Surfacing Splicing: A Method of Quadrilateral Mesh Generation and Modification for Surfaces by Dual Creation and Manipulation

Benjamin Todd Grover
Brigham Young University - Provo
SURFACE SPICING: A METHOD OF QUADRILATERAL MESH
GENERATION AND MODIFICATION FOR SURFACES
BY DUAL CREATION AND MANIPULATION

BENJAMIN TODD GROVER
SURFACE SPICING: A METHOD OF QUADRILATERAL MESH GENERATION AND MODIFICATION FOR SURFACES BY DUAL CREATION AND MANIPULATION

by
Benjamin Todd Grover

A thesis submitted to the faculty of Brigham Young University in partial fulfillment of the requirements for the degree of Master of Science

Department of Civil and Environmental Engineering Brigham Young University
April 2002
BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Benjamin Todd Grover

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

Date

Jan 16, 2002

Steven E. Benzley, Chair

Date

Jan 16, 2002

Richard J. Balling

Date

Jan 17, 2002

Ray J. Meyers
As chair of the candidate’s graduate committee, I have read the thesis of Benjamin Todd Grover in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Jan 16, 2002

Date

Steven E. Benzley
Chair, Graduate Committee

Wayne Downs
Graduate Coordinator,
Department of Civil and
Environmental Engineering

Accepted for the Department

Accepted for the College

Douglas M. Chabries
Dean, College of Engineering and Technology
ABSTRACT

SURFACE SPLICING: A METHOD OF QUADRILATERAL MESH GENERATION AND MODIFICATION FOR SURFACES BY DUAL CREATION AND MANIPULATION

Benjamin Todd Grover
Department of Civil and Environmental Engineering
Master of Science

The effective generation of high quality quadrilateral surface meshes is an area of important research and development for the finite element community. Quadrilateral elements generally lead to more efficient and accurate finite element results. In addition, some all hexahedral volume meshing algorithms are based on an initial quadrilateral mesh surface mesh that has specific connectivity requirements.

This thesis presents a new and unique procedure named “Surface Splicing”. Surface Splicing allows for the generation of all quadrilateral surface meshes as well as the ability to edit these meshes via the dual. The dual contains the same data as the mesh but, unlike the mesh, the dual directly allows the visualization of how surface and volume elements interrelate and connect with one another. The dual also provides mesh
connectivity information that is crucial in forming an all-quadrilateral surface mesh that can form the basis of an all hexahedral volume mesh.
Although I wrote this thesis, it contains the contribution of many different people that helped me along the way.

I'd first and foremost like to thank Dr. Steven Benzley for all of his help and support. His knowledge and vision of meshing always kept this research interesting and exciting. He also helped a great deal in the writing and editing of this thesis.

Secondly, I am grateful for the mentorship of Jason Shepherd at Sandia National Laboratories. I have never worked with someone that was so able to help me think and learn. I also appreciate his help in the editing of this thesis.

The CUBIT research team at BYU deserves thanks also. Mike Borden, Mark Nugent and Kirk Walton all have helped me numerous times when I was stuck debugging an algorithm or trying to come up with an idea to solve a problem. I am also grateful for their friendship and the good times that we had working together.

Lastly, I am grateful for my wife, Carrie and my parents, Mike and Connie. Carrie and my parents have always been so supportive of my educational pursuits. My parents helped to instill in me a great work ethic and the importance of education. Carrie has been so supportive as I have spent many late nights on campus working towards my Master's degree. She also has helped in the editing of this thesis. I am so thankful for her.
# TABLE OF CONTENTS

LIST OF FIGURES ........................................................................................................... ix  
LIST OF TABLES ............................................................................................................. xiii  

1 INTRODUCTION ........................................................................................................... 1  

2 MESHING ALGORITHMS .......................................................................................... 3  

2.1 Current Quad-Hex Meshing Algorithms ............................................................... 3  

3 THE DUAL OF A MESH ............................................................................................... 13  

3.1 The Dual 2-D Triangle ........................................................................................... 13  

3.2 The Dual for 2-D Quadrilaterals .......................................................................... 15  

3.3 The Dual of 3-D Hexahedra ................................................................................. 20  

3.4 Attributes of the Dual .......................................................................................... 22  

4 MESHING CAPABILITIES OF THE DUAL ............................................................... 25  

4.1 Non—Editability of Existing Surface Meshes ....................................................... 25  

4.2 Self-Intersecting Chords on a Surface ................................................................. 26  

4.3 Global Self-Intersections ..................................................................................... 27  

4.4 Self-Tangent Element Chords on a Surface ....................................................... 28  

4.5 Global Self-Tangent Chords ................................................................................ 29  

5 THE SURFACE SPLICING ALGORITHM .................................................................. 31  

5.1 Definitions ............................................................................................................ 32  

5.2 Steps of the Algorithm ....................................................................................... 33
LIST OF FIGURES

