesh onto the mesh shown in Figure 6-28 (a), another solution to Schneiders' Open Problem is completed.

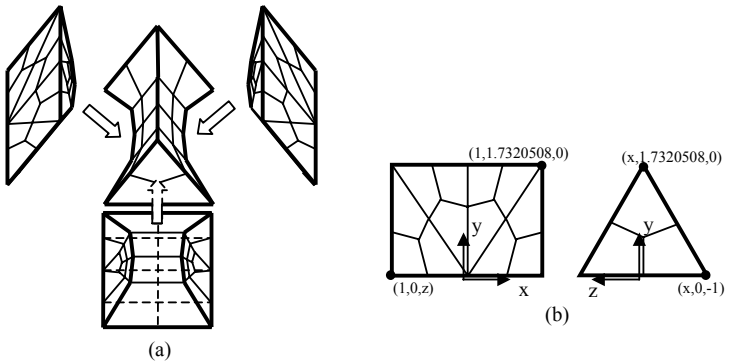


Figure 6-26 A HEXHOOP template for a prism

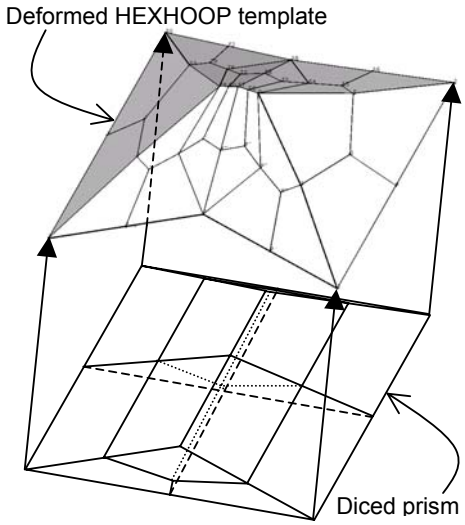
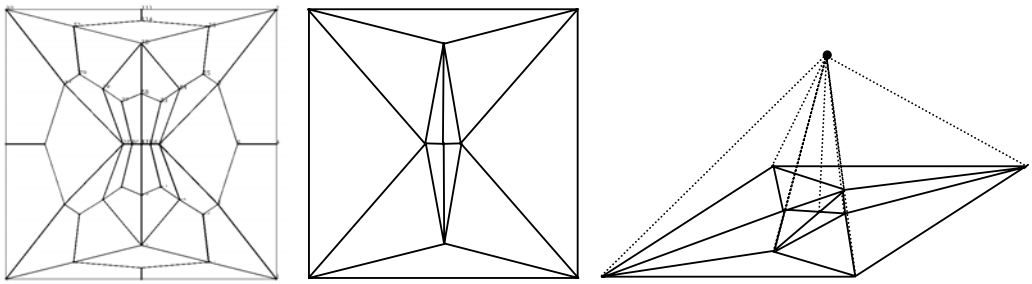


Figure 6-27 Attaching a diced prism from the bottom



(a) (b) (c)

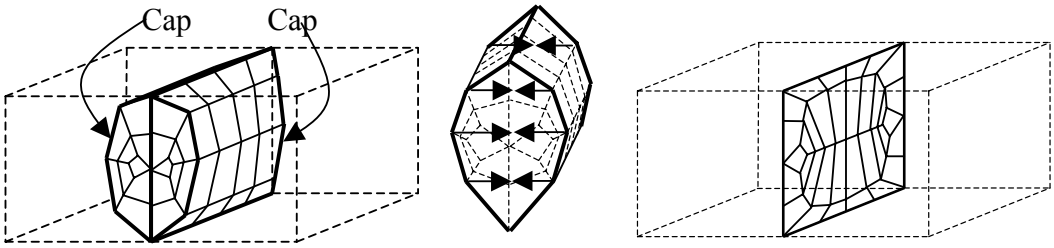
Figure 6-28 Equivalent triangular mesh and tetrahedral mesh

6.6 Post Processes

This section presents three types of post-processes; one of them reduces the number of elements while maintaining topological conformity, and the other two improve the geometric quality of the all-hex mesh created by the proposed templates.

6.6.1 Cap Suppression

Cap suppression reduces the number of hexahedrons by deleting a pair of caps that are sharing T-faces when two HEXHOOP templates are adjacent as shown in Figure 6-29 (a). Since the two caps have identical n_f and n_s, the left-hand side and the right-hand side of the caps have the identical subdivision pattern. Hexahedrons included in both caps can be eliminated by joining corresponding nodes as shown in Figure 6-29 (b). The subdivision pattern between the two HEXHOOP templates is shown in Figure 6-29 (c). $(4n_f n_s + 2n_s)$ elements are reduced by this operation. Since the quality of the elements in cap modules is relatively lower than other elements, cap elimination improves the overall quality of the all-hex mesh as well.



(a) Cap modules facing each other (b) Joining corresponding nodes (c) After cap suppression

Figure 6-29 Cap suppression

6.6.2 Volume Equalization

Volume equalization moves one node at a time and attempts to improve the smallest Jacobian determinant. Although Jacobian determinant is not directly related to the shape quality of the element, larger Jacobian determinant is desired in general, and Jacobian determinant must be positive everywhere in a mesh to have a valid solution in finite element analysis.

As explained in Section 3.2.2, Jacobian determinant of element e at node $\mathbf{n}$, $J_e(\mathbf{n})$, is a function of the location of the node and is defined as:

$$J_e(\mathbf{n}) = (\mathbf{m}_{e2} - \mathbf{m}_{e1}) \cdot (\mathbf{m}_{e3} - \mathbf{m}_{e1}) \times (\mathbf{n} - \mathbf{m}_{e1}) ,$$

where $\mathbf{m}_{e1}$, $\mathbf{m}_{e2}$ and $\mathbf{m}_{e3}$ are three nodes of element e directly connected to $\mathbf{n}$, as illustrated in Figure 6-30. $\mathbf{m}_{e1}$, $\mathbf{m}_{e2}$ and $\mathbf{m}_{e3}$ are ordered so that $J_e(\mathbf{n}) = 1$ when element e is a unit cube. $J_e(\mathbf{n})$ is also called as a local volume at $\mathbf{n}$, and it is proportional to the signed volume of a tetrahedron T shown in Figure 6-30. Since more than one element may share node $\mathbf{n}$, multiple Jacobian determinant values may exist at node $\mathbf{n}$. These multiple values are denoted as $J_{ei}(\mathbf{n})$, where e_i is the ith element sharing node $\mathbf{n}$.

If node $\mathbf{n}$ is moved to the new location such that $\sum_i J_{ei}(\mathbf{n})^2$ is minimized, the values of $J_{ei}(\mathbf{n})$ become as even as possible. Consequently, the minimum value of $J_{ei}(\mathbf{n})$ is expected to be improved.

To minimize $\sum_i J_{ei}(\mathbf{n})^2$ at point $\mathbf{n} = (x, y, z)$ the following optimality conditions must be satisfied.

$$\frac{\partial}{\partial x}\sum_i J_{ei}(\mathbf{n})^2 = \frac{\partial}{\partial y}\sum_i J_{ei}(\mathbf{n})^2 = \frac{\partial}{\partial z}\sum_i J_{ei}(\mathbf{n})^2 = 0 \ .$$

These three simultaneous equations turn out three linear equations of three unknowns, x, y and z. The new location of node $\mathbf{n}$ is computed by solving these simultaneous linear equations.

Although the volume equalization usually improves the mesh quality, it may worsen the quality when the node distribution is not uniform or when high aspect ratio elements are present. When volume equalization does not improve the quality, exhaustive method is applied, as explained in the next section.

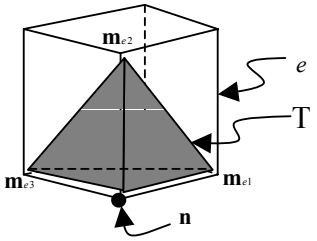


Figure 6-30 Jacobian determinant

6.6.3 Exhaustive Method

The exhaustive method moves one node at a time to make the minimum scaled Jacobian larger. As explained in Section 3.2.2, the scaled Jacobian of element e at node $\mathbf{n}$ is defined as:

$$Js_e(\mathbf{n}) = \frac{J_e(\mathbf{n})}{\|\mathbf{m}_{e1}-\mathbf{n}\|\|\mathbf{m}_{e2}-\mathbf{n}\|\|\mathbf{m}_{e3}-\mathbf{n}\|} \ ,$$

where $\|\mathbf{v}\|$ is the length of vector $\mathbf{v}$.

Scaled Jacobian $Js_e(\mathbf{n})$ becomes the maximum value of 1.0 only when the solid angle of element e at node $\mathbf{n}$ is identical to the solid angle of a cube. Thus, the solid angles of the hex elements will be improved by maximizing the scaled Jacobian.

The exhaustive method tests 27 candidate locations of $\mathbf{n}$:

$$\mathbf{n}_c = \mathbf{n} + l\frac{\mathbf{d}}{\|\mathbf{d}\|} \ , \ \mathbf{d} = (dx, dy, dz) \ , \ dx, dy, dz = \{-1,0,1\} \ , \ l = \frac{\text{average length of edges connected to } \mathbf{n}}{20.0}$$

and moves node $\mathbf{n}$ to the one that maximizes the minimum of $Js_{ei}(\mathbf{n}), Js_{ei}(\mathbf{m}_{ei1}), Js_{ei}(\mathbf{m}_{ei2}), Js_{ei}(\mathbf{m}_{ei3})$. $Js_{ei}(\mathbf{m}_{ei1})$, $Js_{ei}(\mathbf{m}_{ei2})$ and $Js_{ei}(\mathbf{m}_{ei3})$ are taken into account because the location of $\mathbf{n}$ affects the values of $Js_{ei}(\mathbf{m}_{ei1})$, $Js_{ei}(\mathbf{m}_{ei2})$ and $Js_{ei}(\mathbf{m}_{ei3})$. If $\mathbf{d} = (0,0,0)$ is chosen, the node is not moved. Choice of l is based on the experiments in the current implementation.