Figure 2.1 - Four logical corners used by the mapping algorithm ......................... 4
Figure 2.2 - Surface mesh created by the mapping algorithm ................................. 4
Figure 2.3 - Mapped volume mesh ................................................................. 5
Figure 2.4- Paving of a five-sided surface ...................................................... 6
Figure 2.5 - Continued paving of the interior of a five-sided surface .......................... 6
Figure 2.6- Fully paved surface mesh ............................................................. 7
Figure 2.7- Node with 6 valent element edges ................................................ 8
Figure 2.8- Volume mesh created by the sweeping algorithm ............................ 9
Figure 2.9- Surface mesh and corresponding whisker sheet diagrams .................... 10
Figure 2.10- Dangling whiskers on the whisker sheets ........................................ 10
Figure 2.11- Completed whisker woven sheets .................................................. 11
Figure 2.12- Hook shaped object meshed with the whisker weaving algorithm .......... 12
Figure 3.1- The Voronoï diagram used to find the triangulation of a set of points on a
surface ........................................................................................................... 14
Figure 3.2- The Delauny Triangulation of the set of points in Figure 3.1 ................. 14
Figure 3.3- 2-cell created by the intersection of multiple dual chords. The mesh is shown
in dotted lines and the dual is shown in solid lines ........................................ 15
Figure 3.4- Centroid in dual space created by the intersection of two chords. The mesh is
shown in dotted lines and the dual is shown in solid lines ................................ 16
Figure 3.5- Internal dual chord ................................................................. 17
Figure 3.6- Creation steps of the dual of a surface mesh ......................... 18
Figure 3.7- Two tangent dual chords ...................................................... 18
Figure 3.8- Mesh and dual entities ....................................................... 19
Figure 3.9- Mesh Entities of a surface mesh ........................................... 20
Figure 3.10- Dual entities of a surface mesh .......................................... 20
Figure 3.11- A dual chord representing a row of hexahedra .................... 21
Figure 3.12- A dual sheet ...................................................................... 22
Figure 4.1- Meshed geometry with an undesirable mesh interval ............ 26
Figure 4.2- Self Intersecting dual chord on one surface ......................... 27
Figure 4.3- Globally self-intersecting dual chord .................................... 28
Figure 4.4- Surface self-tangency. The three edges of tangency are shown in bold. 28
Figure 4.5- Global self-tangency ......................................................... 29
Figure 5.1- Components of the dual. The numbers represent the ids of the chords and
  junctions on the surface. .................................................................... 32
Figure 5.2- Counter-clockwise ordering of splice chord segments ............ 34
Figure 5.3- Surface with one splice chord and two junctions created ....... 35
Figure 5.4- Multiple chords on a surface with junctions created at the intersection. Chord
  segments are also shown.................................................................... 35
Figure 5.5- Interior splice chord (3) that starts and ends on the same junction 36
Figure 5.6- A fully completed dual mesh on the surface of a brick .......... 37
Figure 5.7- Reference 2-cell ............................................................... 37
Figure 5.8- A dual junction with some of its associated segment pairs ....... 39
Figure 5.9- Boundary 2-Cell showing nodes and edges ................................................. 40
Figure 5.10- Boundary segment and its associated nodes ............................................. 41
Figure 5.11- Quadrilateral element showing nodes and edges stored in ccw order. Nodes
are identified with numbers and edges with letters ................................................. 42
Figure 5.12- Dual and mesh of a rectangular surface. Dual entities have id numbers
shown on them ........................................................................................................ 43
Figure 5.13- Global self-intersection .......................................................................... 44
Figure 5.14- Local self-intersection ............................................................................ 44
Figure 5.15- Boundary junctions on the edge of a meshed brick .................................. 46
Figure 5.16- Meshed brick with boundary junction and newly created chord .............. 47
Figure 5.17- Two chords that are on non-adjacent surfaces ..................................... 48
Figure 5.18- Chord created on front surfaces that connect the two non-adjacent chords,
allowing the creation of a new super chord ............................................................. 48
Figure 5.19- Quadrilateral element showing opposite edges as well as the edges being
added to tangency_list .............................................................................................. 50
Figure 5.20- Self-tangency on a surface mesh. Bolded edges show the area of tangency.
..................................................................................................................................... 52
Figure 6.1- Meshed circle (highlight shows boundary edges) ..................................... 54
Figure 6.2- Surface of a circle with chords of opposite edges highlighted .................... 55
Figure 6.3- Created dual on an element ...................................................................... 56
Figure 6.4- Partial traversal of interior chord .............................................................. 57
Figure 6.5- Fully traversed interior chord .................................................................. 58
Figure 6.6- All dual chords created on the mesh ......................................................... 58
Figure 7.1- Hook shaped object meshed with the whisker-weaving algorithm............. 62
Figure 7.2- Dual sheet in the hook object shown in Figure 7.1........................................ 63
Figure 7.3- Hook figure with dual sheets removed......................................................... 64
Figure 7.4- Undesirable mesh interval on the surface.................................................... 64
Figure 7.5- Deleted dual and mesh ................................................................................. 65
Figure 7.6- Newly created dual....................................................................................... 65
Figure 7.7- Newly created surface mesh from edited dual............................................. 66
Figure 7.8- 5 by 5 mapped mesh................................................................................... 66
Figure 7.9- Dual representation of the mesh shown in Figure 7.8................................. 67
Figure 7.10- New chord created in the existing dual.................................................... 67
Figure 7.11- Mesh created from the edited dual in Figure 7.10........................................ 68
Figure 7.12- Mapped rectangular surface mesh............................................................ 68
Figure 7.13- Dual representation of the mesh shown in Figure 7.14, with additional chords inserted to spell “10th I.M.R.”......................................................... 69
Figure 7.14- Insertion of dual chords to spell 10th I.M.R............................................. 69
Figure 8.1- Local self-intersection.................................................................................. 73
Figure 8.2- Self-intersection changed to two loops....................................................... 73
Figure 8.3- Self-intersection changed to one loop......................................................... 74
LIST OF TABLES

Table 3.1- Relationship of mesh and dual entities in two dimensions. ......................... 19
Table 3.2- Relationships of a dual mesh and mesh entities in three dimensions. .......... 21
Table 7.1-Mesh quality metrics for Figures 7.1 and 7.3.............................................. 63
1 INTRODUCTION

The effective generation of high quality quadrilateral surface meshes is an area of important research and development for the finite element community. Quadrilateral elements generally lead to more efficient and accurate finite element results. In addition, some all hexahedral volume-meshing algorithms are based on an initial quadrilateral surface mesh that has specific connectivity requirements.

This thesis presents a new and unique mesh generation procedure that provides for the creation and editing of quadrilateral surface finite elements based on the dual of the mesh. The dual, which will be explained in detail in Chapter 4, contains the same data as the mesh. The dual, however, helps to visualize how all elements on a surface or in a volume interrelate and connect to one another. Being able to visualize this interrelationship is a powerful tool.

Frequently in finite element mesh generation, automatically created meshes need to be edited in some areas so the mesh will be of higher quality and in turn give more accurate analysis results. By utilizing the dual to edit the mesh, one can easily see what changes will propagate thorough a mesh if one element is deleted or moved. Without utilizing the dual, monitoring these changes is difficult. The dual provides direct information about the global connectivity of the mesh. This global connectivity is crucial in generating both high quality surface meshes as well as admissible quadrilateral surface
Chapter 4 explains this information in greater detail.

Chapter 5 introduces the Surface Splicing algorithm. This algorithm is used to create a mesh from the dual. As explained previously, the dual of a mesh is a powerful way to modify and monitor meshes, but the dual is not the mesh. The surface splicing algorithm is used to convert the data stored in the dual to a high quality surface mesh.

Chapter 6 explains in greater detail how existing surface meshes are first converted to a dual representation of the mesh, edited, and then changed back to a surface mesh.

Chapter 7 gives some examples of mesh editing. Finally, Chapter 8 suggests future research topics that may be pursued.
2 MESHING ALGORITHMS

There are many ways to generate meshes of surfaces and volumes. This chapter gives some background on current quadrilateral (quad) and hexahedral (hex) meshing algorithms. The algorithms explained in this chapter use familiar concepts and require little background information.

2.1 Current Quad-Hex Meshing Algorithms

The following sections outline some of the major quad-hex meshing algorithms that are in use today. Some of the algorithms, like mapping, create a very structured mesh and some, like paving, create an unstructured mesh. A structured mesh is one in which all interior nodes are connected to exactly four element edges. An unstructured mesh can have less than or more than four element edges connected to an interior node.