An alternative to the exhaustive method is mesh optimization. Knupp presents a technique that improves the quality of a mesh based on numerical optimization scheme [68]. Freitag et al. present a method that untangles a quadrilateral mesh based on linear programming [71], and their method can be extended to an all-hex mesh. These techniques move node locations to improve the quality of a mesh. Although the volume equalization scheme and the exhaustive method give a good improvement of mesh quality, the optimization-based methods potentially give even better results. The effect of those methods for an all-hex mesh created with the HEXHOOP templates is still unknown and needs future research.

6.7 Qualities of Hexahedrons Included in the HEXHOOP Templates

The qualities of hexahedrons included in the output all-hex mesh depend on the qualities of the hexahedrons included in the HEXHOOP templates. During the mesh conversion process, the HEXHOOP template is deformed so that the exterior shape fits the hexahedron or the prism to be replaced with the template, and the hexahedrons included in the template are deformed accordingly. If ill shaped hexahedrons are included in the template, the ill shaped hexahedrons will be even more ill shaped after the deformation. This section shows the qualities of the hexahedrons included in the typical HEXHOOP templates.

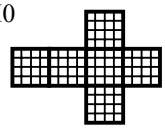
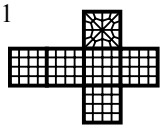
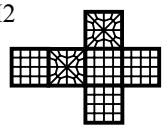
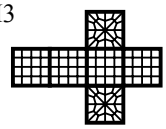
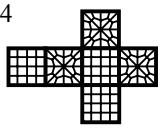
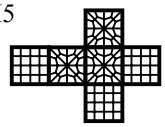
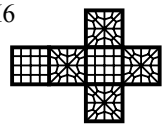
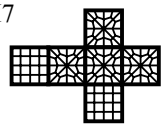
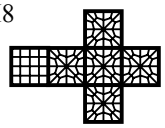
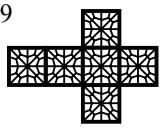
There must be at least ten templates for a hexahedron and four templates for a prism. A hexahedron has six faces, and each of them can be subdivided into either a rectangular pattern of a triangular pattern. The number of total combination is thus $2^6 = 64$. However, because of symmetry, the number of necessary templates is reduced to ten. A prism has three quadrilateral faces and each of them can be subdivided into either a rectangular pattern or a triangular pattern. The number of total combination is thus $2^3 = 8$. However, because of symmetry, the number of necessary templates is reduced to four.

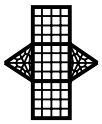
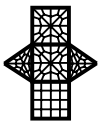
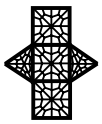
There is also a freedom in the choice of n_f, n_1 and n_2. To have a triangular pattern, the minimum n_f is four. If $n_f = 4$ is chosen, the easiest choice of n_1 and n_2 is $n_1 = n_2 = 4$, which makes the triangular pattern that appears on the quadrilateral face of the template symmetric, and the choice $n_f = n_1 = n_2 = 4$ makes the templates easy to apply. Thus, the qualities of the thirteen templates, ten hexahedron-templates and three prism templates, with $n_f = n_1 = n_2 = 4$ are presented in Table 6-1. The qualities of the templates are measured after applying smoothing schemes presented in Section 6.6. Qualities are presented in minimum scaled Jacobian, average scaled Jacobian, maximum aspect ratio and average aspect ratio. The definition of scaled Jacobian is explained in Section 3.2.2, and aspect ratio in Section 3.2.3, and the best value of scaled Jacobian is 1.0 and the worst value of scaled Jacobian is –1.0, and the best value of aspect ratio is 1.0, and the worst value of aspect ratio is ∞.

Hexahedrons of the hexahedron-templates except H5 are well shaped. Minimum scaled Jacobian of them are more than 0.27, and maximum aspect ratio is less than 4.0. Thus, if the original hexahedron to be replaced with one of these templates is well shaped, minimum scaled Jacobian is likely to be more than 0.27, and maximum aspect ratio is likely to be less than 3.9. Hexahedrons of template H5 are not as well shaped as other templates because the minimum scaled Jacobian is about 30% smaller than that of the others, and the maximum aspect ratio is about 17% larger than that of the others. Template H5 is using double-hoop core, explained in Section 6.4.1. Although the qualities of hexahedrons of template H5 are not too ill shaped, if the input hex-dominant mesh is built so carefully that it is unnecessary to use template H5, the quality of the output all-hex mesh will be improved.

Hexahedrons of the prism-templates are not as well shaped as hexahedrons of the hexahedron-templates. Average scaled Jacobian of the prism-templates are lower than that of any hexahedron-templates, and the average aspect ratio of prism-templates are larger than that of any hexahedron-templates. Thus, it is expected that qualities of hexahedrons created by applying prism-templates will be lower than the qualities of those created by applying hexahedron-templates, and if the prism to be replaced with a prism-template is not sufficiently well shaped, subdividing the prism into tetrahedron and applying tetrahedron-templates possibly create a better hexahedrons. One possible solution to this problem is to create the input hex-dominant mesh with only hexahedrons and tetrahedrons.

Table 6-1 Qualities of hexahedrons included in the HEXHOOP templates

H0	H1	H2	H3	H4
				
N/A (Does not need a template)	(1) 0.29 (2) 0.64 (3) 3.83 (4) 1.94	(1) 0.29 (2) 0.60 (3) 3.88 (4) 1.96	N/A (Does not need a template)	(1) 0.30 (2) 0.58 (3) 3.86 (4) 1.99
H5	**H6**	**H7**	**H8**	**H9**
				
(1) 0.20 (2) 0.51 (3) 4.56 (4) 2.18	(1) 0.28 (2) 0.55 (3) 3.74 (4) 2.01	(1) 0.28 (2) 0.55 (3) 3.89 (4) 2.15	(1) 0.28 (2) 0.52 (3) 3.88 (4) 2.14	(1) 0.28 (2) 0.50 (3) 3.77 (4) 2.14

P0	P1	P2	P3	
				(1) Minimum scaled Jacobian (2) Average scaled Jacobian (3) Maximum aspect ratio (4) Average aspect ratio
N/A (Does not need a template)	(1) 0.27 (2) 0.47 (3) 5.41 (4) 2.75	(1) 0.27 (2) 0.46 (3) 5.36 (4) 2.73	(1) 0.27 (2) 0.45 (3) 5.36 (4) 2.73	

6.8 Results

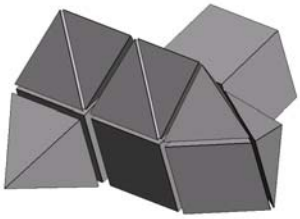
This section shows some examples of the conversion from a hex-dominant mesh to an all-hex mesh. The examples shown in this section are tested by a program that confirms that they are topologically valid, and that there are no gaps and overlaps, and that all scaled Jacobians are positive. Quality of each example is shown by using two measurements, the histogram of minimum scaled Jacobian and the histogram of aspect ratio. In each example, the number of templates applied to the hex-dominant mesh is presented for each type of template.

Figure 6-31 (a) shows a hex-dominant mesh, which consists of 3 hexahedrons, 1 prism, 4 pyramids, and 3 tetrahedrons, and Figure 6-31 (b) shows a resultant all-hex mesh consisting of 542 hexes. Figure 6-31 (c) and Figure 6-31 (d) show cross-sections after the conversion. The hexahedrons that is adjacent to two pyramids and a hexahedron is replaced with a HEXHOOP template made from a 4x2 rectangular core, a 2x4 triangular cap, a 4x4 triangular cap, a 2x4 rectangular cap, and a 4x4 rectangular cap. The hexahedron

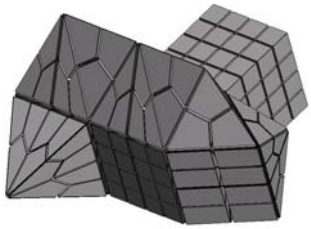
that is adjacent to a pyramid, two hexahedrons, and a prism is replaced with a HEXHOOP template made from a 4x2 rectangular core, a 2x4 triangular cap, a 2x4 rectangular cap, and two 4x4 rectangular caps. The prism that is adjacent to a hexahedron and a pyramid is replaced with a HEXHOOP template made from a 2x prism core, a 2x4 triangular cap, and two 2x4 rectangular caps. Pyramids and tetrahedrons are diced accordingly.

Examples of more general geometries are also presented to show the applicability of the HEXHOOP templates. Figure 6-32 shows a hex-dominant mesh and an all-hex mesh of a geometric domain made by combining three octagonal prisms. The input hex-dominant mesh for this example is manually created. Figure 6-33 shows a hex-dominant mesh and an all-hex mesh of a geometric domain with a fillet, and Figure 6-34 shows a hex-dominant mesh and an all-hex mesh of a mechanical part. The input hex-dominant meshes for these examples are created by the method presented in Chapter 5 and consists of only hexahedrons and tetrahedrons. Figure 6-35 shows a hex-dominant mesh and an all-hex mesh of a left portion of an airplane. The input hex-dominant mesh for this example is manually created. Hexahedrons and prisms that are adjacent to a pyramid are replaced with HEXHOOP templates of $n_1 = n_2 = n_f = 4$, and the other elements are subdivided accordingly.

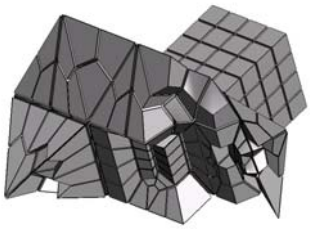
These examples show high quality in terms of the aspect ratio; every histogram has a sharp peak at 1.0. Every histogram of scaled Jacobian has two peaks at around 0.4 and 1.0. A peak around 0.4 is a result of non-hex elements in the input hex-dominant meshes. This observation is confirmed by comparing the result shown in Figure 6-32 and the result shown in Figure 6-35. The hex-dominant mesh shown in Figure 6-32 (a) has 87% of non-hex elements, and the histogram shown in Figure 6-32 (e) has a clear peak at around 0.4. In contrast, the hex-dominant mesh shown in Figure 6-35 (a) has only 27% of non-hex elements, and the histogram shown in Figure 6-35 (e) has no clear peak at 0.4. This indicates that the proposed templates create a higher-quality all-hex mesh if the input hex-dominant mesh has fewer non-hex elements.



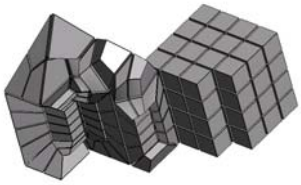
(a) A hex-dominant mesh
(3 hexahedrons, 1 prism, 4 pyramids,
and 3 tetrahedrons)



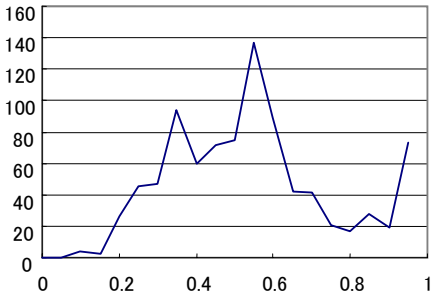
(b) An all-hex mesh
(542 hexahedrons)



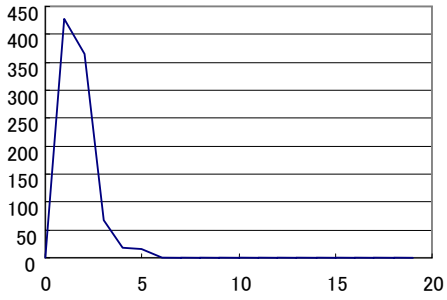
(c) A cross-section of the all-hex mesh



(d) A cross-section of the all-hex mesh



(e) Histogram of scaled Jacobian



(f) Histogram of aspect ratio

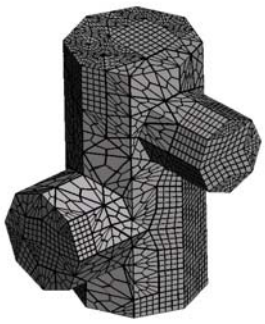
H0	1	H4	0	H8	0	P0	0
H1	1	H5	0	H9	0	P1	1
H2	1	H6	0			P2	0
H3	0	H7	0			P3	0

(g) Number of templates applied

Figure 6-31 An example of the conversion from a hex-dominant mesh to an all-hex mesh

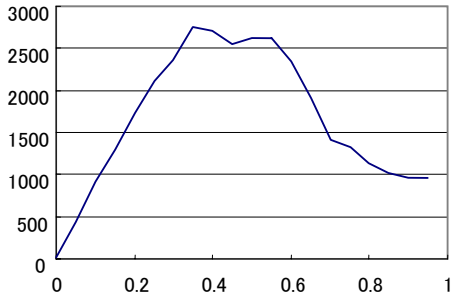
(a) A hex-dominant mesh of an object consisting of three circular bars (93 hexahedrons, 127 prisms, 100 pyramids, and 417 tetrahedrons)

(b) A cross-section of the hex-dominant mesh

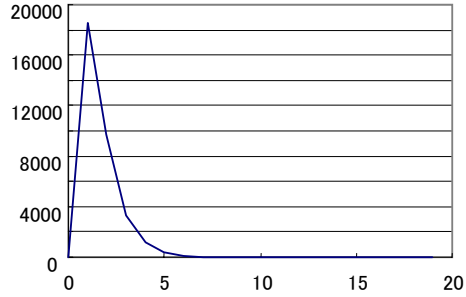


(c) An all-hex mesh, which is converted from (a) (35,120 hexahedrons)

(d) A cross-section of the all-hex mesh



(e) Histogram of scaled Jacobian



(f) Histogram of aspect ratio

H0	46	H4	0	H8	0	P0	94
H1	35	H5	1	H9	0	P1	31
H2	10	H6	0			P2	2
H3	1	H7	0			P3	0

(g) Number of templates applied

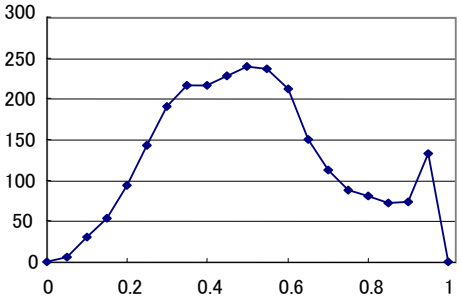
Figure 6-32 A hex-dominant mesh and an all-hex mesh of an object consisting of three circular bars

(a) Input hex-dominant mesh
(101 hexahedrons and 315 tetrahedrons)

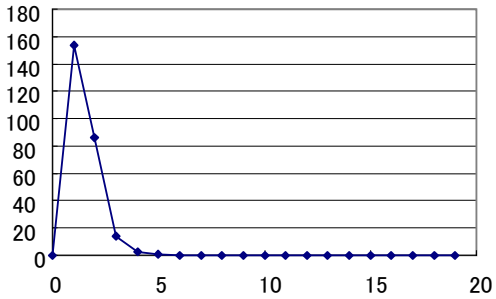
(b) Input hex-dominant mesh (cross-section)

(c) Output all-hex mesh of a fillet shape
(25,756 hexahedrons)

(d) Output all-hex mesh (cross-section)



(e) Histogram of scaled Jacobian

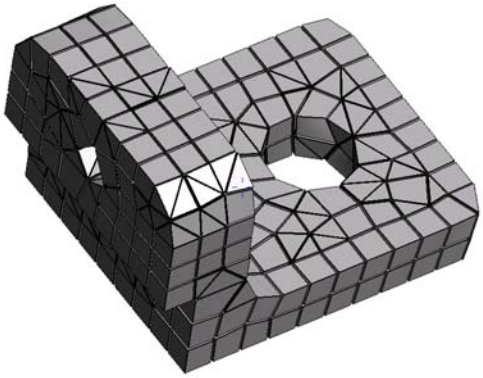


(f) Histogram of aspect ratio

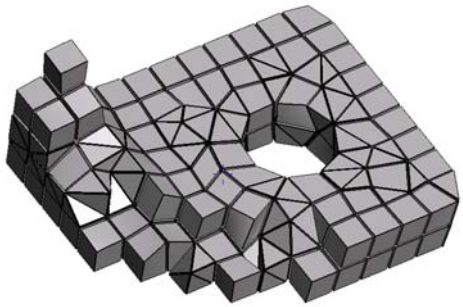
H0	26	H4	5	H8	2	P0	0
H1	37	H5	8	H9	0	P1	0
H2	15	H6	0			P2	0
H3	2	H7	6			P3	0

(g) Number of templates applied

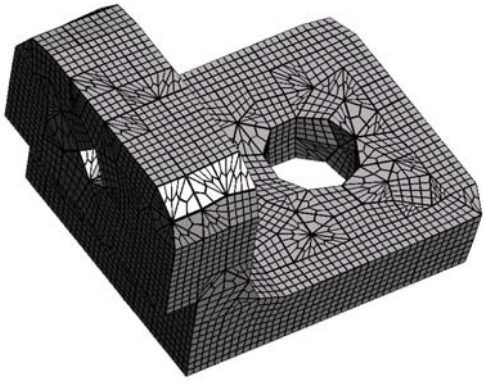
Figure 6-33 A hex-dominant mesh and an all-hex mesh of a fillet shape and statistics



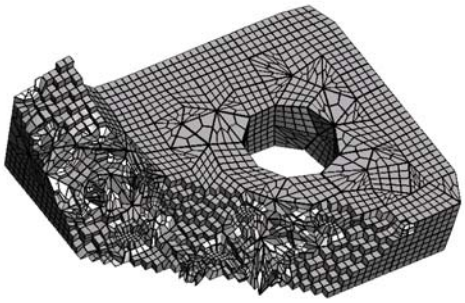
(a) A hex-dominant mesh of a mechanical part
(225 hexahedrons and 582 tetrahedrons)



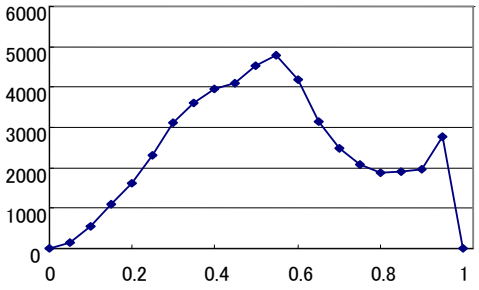
(b) A cross-section of the hex-dominant mesh



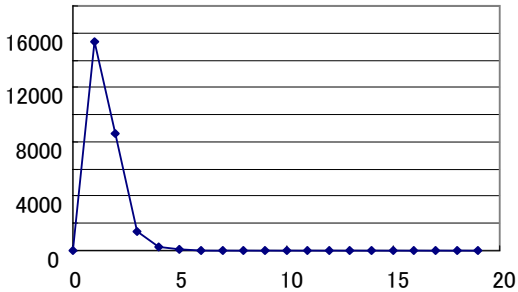
(c) An all-hex mesh of the mechanical part
(50,124 hexes)