2.1.1 Surface Mapping

This method [4] provides a quick meshing algorithm to create structured surface meshes. The process of mapping is quite simple. Four logical corners are selected on the surface and the interval counts on the opposite sets of curves are constrained to be equal. A grid is then overlaid on the surface that satisfies the interval counts specified on the four sets of curves. Figure 2.1 shows four logical corners on a surface. The opposite sets of curves must have the same interval count so that the grid can be properly created,
guaranteeing an all-quadrilateral mesh. Mapping is very efficient for surfaces on which four logical corners are easily found. Mapping always generates a structured surface mesh, i.e. a mesh that always has well formed quadrilateral elements with interior nodes connected to exactly four element edges. Figure 2.2 shows a mapped mesh.

![Logical corners](image)

**Figure 2.1-** Four logical corners used by the mapping algorithm.

![Surface mesh](image)

**Figure 2.2-** Surface mesh created by the mapping algorithm

### 2.1.2 Volume Mapping

Volume mapping is the three-dimensional extension to surface mapping. Instead of defining four logical corners on a surface, 8 logical corners are defined on a volume,
and intervals on opposite sides are constrained to be equal. A grid is then created in the volume, which forms an all hexahedral mesh. Figure 2.3 shows a mapped volume mesh.

![Figure 2.3- Mapped volume mesh.](image)

### 2.1.3 Paving

The paving algorithm [2], unlike mapping, is used to create an unstructured mesh. The paving algorithm does not require a surface to have four logical corners or conformity of intervals on opposite sets of curves. Instead, the only constraint, which must be satisfied for paving is that, the sum of the intervals around the curves of the surface must be even. The paving algorithm then uses an advancing front method to create quads, first on the boundaries of the surface and then advancing inward.

This algorithm starts the meshing process by using the intervals assigned to the boundary curves on the surface, then projecting from the boundary into the surface to create quads on the boundary curves of the surface. It then advances inward in a counter-
clockwise direction creating more quads on the “front”. Figure 2.4 shows how the paving algorithm advances on a surface, and Figure 2.5 shows the paving algorithm after it has advanced around the boundary once and continues on the interior of the surface.

![Figure 2.4- Paving of a five-sided surface.](image1)

![Figure 2.5- Continued paving of the interior of a five-sided surface.](image2)

If an interior fixed boundary exists, then the paving algorithm also starts to create quads on the interior boundary in a clockwise direction. The paving algorithm makes
each row of quads by projecting the new quad edges outward into the interior of the surface and checks to make sure that rows of quads do not intersect and that the quads are generally the same size as the other quads already created. Once one complete row of quads is created, the innermost row of newly created nodes becomes the new boundary and the paving algorithm continues to advance until the surface is meshed. Figure 2.6 shows a completely paved surface mesh.

Paving is very good at conforming to irregular shaped surfaces. The paving algorithm also tries to respect the initial interval count of the boundary but sometimes cannot depending on the shape of the surface or the high valency of nodes that are created. An example of high node valency is given in Figure 2.7.

The paving algorithm operates locally. Operating locally means it creates quads based on near field conditions and does not use information about adjacent surfaces or even other regions of the surface being meshed. Operating locally can be a shortcoming of paving. Global control is often desirable so that the surface mesh can be of a proper connectivity to create a valid all hexahedral volume mesh.

![Fully paved surface mesh.](image)
2.1.4 Sweep

The sweeping algorithm [8] is a volume-meshing scheme that propagates a mesh from a source surface to a target surface. A source and a target surface are connected by a mapped linking surface. Source and target surfaces can either be structured or unstructured. The interval count on the linking surface is specified and then the mesh on the source surface is "swept" in the direction of the linking surface to the target surface. A swept mesh is shown in Figure 2.8. Sweeping propagates a surface mesh through a volume to create a well-structured volume mesh. A volume is sweepable only if the source and target surfaces are topologically equivalent [15].

The sweeping algorithm cannot be used for geometries that do not fit the constraints explained above. Other algorithms are currently under development so that any type of volume can be meshed with good quality hexahedral elements [10,12,14]. The whisker weaving algorithm [7] allows the meshing of more complex volumes by

Figure 2.7- Node with 6 valent element edges.
using the information stored in the surface mesh to generate a volume mesh. The whisker weaving algorithm is explained in the next section.

![Source Surface](image1)
![Target Surface](image2)

**Figure 2.8- Volume mesh created by the sweeping algorithm.**

### 2.1.5 The Whisker Weaving Algorithm

The whisker weaving algorithm [17], relies on the information contained in the surface mesh to create a mesh for a volume. The whisker weaving algorithm works by first creating “chords” across the elements on a surface mesh. The “chords” are created by connecting opposite edges of elements on a mesh until the chord forms a closed loop. Surface chords that are created by the whisker weaving algorithm are shown in Figure 2.9. These closed loops form “whisker sheets” which are shown in Figure 2.10. The dashed lines in Figure 2.9 are the boundaries of the four whisker sheets. The whisker weaving algorithm begins by creating dangling “whiskers” on the sheets. The dangling whiskers are shown in Figure 2.10. These whiskers represent the exterior quadrilateral
faces and are shown in Figure 2.9. The algorithm finds three pair wise adjacent chords on the whisker diagrams and crosses them. The algorithm continues until no dangling whiskers are left. The complete whisker woven sheets are shown in Figure 2.22.

Figure 2.9- Surface mesh and corresponding whisker sheet diagrams.

Figure 2.10- Dangling whiskers on the whisker sheets.
The whisker weaving algorithm is automatic and creates reliable all hex meshes. The whisker weaving algorithm also works well for arbitrary geometry, which is very common in finite element meshes [7]. However, the quality of the elements created by the whisker weaving algorithm are often poor [16]. It has been demonstrated that the surface mesh of an object constrains the way the volume mesh is created in whisker weaving [7]. Global and local connectivity of the surface mesh plays a role, as well as global and local self-intersections, and element self-tangencies of the surface chords. A self-intersection is when a chord intersects itself. A self-tangency exists when a surface chord is parallel to itself with no other chords located between the region of tangency. Figure 2.12 shows a surface self-tangency. Self-intersections and self-tangencies will be explained in greater detail in the following chapters.
The chords and sheets used by the whisker weaving algorithm are part of a bigger data structure, known as the dual. The dual and its components are explained in the next chapter as a precursor to understanding surface splicing.
3 THE DUAL OF A MESH

3.1 The Dual 2-D Triangle

The work of this thesis focuses on quadrilateral meshing. However, since original work on the dual of a mesh was developed for triangular meshing the dual of a mesh will be introduced considering triangular elements. The best possible triangular connectivity of a set of points on a surface is the Delauny-triangulation of a surface. The dual of the Delauny triangulation is known as the Voronoï diagram [18]. Although it is called the Voronoï diagram, it is more like a graph that is overlaid on a set of points. This graph is used to connect groups of three points that contain no other points inside them except the three that are to be connected to form a triangle.