(d) A cross-section of the all-hex mesh



(e) Histogram of scaled Jacobian

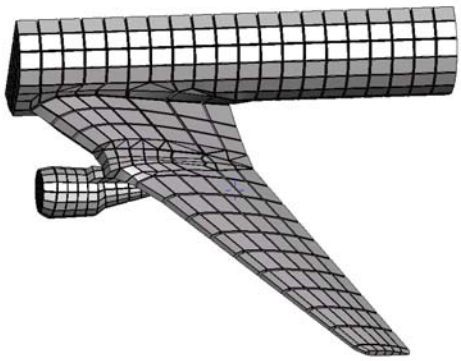


(f) Histogram of aspect ratio

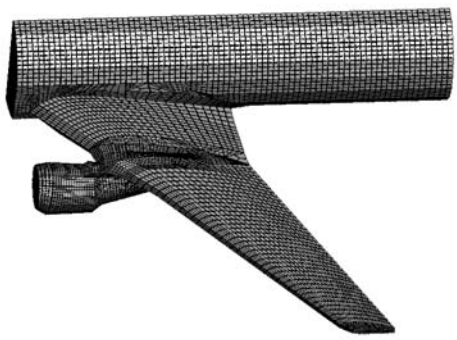
H0	69	H4	5	H8	3	P0	0
H1	81	H5	8	H9	0	P1	0
H2	44	H6	1			P2	0
H3	7	H7	7			P3	0

(g) Number of templates applied

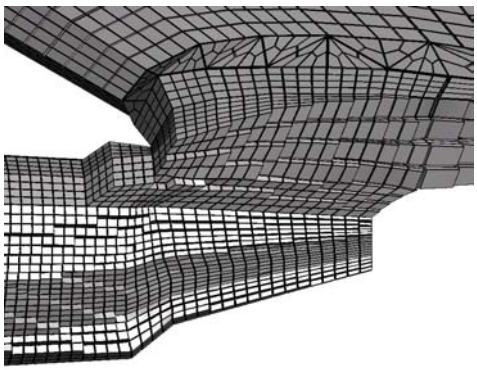
Figure 6-34 A hex-dominant mesh and an all-hex mesh of a mechanical part



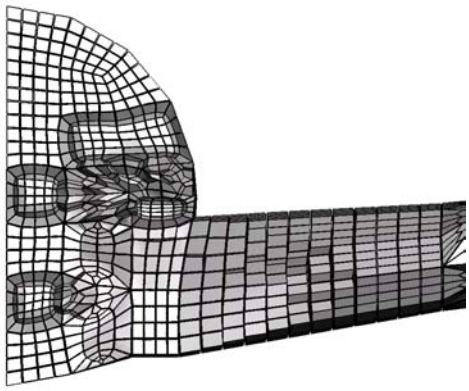
(a) A manually-created hex-dominant mesh of a left portion of an airplane
(764 hexahedrons, 277 tetrahedrons and 12 prisms)



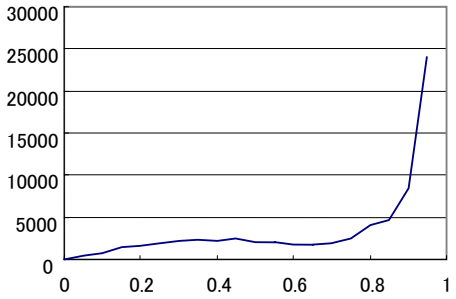
(b) An all-hex mesh of the portion of the airplane
(68344 hexahedrons)



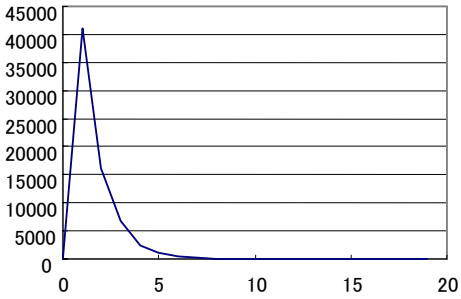
(c) A cross-section of the all-hex mesh



(d) A cross-section of the all-hex mesh



(e) Histogram of scaled Jacobian



(f) Histogram of aspect ratio

H0	658	H4	1	H8	0
H1	91	H5	0	H9	0
H2	13	H6	1		
H3	0	H7	0		

P0	12
P1	0
P2	0
P3	0

(g) Number of templates applied

Figure 6-35 A left portion of an airplane

6.9 Experimental Finite Element Analyses

Experimental finite element analyses of a bending problem are performed to test the effect of the HEXHOOP templates on the accuracy of the finite element analysis, and the results show that the all-hex meshes created with HEXHOOP templates give accurate solutions if non-hex elements in the input hex-dominant mesh are located far from the critical region. The target geometry used in the test is an L-shaped beam with rectangular cross-section. The length of the beam is four, and the thickness is uniformly one. The height of the beam is two from the left end to a cross-section a distance of one from the left end, and the height is one for the remainder. The left end of the beam is cantilevered, and a downward force is applied at the four corner nodes of the right end. The geometry of the beam and the loading condition are also illustrated in Figure 6-36. Accuracy of the finite element analysis is measured based on the total displacement of the four corner nodes on the right end of the beam.

To measure the accuracy, the reference value of the total displacement of the four corner nodes on the right end, which is reasonably close to the exact solution, must be found. Since the solution obtained by the finite element analysis tends to converge near the exact solution as the mesh resolution becomes finer, the reference value is found by analyzing the convergence of the total displacement of the four corner nodes of the right end of the beam, and is examined with eight all-hex meshes of different resolutions. The meshes used in the convergence analysis are created as follows. The L-shaped beam is first meshed into a mesh with five hexahedrons, where each of them is a unit cube aligned with the length-wise direction, the height-wise direction and the thickness-wise direction. Four of them are placed in a row left to right, and the other one is placed on top of the left-most hexahedron. Each of the five hexahedrons is then subdivided into N sections in each direction (thickness-wise, length-wise, and height-wise) to increase the resolution of the mesh. For the convergence analysis, $N=2$, $N=4$, $N=6$, $N=8$, $N=10$, $N=12$, $N=14$ and $N=16$ are chosen. The meshes are fed to ANSYS (http://www.ansys.com), and it solves the bending problem. Obtained total displacements of the four nodes at the right end are plotted in Figure 6-37 against the number of elements, and the plot shows that the total displacement of the four corner nodes at the right end converges to somewhere above 2.2. To facilitate the comparison, the value 2.25 is taken as the reference value of the total displacement of the four corner nodes at the right end, and it is taken as a

reference value to compute % error in the tests of the effect of the HEXHOOP templates on the accuracy of the finite element analysis.

To test the effect of the HEXHOOP templates on the accuracy of the finite element analysis, the solutions to the bending problem of the L-shaped beam are obtained with ANSYS for four different all-hex meshes: (1) the mesh created for the convergence analysis as described above with $N = 4$, (2) an all-hex mesh converted from a hex-dominant mesh that has tetrahedrons between the right end and the cross-section to the left of the right end by the distance of one, (3) an all-hex mesh converted from a hex-dominant mesh that has tetrahedrons between the cross-section to the left of the right end by the distance of one and the cross-section to the right of the left end by the distance of two, and (4) an all-hex mesh converted from a hex-dominant mesh that has tetrahedrons between the cross-section to the left of the right end by the distance of two and the cross-section to the right of the left end by the distance of one. To create meshes (2), (3) and (4), the HEXHOOP templates with $n_f = n_1 = n_2 = 4$ replace the hexahedrons of the hex-dominant mesh, and the tetrahedrons in the hex-dominant mesh are subdivided accordingly. Since all hexahedrons in mesh (1) are cubes, the quality of mesh (1) is the highest among the four meshes. Although the numbers of elements are different, the resolutions of the four meshes are almost identical in terms of edge length. The average edge lengths of meshes (1), (2), (3) and (4) are 0.25, 0.22, 0.20 and 0.20 respectively. Table 6-2 illustrates the hex-dominant meshes, the all-hex meshes, the distributions of 1st principal stress, and the magnified images of the distributions of 1st principal stress near the stress concentration. The qualities and the numbers of elements of the all-hex meshes and the total displacements of the four corner nodes of the right end obtained with the all-hex meshes are tabulated in Table 6-3.

The total displacements of the four corner nodes at the right end, tabulated in Table 6-3, indicate that the accuracy of the solution depends on the location of the non-hex elements of the hex-dominant mesh. The total displacement of the four corner nodes at the right end obtained with the mesh (2) has 22.2% error against the reference value, while the one obtained with the meshes (3) and (4) has 21.8% error and 20.4% error respectively. Since the load is applied at the right end, and the beam is cantilevered at the left end, the displacement is small on the left and large on the right. Also, the obtained total displacement of the four corner nodes of the right end show larger error if tetrahedrons of the input hex-dominant mesh is located closer to the right end. The results imply that if tetrahedrons of the input hex-dominant mesh are located in

the region of smaller displacement, the error of the solution obtained with the all-hex mesh converted from the input hex-dominant mesh becomes smaller.

Although mesh (4) shows smaller error in the total displacement of the four nodes at the right end than meshes (2) and (3), the distribution of the 1st principal stress near the stress concentration obtained with mesh (4) is inaccurate. The contour plots of 1st principal stress near the stress concentration obtained with the four meshes are shown in Table 6-3, and the plots obtained with meshes (1), (2), and (3) show almost identical contours around the stress concentration. However, the plot obtained with mesh (4) shows jaggy contours near the stress concentration. Since the quality of mesh (1) is the best among the four meshes, it is reasonable to assume that the solution obtained with mesh (1) is a reasonably good solution with this mesh resolution. Thus, meshes (2) and (3) give reasonably good stress distribution for this mesh resolution, but the stress distribution obtained with mesh (4) is not accurate. The plots indicate that the all-hex mesh converted from the hex-dominant mesh with the HEXHOOP templates give reasonably accurate distribution of 1st principal stress near the stress concentration if tetrahedrons are located far from the stress concentration.

The results imply that an all-hex mesh converted from a hex-dominant mesh with the HEXHOOP templates yields an accurate solution if non-hex elements in the hex-dominant mesh are located far from the critical region, such as a region with large displacement or with a stress concentration.

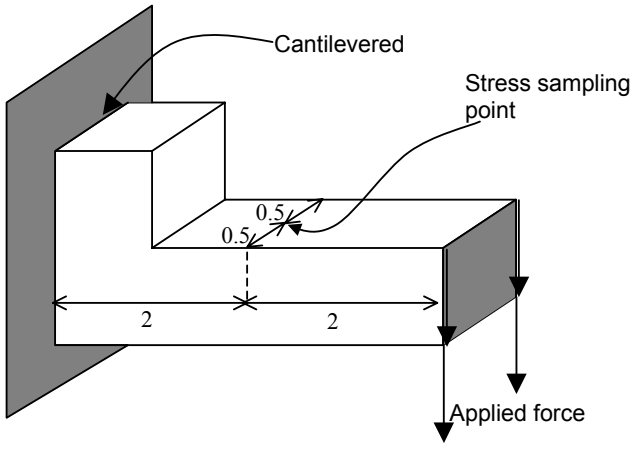


Figure 6-36 Loading condition for the experimental finite element analyses

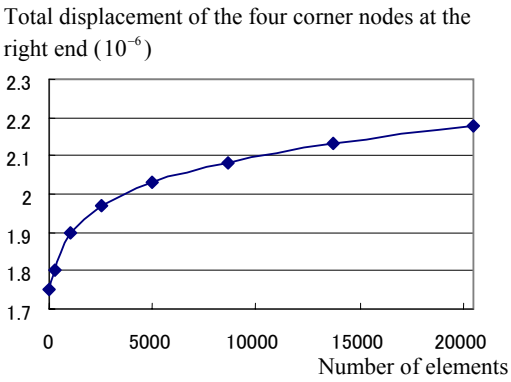


Figure 6-37 Convergence of the total displacement of the four corner nodes at the right end

Table 6-2 Meshes used in the experimental finite element analyses and obtained stress distributions

	Input hex-dominant mesh	All-hex mesh	Stress distribution	Stress distribution near the stress concentration
(1)				
(2)				
(3)				
(4)				

Table 6-3 Qualities and number of elements of the meshes used in the experimental finite element analyses and obtained 1st principal stress at the sampling point

	Worst minimum scaled Jacobian	Maximum aspect ratio	Number of elements	Obtained total displacement of the four corner nodes ($\times 10^{-6}$) of the right end and % error
(1)	1.0	1.0	320	1.80 (20.0%)
(2)	0.13	3.3	552	1.75 (22.2%)
(3)	0.28	3.1	800	1.76 (21.8%)
(4)	0.28	3.1	800	1.79 (20.4%)

6.10 Discussion

The proposed all-hex conversion templates have two limitations: (1) the number of elements in a final all-hex mesh increases drastically, and (2) the quality of some of the elements in the templates is relatively low.

If $n_1 = n_2 = n_f = 4$ is chosen for all elements in a hex-dominant mesh, the total number of elements in the final all-hex mesh increases to about 60 times more than the number of elements in the original hex-dominant mesh. In order to keep the number of elements in the final all-hex mesh low, the original hex-dominant mesh must be made as coarse as possible. Unfortunately, a coarse mesh is in general more difficult to create than a dense mesh. The method presented in Chapter 5 also cannot create many hexahedrons when a large element size is specified. A method for creating such a coarse hex-dominant mesh is one of the future research issues.

Alternatively, different values of n_1, n_2, and n_f can be chosen for each element so that the number of elements in the all-hex mesh remains low. If $n_1 = n_2 = n_f = 2$ could be chosen for all elements in a hex-dominant mesh, the number of elements would be less than eight times more than the number of elements in the original hex-dominant mesh. However, because of the structure of a cap module, n_f must be at least four to accommodate tetrahedrons adjacent to a quadrilateral face. Although n_1 or n_2 can be smaller than four, developing an algorithm for choosing an optimal combination of n_1 and n_2 for each element of a hex-dominant mesh is not trivial and would require future work.

It is also important that an input hex-dominant mesh be of reasonably good quality. As in all other template-based methods, if an element in the hex-dominant mesh is highly ill shaped most of the hexahedrons resulting from this element will also be ill shaped.

Several experiments are conducted to test the effect of the quality of the input hexahedron on the quality of the output hexahedrons. The tests apply three types of the deformation, *taper*, *shear* and *twist* to the input hexahedron. The input hexahedron before the deformation is enclosed by six planes $x = \pm 0.5$, $y = \pm 0.5$ and $z = \pm 0.5$. The top quadrilateral face lies on the plane $y = 0.5$, the right face lies on the plane $x = 0.5$, and the front face lies on the plane $z = 0.5$. The taper deformation shrinks the top face on the plane $y = 0.5$ without deforming the bottom face. The shear deformation moves the top face on the plane $y = 0.5$ to the right without moving the bottom face. The twist deformation rotates the top face about the Y axis without moving the bottom face. The deformed hexahedron is subdivided by the H1 HEXHOOP template with the triangular pattern on the top face.

The quality of the output mesh is shown by the smallest of all minimum scaled Jacobians. The smallest minimum scaled Jacobian is plotted against the minimum scaled Jacobian of the input hexahedron in Figure 6-38. If the twist deformation or the shear deformation is applied to the input hexahedron, the smallest minimum scaled Jacobian of the output mesh monotonicly decreases with the minimum scaled Jacobian of the input hexahedron. When the taper deformation is applied, the smallest minimum scaled Jacobian of the output mesh improves slightly when the deformation is small, but it decreases with a large deformation. The plot implies that the hexahedrons in the input hex-dominant mesh must be reasonably well shaped to have an output all-hex mesh of reasonably high quality.

One possible remedy for this problem is to try limiting the usage of non-hex elements, for which the hexahedrons in the templates are more distorted, to non-critical regions of a FEM analysis. Such a hex-dominant mesh generation technique, in which the user can specifically tell the program where to create hexahedrons and where to create tetrahedrons, is one of the future research topics.

There is a trivial solution to the all-hex mesh generation problem. This trivial solution is obtained by first generating a tetrahedral mesh of the input domain, and then subdividing each tetrahedron into four hexahedrons by the template shown in Figure 6-3 (b). An experimental result shows that an all-hex mesh

generated through this trivial solution has better extreme (worst) quality than an all-hex mesh obtained with the HEXHOOP template. However, the trivial solution cannot produce very high quality elements that are often needed in critical regions of an analysis. Figure 6-39 shows a comparison between all-hex meshes that were created by the trivial solution and by the HEXHOOP templates. Figure 6-39 (a) shows an all-hex mesh created by the trivial solution. The input geometry is a mechanical part, and its geometry is identical to the input geometry of the example shown in Figure 6-34. Figure 6-39 (b) and (c) show histograms of minimum scaled Jacobian of the trivial solution and HEXHOOP generated meshes. These two histograms clearly indicate that an all-hex mesh created with the trivial solution has better extreme quality than an all-hex mesh created with the HEXHOOP templates with respect to minimum scaled Jacobian. However, an all-hex mesh created with the trivial solution does not have well shaped elements that have minimum scaled Jacobian of more than 0.78. Thus, although the trivial solution does not create very flat elements, it cannot create very good elements either. On the other hand, the HEXHOOP templates may create some very flat elements, but it can also create very high quality elements. Hence, if a finite element analysis requires the extreme quality to be better than a certain quality threshold, the trivial solution may work well, assuming that a very high quality tetrahedral mesh can be obtained. However, if a finite element analysis requires very high quality elements in a critical region, the HEXHOOP templates can satisfy such requirement, while the trivial solution cannot.

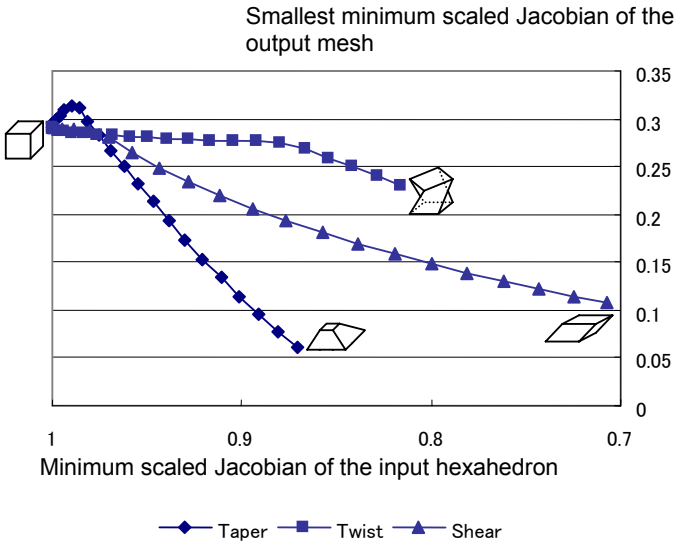
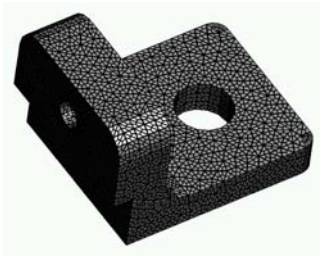
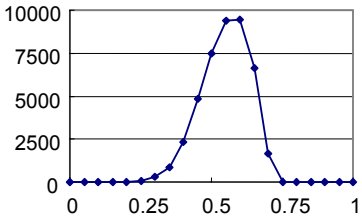


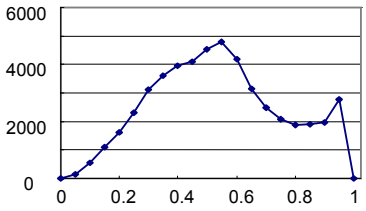
Figure 6-38 Plot of the smallest minimum scaled Jacobian against the minimum scaled Jacobian of the input hexahedron



(a) An all-hex mesh obtained with tetrahedral dicing



(b) Histogram of minimum scaled Jacobian of hexahedrons included in the all-hex mesh created with tetrahedral dicing



(b) Histogram of minimum scaled Jacobian of hexahedrons included in the all-hex mesh created with the HEXHOOP templates

Figure 6-39 Comparison between the histograms of the all-hex mesh created with the HEXHOOP templates and with tetrahedron dicing

6.11 Conclusions

This chapter has presented a set of new mesh conversion templates that automatically convert a hex-dominant mesh to an all-hex mesh. The hex-dominant to all-hex conversion problem had not been solved because it is difficult to subdivide elements in the hex-dominant mesh to smaller hexahedrons without breaking interface conformity when triangular faces of non-hex elements are adjacent to a quadrilateral

face of a hexahedron or a prism. A new modular approach to designing all-hex mesh conversion templates called HEXHOOP is proposed to overcome the problem. A template for a hexahedron or a prism is created by assembling a sub-template called a core and sub-templates called caps. The modular approach allows combining rectangular patterns, which are appropriate to have an adjacent hexahedron, and triangular patterns, which are appropriate to have an adjacent tetrahedron or pyramid, arbitrarily on each quadrilateral face of a template.