The Voronoï diagram is achieved by marking off every point on a surface that is closer to one of the specified points than it is to any other point. In other words, a line is drawn from each point to every other point it is next to and the perpendicular bisector of that line is drawn. All the perpendicular bisectors of a set of lines from a point then are connected to form a closed polygon. This Voronoï tile marks off the region that is closest to this point than any other. All these tiles create vertices in the Voronoï diagram. These vertices represent the circumcenter of a triangle. The three points that are on this triangle create the Delauny triangulation of the surface. The Voronoï diagram creates triangles that contain no other defined points on the surface except there own. A Voronoï diagram is
shown in Figure 3.1. The Delauny triangulation of the points shown in Figure 3.1 is shown in Figure 3.2.

Figure 3.1- The Voronoi diagram used to find the triangulation of a set of points on a surface

Figure 3.2- The Delauny Triangulation of the set of points in Figure 3.1
3.2 The Dual for 2-D Quadrilaterals

A 2-D dual for a surface quadrilateral mesh as defined by Murdoch [11] is a collection of centroids and chords. A “centroid” is the dual representation of quadrilateral element, and a “chord” is a dual representation of row of quadrilateral elements. A chord was first introduced in the previous chapter. A centroid is formed at the intersection of two dual chords, and this centroid corresponds to one specific quadrilateral element.

The following figures show a graphical representation of the dual. Figure 3.3 shows a “2-cell” and its associated node. Note that each 2-cell in dual space corresponds to a specific node of the mesh. A 2-cell is an n-sided polygon whose edges are composed of segments of dual chords. A 2-cell is the smallest closed dual loop on a surface. Figure 3.4 shows a centroid in dual space. Each centroid corresponds with a specific quadrilateral element of the mesh.

Figure 3.3- 2-cell created by the intersection of multiple dual chords. The mesh is shown in dotted lines and the dual is shown in solid lines.
The previous figures show dual chords that begin and terminate at the edges. This is the analogue of an all-quadrilateral mesh having an even number of surface boundary intervals. Dual chords can also be created totally on the interior of a surface. Figure 3.5 shows an internal dual chord, which closes on itself.
Figure 3.6 shows the steps to create the dual for a given surface mesh. The chords can be created in any order as long as all chords are generated. All chords have been totally formed when two chords intersect at the face of every quadrilateral element. As the dual is created for a given surface mesh, three rules must be obeyed. These rules are:

1. Dual chords can be nowhere tangent. A tangency of dual chords is shown in Figure 3.7.
2. Two chords always intersect in the interior of a quadrilateral element
3. Only two chords can intersect at any given point.
Figure 3.6- Creation steps of the dual of a surface mesh.

Figure 3.7- Two tangent dual chords.
These rules are required both when creating the dual from a surface mesh, and creating a dual with no existing mesh. When a dual is created previous to the mesh, it is not constrained by a surface mesh and the above rules are easier to violate. Table 3.1 shows a summary of the dual entities explained in this section. Figure 3.8 shows the entities summarized in Table 3.1. Figures 3.9 and 3.10 show mesh entities and dual entities respectively.

Table 3.1- Relationship of mesh and dual entities in two dimensions.

<table>
<thead>
<tr>
<th>Mesh Entity</th>
<th>Dimension</th>
<th>Dual Entity</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face</td>
<td>2</td>
<td>Centroid</td>
<td>0</td>
</tr>
<tr>
<td>Edge</td>
<td>1</td>
<td>Chord</td>
<td>1</td>
</tr>
<tr>
<td>Node</td>
<td>0</td>
<td>2-Cell</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 3.8- Mesh and dual entities.
3.3 The Dual of 3-D Hexahedra

The dual in three dimensions as defined by Murdoch [11] is an extension to its two dimensional counterpart. Using Murdoch's terms the dual of an all hexahedral volume mesh is the spatial twist continuum. The dual entities in three dimensions are created in a similar fashion to the two dimensional dual. Connecting opposite faces of a row of hexahedra creates a dual chord in three dimensions. A chord is the dual of a row of hexahedral elements. Figure 3.11 shows a dual chord.
A sheet in dual space represents a topological layer of hexahedra in a volume mesh. Where three dual sheets meet a centroid is formed that is the dual of a hexahedral element. A 3-cell in three-dimensional dual space corresponds to a node in the mesh. Figure 3.12 shows a dual sheet. Table 3.2 shows a summary of the three dimensional dual.

<table>
<thead>
<tr>
<th>Mesh Entity</th>
<th>Dimension</th>
<th>Dual Entity</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hex Element</td>
<td>3</td>
<td>Centroid</td>
<td>0</td>
</tr>
<tr>
<td>Face</td>
<td>2</td>
<td>Chord</td>
<td>1</td>
</tr>
<tr>
<td>Edge</td>
<td>1</td>
<td>2-Cell</td>
<td>2</td>
</tr>
<tr>
<td>Node</td>
<td>0</td>
<td>3-Cell</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3.2- Relationships of a dual mesh and mesh entities in three dimensions.
3.4 Attributes of the Dual

The previous figures illustrate the great benefits of the dual of a mesh. By looking at the dual of a mesh it is easy to see how whole groups of elements interrelate. If a sheet, for example, is moved or deleted it is easily apparent at how moving or deleting this sheet affects the rest of the mesh. The dual gives a visual continuity to the mesh that is not as noticeable in mesh space. The dual of a mesh also gives the ability to see global and local connectivity of the mesh [7].

In addition to monitoring the correlation between elements, the dual also gives the ability to alter the mesh on a large or small scale. Whole sheets can be moved or just small portions of the sheets. This in turn will alter a whole sheet of hexes or just a small group of hexes.

Lastly, when working in dual space, chords and sheets can be manipulated independently of one another [16]. In mesh space the same does not hold true. A sheet or row of hexes cannot be moved independently of the other hexes surrounding the area.
being changed, since the changing of a hex also entails the changing of nodes, edges and faces related to the hex being altered.
Chapter 2 briefly described some of the problems that exist with current meshing algorithms. Each algorithm has its own set of constraints so that the algorithm is robust enough to handle most surfaces and volumes. However, these constraints introduce problems when trying to mesh complex geometries. Some limitations of meshing algorithms and how they can be remedied via the dual are explained in the following sections.

4.1 Non-Editability of Existing Surface Meshes

Presently in the CUBIT [5] mesh generation tool kit, a mesh is not editable once it has been created. In some cases it is desirable for the user to manually change a mesh in certain problem areas. The situation often occurs when a given geometry is easily meshable by an unstructured meshing algorithm such as paving. Figure 4.1 shows such a case. The object in Figure 4.1 was meshed using the paving algorithm. However, due to constraints in the paving algorithm, the element interval in the “thin” direction changes from three to two and back to three. The mesh in Figure 4.1 can only be changed if the mesh is deleted and a new surface-meshing algorithm is specified and the surface is re-meshed. For this reason it would be beneficial to modify or edit the mesh.
However, changing a mesh after it has been created can introduce its additional problems. If an element is deleted from a mesh, then it must be determined what to do with the nodes and edges that now are not fully utilized. In other words, an element cannot be edited or deleted independently of the elements surrounding it. This constraint can be bypassed if the dual representation of a mesh is used instead of the mesh itself. Dual chords can be moved and deleted independently of the other chords on the surface or in the volume. The dual of the mesh introduces a powerful new capability of editing a mesh.

4.2 Self-Intersecting Chords on a Surface

Algorithms such as paving create good quality surface meshes but do not honor the self-intersection rule that is needed for a volume mesh with algorithms such as whisker weaving [17]. In addition, self-intersections need to be identified on a given surface and then removed so that whisker weaving can successfully mesh the volume. The current whisker weaving algorithm automatically removes self-intersections, but often does it in a manner that is undesirable. A simpler method of chord self-intersection removal is desirable.
Figure 4.2 shows an example of a self-intersecting surface mesh. The line in the figure represents the dual chord that self intersects. The introduction of the dual representation of a surface mesh would allow tracking of these surface self-intersections and eventually the capacity to remove them.

Figure 4.2- Self Intersecting dual chord on one surface.

4.3 Global Self-Intersections

The whisker weaving algorithm not only removes chords that self intersect on one geometric surface, but also with chords that self intersect globally (i.e. chords that traverse multiple surfaces). Figure 4.3 shows such a globally self-intersecting chord. Introducing the dual representation of all surfaces and keeping track of what local chords intersect with each other facilitates in monitoring the self-intersections of a mesh.
4.4 Self-Tangent Element Chords on a Surface

Chapter 2 briefly explained element self-tangencies that can exist in a mesh. An element self-tangent chord on a surface is one where two rows of quadrilateral elements that belong to the same chord share at least one edge. The ability to find and remove self-tangencies will allow algorithms such as whisker weaving to create better quality meshes. A self-tangent surface chord is shown in Figure 4.4.
4.5 Global Self-Tangent Chords

Global self-tangencies, like global self-intersections, are found when chords are created across multiple surfaces. A global self-tangency is created when rows of elements that have the same global chord are adjacent to one another and share edges. A global self-tangency is shown in Figure 4.5.

![Global self-tangency](image)

Figure 4.5- Global self-tangency

Mesh editability, self-intersections and self-tangencies can be addressed by utilizing the dual of the mesh. The algorithm presented in the following chapters is developed to this need.
5 THE SURFACE SPLICING ALGORITHM

The concept of the dual explained in the previous chapter is the basis of the Surface Splicing Algorithm. The name “surface splicing” comes from the idea to “splice” or connect two well-defined surface meshes by means of the dual. Throughout the development of the algorithm it became apparent that this algorithm could do more than just connect two surfaces. The algorithm is explained in this chapter.

Surface Splicing uses a series of steps to create a surface mesh. It is assumed that previous to these steps that the boundary curves are meshed. The steps to create the mesh are:

1. Create the dual, either automatically or by manually.
2. Traverse the dual and create 2-cells
3. Find existing nodes and create new ones
4. Connect nodes to create quadrilateral elements

Before these steps are explained in detail, the definitions used for data entities will be given to facilitate understanding Surface Splicing Algorithm. Many of the data types in the algorithm were introduced in chapter 3.
5.1 Definitions

2-Cells: A closed loop of splice chord segments that create an n-sided polygon in dual space. The 2-cell is the lowest level closed loop in dual space. The 2-cell is dual to a node, meaning one two cell exists for every node on the surface mesh. Figure 5.1 shows the components of the dual.

![Diagram showing components of the dual]

Figure 5.1- Components of the dual. The numbers represent the ids of the chords and junctions on the surface.

Dual Junction: A dual junction is a point on a splice chord. It can be either an intersection of two splice chords or a control point on a splice chord. A dual junction can have either two splice chord segments joining into it or four splice chord segments joining into it. A junction that is a centroid has four adjoining splice chord segments. The name junction is used since a junction does not always define a centroid, but is defined by an intersection of two or four splice chord segments. A dual junction must know if it is in the interior of a surface or on a boundary. A dual junction with four connecting chord segments, i.e. a centroid, is dual to a quadrilateral face on the surface mesh.
Splice Chord: A splice chord is a collection of one or more splice chord segments. It is the dual of a row of quadrilaterals in a mesh. A splice chord can start at an edge and terminate at another edge (i.e. a chord with two end points), or it can loop back and close on itself (i.e. a chord with no endpoints). A splice chord must contain at least one splice chord segment.

Splice Chord Segment: A splice chord segment is a piece on a splice chord that is between two junctions. A group of ordered segments defines a splice chord.

5.2 Steps of the Algorithm

5.2.1 Step 1-Create the Dual

The first step of surface splicing is to create the dual. The splice chord is the basis of the dual and it can be created in several different ways. It can be created by connecting two edges on the same surface, by connecting a series of junctions together, or by connecting edges to junctions. The dual can also be created automatically. This will be explained in the next chapter. When creating a chord, it must always start and terminate at two unique edges or start and terminate at the same junction. Figure 5.3 shows a chord created on a rectangular surface. Each chord in the dual has a unique identifying number, i.e. an ID. As chords cross, junctions are automatically created, and the chord is broken into segments. Each junction can store up to four connecting chord segments. These segments are stored in an array in a counter-clockwise order. Figure 5.2 shows the counter clockwise order of segments on a junction.
Figure 5.2- Counter-clockwise ordering of splice chord segments.

The counter clockwise orientation is important for mesh creation, and it will be explained in a later step. Figure 5.4 shows multiple chords created on a face with the junctions that are created at the intersections. Figure 5.5 shows a chord created on the interior of a surface that starts and terminates on the same junction.
Figure 5.3- Surface with one splice chord and two junctions created.

Figure 5.4- Multiple chords on a surface with junctions created at the intersection. Chord segments are also shown.
5.2.2 Step 2- Traverse the Dual and Create 2-Cells

With the surface dual complete, the Surface Splicing Algorithm is ready to traverse across the dual mesh and create individual 2-cells. Figure 5.6 shows a complete dual mesh on the surface.
2-cells are all the n-sided polygons that exist on the surface. Figure 4.7 will be used as a reference for the explanation of the algorithm.
The steps of the algorithm go in the following sequence:

1. Find the next interior junction (junction 1).
2. Find a pair of adjacent segments on junction 1 (segments A and B).
3. Take the second segment in the pair (segment B) and find the junction opposite of junction 1 (junction 2).
4. At junction 2 find the chord adjacent to segment B (segment C).
5. Repeat steps 3 and 4 until the polygon closes or you reach an edge or a segment that has already been used twice to create a 2-cell.
6. Create a node at the center of an interior 2-cell, or associate a node already existing on the edge for a boundary 2 cell.