There are two major advantages of the HEXHOOP templates, (1) there is no restriction on the configuration of the input hex-dominant mesh like Geode template has [86] since rectangular patterns and triangular patterns are flexibly combined on each quadrilateral faces of a template, and (2) the HEXHOOP templates will create a good quality all-hex mesh if the input hex-dominant mesh is of reasonably good quality. Two solutions to Rob Schneiders' Open Problem [45, 84] are also derived from HEXHOOP templates; they also demonstrate flexibility of the HEXHOOP templates.

Chapter 7 Conclusions

This thesis has presented three new computational methods for 3D mesh generation: (1) the anisotropic tetrahedral mesh generation, (2) the anisotropic hex-dominant mesh generation and (3) the hex-dominant to all-hex mesh conversion. These methods solve unsolved problems including:

- Anisotropic node locating algorithms for a tetrahedral mesh and a hex-dominant mesh
- A 3D node connection algorithm with anisotropy taken into account
- Automatic hex-dominant to all-hex mesh conversion templates

At the same time, they also raise some new research problems, the solutions to which will further improve the output mesh quality and efficiency of the mesh generation process. This chapter presents the conclusions and plans for future research.

7.1 Anisotropic Tetrahedral Mesh Generation

The proposed method takes a geometric domain and desired anisotropy as input and packs ellipsoidal bubbles on the boundary of and inside the input geometric domain to obtain good node locations for an anisotropic tetrahedral mesh. Shapes of the ellipsoidal bubbles are controlled by a user-prescribed anisotropy function. Closely packed ellipsoids mimic an anisotropic Voronoi diagram and give good node locations for an anisotropic tetrahedral mesh. The nodes obtained by the ellipsoidal bubbles are then connected by the advancing front method, and the tetrahedrons are created. Finally, the qualities of tetrahedrons are further improved by the local transformation method [9]. The results show that the proposed method creates high quality anisotropic tetrahedral meshes.

Although there is no theoretical guarantee that the bubbles converge in a certain number of iterations, the experiments conducted in this research show that the bubbles generally converge in 700 iterations for a simple geometry, and 1,500 iterations for a complex geometry as discussed in Section 4.2.2. However, when the gradient of $\mathbf{M}(\mathbf{x})$ is large, the rate of convergence often decreases. In fact for some $\mathbf{M}(\mathbf{x})$ the bubbles may never converge. Solving these problems of slow convergence and non-convergence is a future research issue.

The method, however, creates ill shaped elements if the input geometric domain contains features with sharp corners with a small dihedral angle. A small dihedral angle causes the tetrahedron to flatten, negatively affecting the quality of the tetrahedron. A possible solution to the problem is to apply an appropriate pre-process that removes, or rounds off sharp corners from the input geometric domain. Feature suppression techniques such as the one presented by Ribelles et al. [83] can be applied. However, currently available feature removal techniques typically have problems in robustness and/or computational efficiency. A robust and efficient feature removal technique is still an issue for future research.

7.2 Anisotropic Hex-dominant Mesh Generation with Directionality Control

The proposed method takes a geometric domain and desired anisotropy, directionality and element size as input, and creates a hex-dominant mesh. A hex-dominant mesh combines the good aspects of a tetrahedral mesh and an all-hex mesh; it gives a more accurate solution with fewer elements than a tetrahedral mesh, and it is easier to control the grading of element size in a hex-dominant mesh than in an all-hex mesh.

This method packs rectangular solid cells in the target geometric domain. The cells are moved to stable positions by the physically-based particle simulation, and the cells provide good node locations for a hex-dominant mesh. The nodes are then connected and a tetrahedral mesh is created. Some tetrahedrons are merged to create hexahedrons and prisms, and the tetrahedral mesh is converted to a hex-dominant mesh. This method creates a high quality hex-dominant mesh that complies with the prescribed anisotropy, directionality and element size. A method for creating directionality for a boundary-aligned hex-dominant mesh is also presented.

The proposed method can take arbitrary directionality. However, if the directionality is not compatible with the boundary of the target geometric domain, i.e., if one of three principal directions is not parallel to the normal vector on the boundary of the target geometric domain, many non-hex elements will be created near the boundary. If the directionality is created based on the input geometry, the number of non-hex elements near the boundary can be minimized. However, directionality based on geometry may not be compatible with the requirement of the finite element analysis. It requires future research to develop a

method that automatically creates directionality compatible with both the boundary of the target geometric domain and the requirement of the finite element analysis.

Although this method fills most of the volume of the target domain with hexahedrons, it cannot control the locations of non-hex elements. Since hexahedrons give a more accurate finite element solution, the region that is particularly important for the analysis must be filled with hexahedrons. Non-hex elements thus should be located apart from the critical region. A possible solution for controlling the location of non-hex elements is to apply a post-process that reduces non-hex elements in the critical region by reconnecting/deleting/adding the nodes. Such a topological operation for a hex-dominant mesh is, however, still a topic for future research.

The presence of small features also reduces the number of hexahedrons in the output hex-dominant mesh. If a higher hexahedron percentage is desired, such small features must be removed by a feature suppression technique. However, currently available feature suppression techniques are typically unstable and/or computationally costly. An alternative solution is to specify the non-uniform element size so that smaller elements are created in and near the small features. However, currently available feature identification techniques are also typically not robust. The future development of the feature suppression and feature identification techniques will increase the applicability of the proposed hex-dominant meshing method.

7.3 Hex-dominant to All-hex Mesh Conversion

The proposed method constructs a subdivision template called HEXHOOP for a hexahedron or a prism by combining some sub-templates. The template subdivides a hexahedron or a prism while maintaining topological conformity with neighboring elements. In a hex-dominant to all-hex mesh conversion problem, ten hexahedron-templates and four prism-templates are required to convert a hex-dominant mesh to an all-hex mesh without losing interface conformity; while only two templates are required in a quad-dominant to all-quad mesh conversion. The proposed method successfully creates all fourteen necessary templates, which guarantees the successful conversion from a hex-dominant mesh to an all-hex mesh. It also yields a solution to the Schneider's problem, which subdivides a pyramid into hexahedrons while subdividing a

quadrilateral face into four quadrilaterals and each triangular face into three quadrilaterals. Two mesh smoothing schemes are also presented that improve the quality of the all-hex mesh created by the templates.

The HEXHOOP templates, however, increase the number of elements in the mesh; typically the number of hexahedrons in the output all-hex mesh is almost sixty times as many as the number of elements in the input hex-dominant mesh. To compensate for the increase in the number of elements, the input hex-dominant mesh must be made as coarse as possible. However, a coarse hex-dominant mesh is usually more difficult to create than a dense hex-dominant mesh. A coarse hex-dominant mesh can be created by the method presented in Chapter 5 with long edge lengths specified. However, long edge lengths may reduce the ratio of hexahedrons. A possible way for circumventing this problem is to tie the HEXHOOP template with a feature decomposition technique that roughly subdivides the input geometric domain into large hexahedrons, prisms, pyramids and tetrahedrons. Such a feature decomposition technique requires more research.

One limitation of the HEXHOOP template is that relatively low quality elements are created in and around the regions filled with non-hex elements in the input hex-dominant mesh. The output hexahedrons created by subdividing input non-hex elements tend to have relatively low quality compared to the output hexahedrons created by subdividing an input hexahedrons. If the input non-hex element is already ill shaped, the output hexahedrons will be even worse. Also the hexahedrons created by applying the HEXHOOP template show lower quality than the quality of the input hexahedron. Since the HEXHOOP templates are applied to the hexahedrons adjacent to non-hex elements, those input hexahedrons must be sufficiently well shaped. To avoid the creation of severely ill shaped elements, the input hex-dominant mesh must be of reasonably high quality.

One possible solution to the quality problem is to limit the use of non-hex elements and confine the non-hex elements in the non-critical regions, which are less significant regions for the finite element analysis. If such a hex-dominant mesh can be created, relatively low quality elements will be confined in the non-critical regions in the output all-hex mesh, and the effect of these relatively low quality elements on the finite element analysis will be minimized. However, no currently available technique can create such a

hex-dominant mesh automatically, and it requires more research to develop a method that can automatically create a hex-dominant mesh while limiting the use of non-hex elements to non-critical regions.

Another solution to the quality problem is to apply an appropriate post-process, such as the smoothing schemes presented in Section 6.6. However, there can be better smoothing schemes. An existing mesh smoothing scheme, such as Knupp's [68, 69], or Freitag's [70, 71], may give better results, or an even better mesh smoothing scheme might be found. Topological operations of the all-hex mesh --if possible-- might improve the quality of the mesh. Bern and Eppstein present topological operations for an all-hex mesh [87]; however, the effect of their method on the mesh quality is unknown. Currently, no such a topological operation that improves the quality of the all-hex mesh is known; it is a topic for future research.