If the polygon closes, then a new 2-cell is created, and the segments are marked as being used once. The marking is used to prevent a segment from being used in more than two 2-cells. If a segment has been used two times or the polygon does not close, then the algorithm proceeds to the next segment pair on the junction. There will be as many segment pairs as there are segments, e.g. four segments creates four pairs. Figure 5.8 shows a junction with four segments and some of its associated pairs.
5.2.3 Step 3- Find Existing Nodes and Create New Ones

When the Surface Splicing Algorithm has completed traversing the surface and it has visited every interior junction, it then moves onto the next step: creation of nodes. Nodes are either created or found. For every interior 2-cell created a new node is created. For every boundary 2-cell created, a boundary node is found that already exists. Boundary nodes already exist because the curves are already meshed. In the case of the interior 2-cell, the creation of the node is quite simple. The node is created at the average of all the coordinates of the junctions of the 2-cell.

Finding the node for the boundary 2-cell is more complicated. A boundary 2-cell contains two boundary junctions, meaning junctions that are located on a geometric curve. These two junctions know the element edges that they sit upon. Consequently, if a 2-cell contains boundary junctions, then the element edges of those boundary junctions are found. These edges will share a common node since they are adjacent edges. This
common node is found and this becomes the node associated with this 2-cell. Figure 5.9 shows a boundary 2-cell and its associated edges and node.

As each 2-cell is created, whether it is interior or boundary, it is permanently associated with its node. This will assist in creating the element faces in the next step. The id on the node is the same id that is used for the 2-cell.

5.2.4 Step 4- Connect Nodes to Create Quadrilateral Faces

Now that all of the nodes are created, they are ready to be connected to form quadrilateral elements. To connect the nodes correctly, some data needs to be created about how each splice chord segment on the face is related to already existing nodes (i.e. boundary nodes), or how each is related to newly created nodes.
As the Surface Splicing Algorithm traverses across a surface, it stores data about each segment. If the segment is connected to a geometric edge, then that segment is associated with the two nodes on that edge. Figure 5.10 shows a boundary segment (as it is called) and its associated nodes.

![Boundary Segment and Associated Nodes](image)

Figure 5.10- Boundary segment and its associated nodes.

If a segment is an interior segment, meaning it has only associated interior junctions, then the associated nodes need to be found. This is done using the following steps:

1. Find interior segment
2. Find the two 2-cells that use this interior segment
3. Find the node associated with each 2-cell

4. Associate the two nodes with the interior segment

At this point each segment knows about the two nodes with which it is associated. It is now possible to create the new faces. In this step, the segments connecting into each junction are used. Each junction has a group of associated counter-clockwise segments. Each face that will be created needs to have its nodes and edges in counter-clockwise order so they can be generated properly. Figure 5.11 shows a quadrilateral element and the order the edges and nodes are in.

Figure 5.11- Quadrilateral element showing nodes and edges stored in ccw order. Nodes are identified with numbers and edges with letters.

Since the nodes associated with each segment are in an arbitrary order, a few steps need to be taken to insure that the nodes for each new face are in the right order.

Four nodes are needed to create a new element. However, a junction, if it has four associated segments, will have eight nodes—two for each associated segment. In
other words, some duplication exists. This duplication insures that node connectivity is correct. The steps to create a new face are:

1. Get two adjacent segments on a junction
2. Get the two nodes associated with each segment
3. Of the four nodes (two from each segment), find the one node that is common. This common node becomes the middle of the three nodes
4. Get the next segment on the list and find the one node on segment three that is not also on segment two
5. Create the face with the four nodes.

Since each centroid is the dual to exactly one face, it is simple to create exactly the right amount of faces. Just as when the algorithm was traversing the junctions to create 2-cells, it also traverses all the junctions once to create the right number of new faces. Figure 5.12 shows the mesh created, using the Surface Splicing Algorithm, from the dual in Figure 5.6.

Figure 5.12- Dual and mesh of a rectangular surface. Dual entities have id numbers shown on them.
5.3 Global Connectivity of the Dual

This section explains how the global connectivity algorithm is implemented in surface splicing. The work explained in this section is also the first step to gathering more data on how the surface mesh constrains the volume mesh. Figures 5.13 and 5.14 show examples of global and local self-intersections.

Figure 5.13- Global self-intersection.

Figure 5.14- Local self-intersection
5.4 Creating the Global Connectivity of the Dual

The global connectivity of a dual is easily created by using the information that is stored with a dual junction. Recall that a dual junction knows about all the splice chord segments attached to it. A splice chord segment knows who its owner is; that is, each segment knows its associated splice chord. The algorithm that establishes total connectivity of the mesh uses the following sequence of steps each time a chord segment is created.

1. Check the two junctions on the segment to see if they are boundary junctions.

2. If a boundary junction is found, check to see if it has another chord segment attached to it. If another chord is attached to the junction, create a new super splice chord or relate it to the already created super chord.

The steps just listed will be explained in further detail.

5.4.1 Step 1- Check the two junctions on the segment to see if they are boundary junctions.

A boundary junction is shown in Figure 5.15. It is a junction that exists on a curve of the geometric surface on which the dual is created.
The boundary junction is the key to associating chords from different surfaces. Each chord in surface splicing is only associated with the surface on which it resides. Each junction on a boundary, however, is known by both of the surfaces that join into the edge on which the junction resides. In this step, if a boundary junction such as the one shown above is found on a chord segment then the algorithm proceeds to step 2.

5.4.2 Step 2 - If a boundary junction is found, check to see if it has another chord segment attached to it. If another chord is attached to the junction then create a new super splice chord, or relate it to the already created super chord.

When a boundary junction is located, it is important to determine if the boundary junction has another chord segment associated with it other than the one used to find this boundary junction. If no other chord segment is attached to the boundary junction, then there is no way to create global connectivity. If another segment is found, then a chord can be created and a unique id is given to this new “super chord”. A super chord is a global chord that is created by many splice chords. Each chord segment now also knows about the super chord that it belongs to. Figure 5.16 shows an explanation of step 2.
Step 2, as shown in Figure 5.16, works very well if the chords are created in order around the surfaces of an object. Creating chords in order means that after one chord is created the next chord is created on an adjacent surface. This process continues by creating chords on adjacent surfaces until a closed super chord is created. This algorithm also handles the case if the chords are created arbitrarily. If two surface chords exist on non-adjacent surfaces then no connectivity exists until a surface chord is created that links the two chords on non-adjacent edges. This newly created surface chord will have boundary junctions that have other chords on both of the junctions. The algorithm solves this problem quite easily. It only considers one boundary junction at a time. Previous to the addition of the chord that links the two non-adjacent surface chords no super chords have been created. At this point the first two chords on adjacent surfaces are found and a new chord is created. Next, the other boundary junction is checked and it is found to have another chord opposite it. This chord is added to the super chord that already exists on the other two chords. The algorithm relates all new chords to an already existing
chord if it does exist. This happens unrelated to the way the chord is created. Figures 5.17 and 5.18 show how this step progresses.