List of Publications

Conference Papers

Soji Yamakawa and Kenji Shimada, "High Quality Anisotropic Tetrahedral Mesh Generation via Ellipsoidal Bubble Packing", *Proceedings of the 9th International Meshing Roundtable*, pp. 263-273, 2000.

Soji Yamakawa and Kenji Shimada, "QUAD-LAYER: Layered Quadrilateral Meshing of Narrow Two-Dimensional Domain by Bubble Packing and Chordal Axis Transformation", *Proceedings of the 27th ASME Design Automation Conference*, DAC-21149, 2001.

Soji Yamakawa and Kenji Shimada, "HEXHOOP: Modular Templates for Converting a Hex-dominant Mesh to an All-hex Mesh", *Proceedings of the 10th International Meshing Roundtable*, pp. 235-246, 2001.

Soji Yamakawa and Kenji Shimada, "Hex-dominant Mesh Generation with Directionality Control via Packing Rectangular Solid Cells", *to appear in the Proceedings of Geometric Modeling and Processing 2002.*

Journal Papers

Soji Yamakawa and Kenji Shimada, "QUAD-LAYER: Layered Quadrilateral Meshing of Narrow Two-Dimensional Domain by Bubble Packing and Chordal Axis Transformation", *to appear in the ASME Journal of Mechanical Design.*

Soji Yamakawa and Kenji Shimada, "HEXHOOP: Modular Templates for Converting a Hex-dominant Mesh to an All-hex Mesh", *to appear in Engineering with Computers.*

Soji Yamakawa and Kenji Shimada, "High Quality Anisotropic Tetrahedral Mesh Generation via Ellipsoidal Bubble Packing", *submitted to International Journal for Numerical Methods in Engineering.*

Soji Yamakawa and Kenji Shimada, "Hex-dominant Mesh Generation with Directionality Control via Packing Rectangular Solid Cells", *submitted to International Journal for Numerical Methods in Engineering.*

Pending Patent

Soji Yamakawa and Kenji Shimada, “A System and Method for Converting a Hex-Dominant Mesh to an All-Hexahedral Mesh”

References

[1] T. Itoh, K. Shimada, K. Inoue, A. Yamada, and T. Furuhata, “Automated Conversion of 2D Triangular Mesh into Quadrilateral Mesh with Directionality Control,” Proceedings of 7th International Meshing Roundtable, pp. 77-86, 1998.

[2] K. Shimada, "Physically-Based Mesh Generation : Automated Triangulation of Surfaces and Volumes via Bubble Packing", Ph. D thesis, *Massachusetts Institute of Technology*, 1993

[3] K. Shimada and D. C. Gossard, “Bubble Mesh : Automated Triangular Meshing of Non-manifold Geometry by Sphere Packing,” *3rd Symposium on Solid Modeling and Applications*, pp. 409-419, 1995.

[4] K. Shimada, A. Yamada, and T. Itoh, “Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles,” Proceedings of 6th International Meshing Roundtable, pp. 375-390, 1997.

[5] K. Shimada and D. C. Gossard, “Automatic Triangular Mesh Generation of Trimmed Parametric Surfaces for Finite Element Analysis,” *Computer Aided Geometric Design*, vol. 15, pp. 199-222, 1998.

[6] K. Shimada, J.-H. Liao, and T. Itoh, “Quadrilateral Meshing with Directionality Control through the Packing of Square Cells,” Proceedings of 7th International Meshing Roundtable, pp. 61-75, 1998.

[7] R. Löhner and J. R. Cebral, “Parallel Advancing Front Grid Generation,” Proceedings of 8th International Meshing Roundtable, pp. 67-74, 1999.

[8] P. Fleischmann and S. Selberherr, “Three-Dimensional Delaunay Mesh Generation Using a Modified Advancing Front Approach,” Proceedings of 6th International Meshing Roundtable, pp. 267-278, 1997.

[9] B. Joe, “Construction of Three-dimensional Improved-quality Triangulations Using Local Transformations,” *SIAM Journal on Scientific Computing*, vol. 16, pp. 1292-1307, 1995.

[10] S. J. Owen, S. A. Canann, and S. Sagal, "Pyramid Elements for Maintaining Tetrahedra to Hexahedra Conformability," *AMD-Vol. 220 Trend in Unstructured Mesh Generation, ASME*, pp. 123-129, 1997.

[11] S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*, fourth edition ed: McGraw-Hill, 2002.

[12] E. B. Becker, G. F. Carey, and J. T. Oden, *Finite Elements: an Introduction*, vol. 1: Prentice Hall, 1981.

[13] J. Peraire, J. Peiró, and K. Morgan, "Adaptive Remeshing for Three-Dimensional Compressible Flow Computation," *Journal of Computational Phzsics*, vol. 103, pp. 269-285, 1992.

[14] J. Peraire, J. Peiro, L. Formaggia, K. Morgan, and O. C. Zienkiewicz, "Finite Element Euler Computations in Three Dimensions," *International Journal for Numerical Methods in Engineering*, vol. 26, pp. 2135-2159, 1988.

[15] J. Peraire, M. Vahdati, K. Morgan, and O. C. Zienkiewicz, "Adaptive Remeshing for Compressible Flow Computations," *Journal of Computational Physics*, vol. 72, pp. 449-466, 1987.

[16] M. J. Castro-Diaz, F. Hecht, and B. Mohammadi, "New Progress in Anisotropic Grid Adaptation for Inviscid and Viscous Flows Simulations," Proceedings of 4th International Meshing Roundtable, pp. 73-85, 1995.

[17] W. G. Habashi, J. Dompierre, Y. Bourgault, M. Fortin, and M. G. Vallet, "Certifiable Computational Fluid Dynamics Through Mesh Optimization," *AIAA Journal on Computational Fluid Dynamics Simulations*, vol. 36, pp. 703-711, 1998.

[18] O. C. Zienkiewicz and J. Z. Zhu, "Adaptivity and Mesh Generation," *International Journal for Numerical Methods in Engineering*, vol. 32, pp. 783-810, 1991.

[19] N. Viswanath, K. Shimada, and T. Itoh, "Quadrilateral Meshing with Anisotropy and Directionality Control via Close Packing of Rectangular cells," Proceedings of 9th International Meshing Roundtable, pp. 227-238, 2000.

[20] S. J. Owen, "A Survey of Unstructured Mesh Generation Technology," Proceedings of 7th International Meshing Roundtable, pp. 239-263, 1998.

[21] D. F. Watson, "Computing the N-dimensional Delaunay Tessellation with Application to Voronoi Polytopes," *The Computer Journal*, vol. 24, pp. 167-172, 1981.

[22] A. Bower, "Computing Dirichlet tessellations," *The Computer Journal*, vol. 24, pp. 162-166, 1981.

[23] P. L. George, F. Hecht, and E. Saltel, "Automatic Mesh Generator with Specified Boundary," *Computer Methods in Applied Mechanics and Engineering*, vol. 92, pp. 269-288, 1991.

[24] N. P. Weatherill and O. Hassan, "Efficient Three-dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints," *International Journal for Numerical Methods in Engineering*, vol. 37, pp. 2005-2039, 1994.

[25] K. Sugihara and H. Inagaki, "Why is the 3D Delaunay Triangulation Difficult to Construct?," *Information Processing Letters*, vol. 54, pp. 275-280, 1995.

[26] P. L. George, "Tet Meshing: Construction, Optimization and Adaptation," Proceedings of 8th International Meshing Roundtable, pp. 133-141, 1999.

[27] J. R. Shewchuk, "Tetrahedral Mesh Generation By Delaunay Refinement," Proceedings of 14th Annual Conference on Computational Geometry, pp. 86-95, 1998.

[28] B. Joe, "Construction of Three-dimensional Delaunay Triangulation Using Local Transformations," *Computer Aided Geometric Design 8*, pp. 123-142, 1991.

[29] S. H. Lo, "A New Mesh Generation Scheme for Arbitrary Planar Domains," *International Journal for Numerical Methods in Engineering*, vol. 21, pp. 1403-1426, 1985.

[30] P. Möller and P. Hansbo, "On Advancing Front Mesh Generation in Three Dimensions," *International Journal for Numerical Methods in Engineering*, vol. 38, pp. 3551-3369, 1995.

[31] R. Löhner, "A Parallel Advancing Front Grid Generation Scheme," *International Journal for Numerical Methods in Engineering*, vol. 51, pp. 663-678, 2001.

[32] T. D. Blacker and M. B. Stephenson, "Paving : A New Approach to Automated Quadrilateral Mesh Generation," *International Journal for Numerical Methods in Engineering*, vol. 32, pp. 811-847, 1991.

[33] T. D. Blacker and R. J. Meyers, “Seams and Wedges in Plastering: A 3-D Hexahedral Mesh Generation Algorithm,” *Engineering with Computers*, vol. 2, pp. 83-93, 1993.

[34] T. J. Tautges, T. Blacker, and S. A. Mitchell, “The Whisker Weaving Algorithm: A Connectivity-Based Method for Constructing All-Hexahedral Finite Element Meshes,” *International Journal for Numerical Methods in Engineering*, vol. 39, pp. 3327-3349, 1996.

[35] S. A. Mitchell, “A Characterization of the Quadrilateral Meshes of a Surface Which Admit a Compatible Hexahedral Mesh of the Enclosed Volume,” Proceedings of 13th Annual Symposium on Theoretical Aspects of Compute Science (STACS), pp. 465-476, 1996.

[36] R. J. Meyers, T. J. Tautges, and P. M. Tuchinsky, “The "Hex-Tet" Hex-Dominant Meshing Algorithm as Implemented in CUBIT,” Proceedings of 7th International Meshing Roundtable, pp. 151-158, 1998.

[37] P. M. Tuchinsky and B. W. Clark, “The "HexTet" Hex-Dominant Automesher: An Interim Progress Report,” Proceedings of 6th International Meshing Roundtable, pp. 183-193, 1997.

[38] S. J. Owen and S. Sagal, “H-Morph: an Indirect Approach to Advancing Front Hex Meshing,” *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 289-312, 2000.

[39] S. J. Owen, M. L. Staten, S. A. Canann, and S. Sagal, “Q-Morph: An Indirect Approach to Advancing Front Quad Meshing,” *International Journal for Numerical Methods in Engineering*, vol. 44, pp. 1317-1340, 1999.

[40] D. S. Thompson and B. K. Soni, “Generation of Quad- and Hex-dominant, Semistructured Meshes Using an Advancing Layer Scheme,” Proceedings of 8th International Meshing Roundtable, pp. 171-178, 1999.

[41] X.-Y. Li, S.-H. Teng, and A. Üngör, “Biting: Advancing Front Meets Sphere Packing,” *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 61-81, 2000.

[42] X.-Y. Li, S.-H. Teng, and A. Ungör, “Biting Ellipses to Generate Anisotropic Mesh,” Proceedings of 8th International Meshing Roundtable, pp. 97-108, 1999.

[43] X.-Y. Li, S.-H. Teng, and A. Ungör, “Biting Spheres in 3D,” Proceedings of 8th International Meshing Roundtable, pp. 85-95, 1999.

[44] M. A. Yerry and M. S. Shephard, "Automatic Three-dimensional Mesh Generation by the Modified-octree Technique," *International Journal for Numerical Methods in Engineering*, vol. 20, pp. 1965-1990, 1984.

[45] R. Schneiders, "A Grid-based Algorithm for the Generation of Hexahedral Element Meshes," *Engineering with Computers*, vol. 12, pp. 168-177, 1996.

[46] L. Maréchal, "A New Approach to Octree-Based Hexahedral Meshing," Proceedings of 10th International Meshing Roundtable, pp. 209-221, 2001.

[47] J. M. Reddy and G. M. Turkiyyah, "Computation of 3D Skeletons Using a Generalized Delaunay Triangulation Technique," *Computer-Aided Design*, vol. 27, pp. 677-694, 1995.

[48] L. Prasad, "Morphological Analysis of Shapes," in *http://cnls.lanl.gov/Highlights/1997-07/*: Los Alamos National Laboratory, 1997.

[49] G. M. Turkiyyah, D. W. Storti, M. Ganter, H. Chen, and M. Vimawala, "An Accelerated Triangulation Method for Computing the Skeletons of Free-form Solid Models," *Computer-Aided Design*, vol. 29, pp. 5-19, 1997.

[50] A. Sudhalkar, L. Gürsöz, and F. Prinz, "Box-skeletons of Discrete Solids," *Computer-Aided Design*, vol. 28, pp. 507-517, 1996.

[51] T. K. H. Tam and C. G. Armstrong, "2D Finite Element Mesh Generation by Medial Axis Subdivision," *Advances in Engineering Software*, vol. 13, pp. 313-324, 1991.

[52] C. G. Armstrong, T. K. H. Tam, D. J. Robinson, R. M. McKeag, and M. A. Price, "Automatic Generation of Well Structured Meshes using Medial Axis and Surface Subdivision," *DE-Vol. 32-2, Advances in Design Automation, ASME*, vol. 2, pp. 139-146, 1991.

[53] W. R. Quadros, K. Ramsawami, F. B. Prinz, and B.Gurumoorthy, "LayTracks: A New Approach To Automated Quadrilateral Mesh Generation using MAT," Proceedings of 9th International Meshing Roundtable, pp. 239-250, 2000.

[54] A. Sheffer and M. Bercovier, "Hexahedral Meshing of Non-linear Volumes Using Voronoi Faces and Edges," *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 329-351, 2000.

[55] M. Etzion and A. Rappoport, “Computing the Voronoi Diagram of a 3-D Polyhedron by Separate Computation of its Symbolic and Geometric Parts,” Proceedings of ACM fifth symposium on Solid Modeling and Applications, 1999.

[56] F. J. Bossen and P. S. Heckbert, “A Pliant Method for Anisotropic Mesh Generation,” Proceedings of 5th International Meshing Roundtable, pp. 63-74, 1996.

[57] H. Borouchaki, P. J. Frey, and P. L. George, “Unstructured Triangular-Quadrilateral Mesh Generation. Application to Surface Meshing.,” Proceedings of 5th International Meshing Roundtable, pp. 229-242, 1996.

[58] R. V. Garimella and M. S. Shephard, “Boundary Layer Meshing for Viscous Flows in Complex Domain,” Proceedings of 7th International Meshing Roundtable, pp. 107-118, 1998.

[59] R. V. Garimella and M. S. Shephard, “Boundary Layer Mesh Generation for Viscous Flow Simulations,” *International Journal for Numereical Methods in Engineering*, vol. 49, pp. 193-218, 2000.

[60] M. S. Shephard, “Meshing Environment for Geometry-based Analysis,” *International Journal for Numerical Methods in Engineering*, vol. 47, pp. 169-190, 2000.

[61] K. E. Jansen and M. S. Shephard, “On Anisotropic Mesh Generation and Quality Control in Complex Flow Problems,” Proceedings of 10th International Meshing Roundtable, pp. 341-349, 2001.

[62] J. Shepherd, S. A. Mitchell, P. Knupp, and D. White, “Methods for Multisweep Automation,” Proceedings of 9th International Meshing Roundtable, pp. 77-87, 2000.

[63] M. Lai, S. Benzley, and D. White, “Automated Hexahedral Mesh Generation by Generalized Multiple Source to Multiple Target Sweeping,” *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 261-275, 2000.

[64] D. R. White and T. J. Tautges, “Automatic Scheme Selection for Toolkit Hex Meshing,” *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 127-144, 2000.

[65] H. Li and G. Cheng, “New Method for Graded Mesh Generation of All Hexahedral Finite Elements,” *Computers and Structures*, vol. 76, pp. 729-740, 2000.

[66] D.-Y. Kwak and Y.-T. Im, "Remeshing for Metal Forming Simulations-Part II: Three-Dimensional Hexahedral Mesh Generation," *International Journal for Numerical Methods in Engineering*, vol. 53, pp. 2501-2528, 2002.

[67] S. Meshkat and D. Talmor, "Generating a Mixed Mesh of Hexahedra, Pentahedra and Tetrahedra from an Underlying Tetrahedral Mesh," *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 17-30, 2000.

[68] P. M. Knupp, "Hexahedral Mesh Untangling and Algebraic Mesh Quality Metrics," Proceedings of 9th International Meshing Roundtable, pp. 173-183, 2000.

[69] 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*, vol. 48, pp. 1165-1185, 2000.

[70] L. A. Freitag and P. Plassmann, "Local Optimization-based Simplical Mesh Untangling and Improvement," *International Journal for Numerical Methods in Engineering*, vol. 49, pp. 109-125, 2000.

[71] L. A. Freitag and P. E. Plassmann, "Local Optimization-based Untangling Algorithms for Quadrilateral Meshes," Proceedings of 10th International Meshing Roundtable, pp. 397-406, 2001.

[72] T. Zhou and K. Shimada, "An Angle-Based Approach to Two-dimensional Mesh Smoothing," Proceedings of 9th International Meshing Roundtable, pp. 373-384, 2000.

[73] N. A. Calvo and S. R. Idelsohn, "All-Hexahedral Mesh Smoothing with a Node-Based Measure of Quality," *International Journal for Numerical Methods in Engineering*, vol. 50, pp. 1957-1967, 2001.

[74] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Second Edition ed: Cambridge University Press, 1992.

[75] P. P. Pébay and T. J. Baker, "A Comparison of Triangle Quality Measures," Proceedings of 10th International Meshing Roundtable, pp. 327-340, 2001.

[76] R. Sedgewick, "Range Searching," in *Algorithms in C++*: Addison-Wesley, 1992, pp. 373-388.

[77] P. Heckbert, “Fast Surface Particle Repulsion,” Carnegie Mellon University, CMU Computer Science Tech. Report CMU-CS-97-130 1997.

[78] S. J. Owen, "Non-Simplicical Unstructured Mesh Generation, Ph.D Thesis", Ph.D Thesis, *The Department of Civil and Environmental Engineering*, 1999

[79] D. L. Dewhirst and P. M. Grinsell, “Joining Tetrahedra to Hexahedra,” Proceedings of MSC World Users' Conference (http://www.mscsoftware.com/support/library/conf/), 1993.

[80] D. L. Dewhirst, S. Vangavolu, and H. Wattrick, “The Combination of Hexahedral and Tetrahedral Meshing Algorithms,” Proceedings of 4th International Meshing Roundtable, pp. 291-304, 1995.

[81] C. R. Dohrmann, S. W. Key, and M. W. Heinstein, “Methods for Connecting Dissimilar Three-Dimensional Finite Element Meshes,” *International Journal for Numerical Methods in Engineering*, vol. 47, pp. 1057-1080, 2000.

[82] *ANSYS*: ANSYS Inc. (http://www.ansys.com).

[83] J. Ribelles, P. Heckbert, M. Garland, T. Stahovich, and V. Srivastava, “Finding and Removing Features from Polyhedra,” Proceedings of ASME Design Automation Conference, Pittsburgh PA, Sept. 2001.

[84] R. Schneiders, “Open Problem, http://www-users.informatik.rwth-aachen.de/~roberts/open.html” .

[85] C. D. Carboner, “Constrained Mesh Generation, http://www-users.informatik.rwth-aachen.de/~roberts/SchPyr/index.html” .

[86] S. A. Mitchell, “The All-Hex Geode-Template for Conforming a Diced Tetrahedral Mesh to any Diced Hexahedral Mesh,” Proceedings of 7th International Meshing Roundtable, pp. 295-305, 1998.

[87] M. Bern and D. Eppstein, “Flipping Cubical Meshes,” Proceedings of 10th International Meshing Roundtable, pp. 19-29, 2001.