Figure 5.17 - Two chords that are on non-adjacent surfaces.

Figure 5.18 - Chord created on front surfaces that connect the two non-adjacent chords, allowing the creation of a new super chord.
Once all of the dual chords are created, either manually created or automatically generated, the information on the junctions is queried. Since all the chords have been created, then the information about local and global self-intersections has been generated. Two questions are asked for each junction on a surface:

1. Are there four chord segments connected to the junction in question?
2. If so, do the segments have the same local and global owner?

If the chords have the same local owner, then a local self-intersection has been found. If the chords have the same global owner, then a global self-intersection has been found. In either case, the self-intersection can be manually removed and then algorithms such as whisker weaving can be invoked to mesh the volume.

5.5 Finding Self-Tangencies of the Dual

Self-tangencies were introduced in Chapters 2 and 4. The method of finding global and local self-tangencies is discussed in this section. The following steps are used to find self-tangencies on a mesh:

1. Get all element edges on all surfaces of the volume and store them in a list.
2. Start at an edge in the list and obtain its opposite edge on one of the quadrilateral elements that it resides upon.
3. At each edge pair traversed, obtain the other two edges not being traversed on the quadrilateral and store them uniquely in a list. If at least one of the edges is already contained in this list then a self tangency has been found.
4. Remove the two edges traversed from the traversal list.
5. Go to step 2. Make the new start edge the opposite edge found in step 2. If the opposite edge is the original start edge then a closed loop has been created. In this case clean out the unique list in step 3 and start the process over.

6. Continue this process until no more edges exist in the list.

The following section will explain these steps in greater detail.

5.5.1 Step 1- Get all element edges on all surfaces of the volume and store them in a list.

This step is very simple. In the CUBIT mesh generation toolkit a volume entity knows about all the element edges that sit upon its surfaces. These edges are stored in a master list, called edge_list.

5.5.2 Step 2- Start at an edge in the edge_list and obtain its opposite edge on one of the quadrilateral elements that it resides upon.

Now that edge_list has all the element edges on the surfaces the first edge can be taken from the list and used to traverse across edges on the surface. Figure 5.19 shows what is meant by opposite edges.

![Figure 5.19- Quadrilateral element showing opposite edges as well as the edges being added to tangency_list.](image-url)
5.5.3 Step 3 - At each edge pair traversed, obtain the other two edges not being traversed on the quadrilateral and store them uniquely in a list. If at least one of the edges is already contained in this list then a self-tangency has been found.

Since each quadrilateral element knows about the four edges that reside upon it, it is easy to find the two edges that are not being traversed. These two edges are added to a list, tangency_list. Before addition to the list the list is checked to see if it already contains the edges, if it does contain the edges then a self-tangency has been found. Figure 5.18 shows the edges in bold that will be added to the tangency_list, and checked.

5.5.4 Steps 4, 5 and 6

These steps are just the continuation of the first three steps. Step 4 removes the two edges that have just been traversed. These are removed since an edge is associated with only one dual chord. In step 5 the process starts over and step 6 assures that this algorithm continues until every edge has been visited. Figure 5.20 shows a self-tangency that was previously shown in Chapter 4.
Figure 5.20- Self-tangency on a surface mesh. Bolded edges show the area of tangency.
6 EDITING EXISTING SURFACE MESHES

6.1 Creating the Dual from and Existing Surface Mesh

The real power of the Surface Splicing Algorithm lies not in its ability to create totally dual-based meshes, but in the ability to edit and create the dual from an existing surface mesh. By editing a mesh instead of recreating a mesh time can be saved generating a mesh. Currently mesh generation is the most time consuming step in the finite element method [1].

A sub-algorithm of the surface splicer is the surf mesh to dual tool. This tool has the ability to either create a total dual representation of a given surface mesh or a dual representation of a given area of the mesh. This chapter will explain how the surf mesh to dual tool works. Examples of edited meshes using this tool are shown in the next chapter. The surf mesh to dual tool uses the following steps to create the dual from the existing surface mesh.

1. Find all boundary edges on a surface
2. Start at any boundary edge and traverse across opposite edges until another boundary edge is reached
3. As each opposite edge pair is found, create three junctions, one in the center of the face and one on each edge. Create segments between each of the junctions.
4. Mark both boundary edges as being traversed

5. Find any edges on the surface that have not yet been traversed. Start on one of the non-traversed edges and continue until you reach the start edge again.

6. Go to step 6

The above steps will now be explained in greater detail. In addition, graphics and examples of the steps will be shown.

6.1.1 Step 1- Find all Boundary Edges on a Surface

As stated in the Chapter 5, all splicing chords must either start and stop on a boundary or start and stop on the same edge (Step 5), i.e. close on itself. Another way to consider this is that a splice chord must either have two or zero endpoints. The first step in finding the chords that already are exiting on a given surface mesh is finding the boundary edges on a surface. The boundary edges are all the edges that lie upon a geometric boundary. Figure 6.1 shows the surface mesh of a cylinder with the boundary edges highlighted.

![Figure 6.1- Meshed circle (highlight shows boundary edges)]
6.1.2 Step 2- Start at any boundary edge and traverse across opposite faces until another boundary edge is reached

Once all the boundary edges on a surface have been found, then the chords can be created. All splice chords are created by connecting opposite edges on a quadrilateral element. Opposite edges are connected in this step until another boundary edge is found. Figure 6.2 shows the meshed circle with a chord being created by stepping across opposite edges.

![Figure 6.2- Surface of a circle with chords of opposite edges highlighted.](image)

6.1.3 Step 3- As each opposite edge pair is found create three junctions, one in the center of the face and one on each edge. Create segments between each of the junctions.

On each edge that is successfully traversed, a dual junction is created as well as a junction in the center of the quadrilateral element being traversed. Each of these new junctions is connected by a splice chord segment. The chord is created in this manner as each edge is traversed and eventually a chord is made. Figure 6.3 shows where the junctions are created on the quadrilateral element.
6.1.4 Step 4- Mark all edges that have been traversed

Step 2, 3 and 4 work simultaneously. As each edge is traversed, it is marked as being traversed. If the creation of the chord is completed successfully, then the edges stay marked as being used. If the creation is not completed, then the edges are unmarked. Each edge can only be associated with one chord, while each element is associated with two chords. Therefore, each edge is marked so that they are only associated with one chord.

6.1.5 Step 5- Find any edges on the surface that have not yet been traversed

Step 5 proceeds similar to step 2 except for the start and end points. In this step, those points are the same; in step 2 they are different. Figure 6.4 shows interior edges being traversed to create an interior chord. Figure 6.5 shows a completed interior chord. In this step, the final edge needs to be checked to see if a junction already exists on the final edge. If the junction already exists, a new one is not created on this edge and then the last chord segment is created with the already existing junction.
6.1.6 Step 6- Start on one of the non-traversed edges and traversed opposite edges until you reach the start edge.

After all the boundary edges have been traversed, a check of all the edges must be made to see if any interior loops exist. These are loops that have no endpoints. They start and end at the same interior edges without ever meeting the boundary of the surface. These edges are found and stored in a list to traverse. Figure 6.6 shows the complete dual of the surface mesh.

Figure 6.4- Partial traversal of interior chord.
While the dual creation tool can be used to create the dual for an entire mesh, it is usually more desirable to create a dual for only a portion of a mesh. The procedure to create a dual of a portion of a mesh is similar to that previously described. The only
difference is that a user must specify what faces are required for the dual to be created. The algorithm then creates a boundary around the specified faces, deletes those faces and creates the dual. This dual can then be edited and the new mesh can be created. The dual can also be deleted and the new dual can be entirely recreated. Since only the area where the dual is created is affected, the editing does not change any of the non-selected elements on the mesh.
7 EXAMPLES

Creation of the dual from an existing mesh, which was explained in the previous chapter, allows for changing only regions of the mesh. This chapter focuses on examples of mesh editing.

7.1 Example 1-Removal of Self-Tangencies on a Surface Mesh

This example shows how the removal of self-tangent chords on a surface can improve the quality of the mesh. Figure 7.1 shows the mesh originally shown in Chapter 2. The mesh was created using the whisker weaving algorithm. Table 7.1 shows the mesh quality metrics of the mesh in Figure 7.1.

The quality metrics used in Table 7.1 are scaled jacobian, skew, aspect ratio and shape [6]. The scaled jacobian is found by dividing the minimum jacobian by the lengths of the three edge vectors. The skew is found by finding the maximum cosine of the angle between edges at the hex center. The aspect ratio is the maximum edge length ratios at the hex center. Lastly, the shape is found by $3/\text{Mean Ratio of the weighted jacobian matrix}$. The optimum ranges for these metrics are, scaled jacobian($0.5-1$), skew($0-0.5$), aspect ratio($1-4$), and shape($0.3-1$) [6]. The optimum values for these metrics are scaled jacobian($1$), skew($0$), aspect ratio($1$), and shape($1$).
The current CUBIT [5] whisker-weaving algorithm automatically removed initial surface self-intersections in Figure 7.1, however it modified the surface mesh in a manner that has an adverse effects on both the surface mesh and volume mesh. The modified surface mesh has a very dense area of quads and the self-tangency shown in Figure 7.1 adversely affects the interior hexes.

The CUBIT mesh generation toolkit [3] provides for the manual deletion of dual sheets. The self-tangent dual sheet of the hook object is shown in Figure 7.2. Such sheets result in poor quality meshes. In general, smooth sheets with little curvature produce high quality elements.
The dual sheet shown in Figure 7.2 was deleted from the volume mesh in Figure 7.1. Sheets continued to be deleted until the new mesh given in Figure 7.3 was created. Table 7.1 shows the mesh quality metrics for Figure 7.3. Figure 7.3 and Table 7.1 show that if meshes can be edited or manipulated in dual space then the metrics that measure mesh quality can be improved. Currently, except for the ability to delete a dual sheet [3], the ability to manipulate or move sheets does not exist.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Average</th>
<th>Std. Deviation</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Before</td>
<td>After</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaled Jacobian</td>
<td>0.5372</td>
<td>0.6208</td>
<td>0.3155</td>
<td>0.1812</td>
</tr>
<tr>
<td></td>
<td>-0.9864</td>
<td>0.08475</td>
<td>0.007467</td>
<td>0.03209</td>
</tr>
<tr>
<td>Skew</td>
<td>0.5922</td>
<td>0.5336</td>
<td>0.2381</td>
<td>0.2133</td>
</tr>
<tr>
<td>Aspect Ratio</td>
<td>2.2</td>
<td>1.92</td>
<td>1.759</td>
<td>0.6042</td>
</tr>
<tr>
<td>Shape</td>
<td>0.5721</td>
<td>0.6267</td>
<td>0.2749</td>
<td>0.1533</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.1634</td>
<td>0.9890</td>
<td>0.9844</td>
</tr>
</tbody>
</table>
7.2 Example 2- Changing the Interval Count Across a Surface Mesh.

Figure 7.4 shows the example that was used in Chapter 4, a mesh that does not keep a consistent amount of intervals across a surface. Figure 7.5 shows the deleted mesh region.

Figure 7.4- Undesirable mesh interval on the surface.
Figure 7.5- Deleted dual and mesh

Figure 7.6 shows the new dual created and finally Figure 7.7 shows the new mesh. Notice that only the mesh in the region specified was changed and the rest of the mesh was left completely unchanged.

Figure 7.6- Newly created dual.
7.3 Example 3- Editing of a Mapped Surface Mesh

The next example begins with a mapped five by five surface mesh. Figure 7.8 shows the original mesh. Figure 7.9 shows the dual that was automatically created for the surface mesh in Figure 7.8.

Figure 7.8- 5 by 5 mapped mesh.
With the dual of the mesh created it can be manipulated in many different ways. In this example, chord 3 was deleted and then a new chord was created by connecting existing junctions on the surface. Figure 7.10 shows the new chord that was created, and Figure 7.11 shows the final mesh that was created from the new dual.
7.4 Example 4 – Inserting Chords into aMapped Surface Mesh

The last example, shown in Figure 7.13, shows the ability to simply add new dual chords into an existing dual. The example shown in Figure 7.13 essentially inserts loops of quadrilateral elements by inserting closed chords in the dual space. The loops were inserted to spell “10th I.M.R.” The insertion of these loops had little effect on the structured mapped mesh of the surface, which is shown in Figure 7.12. The mesh created from the dual in Figure 7.13 is shown in Figure 7.14.
Figure 7.13- Dual representation of the mesh shown in Figure 7.14, with additional chords inserted to spell "10th I.M.R."

Figure 7.14- Insertion of dual chords to spell 10th I.M.R.
8 CONCLUSION

This chapter summarizes the work presented in this thesis. After the summary, future areas of research in surface splicing are discussed.

8.1 Summary

Many of the meshing algorithms used by conventional mesh generation software create good quality structured and unstructured meshes. However, the algorithms that create these meshes generally lack knowledge of the overall connectivity of the mesh. The dual of a mesh provides some global information about the mesh. It supplies data on connectivity, and also helps to visualize and understand some of the constraints on a mesh.

The Surface Splicing Algorithm, developed in this thesis, allows the creation and manipulation of the dual on a surface. The dual can then be traversed and an all-quadrilateral mesh can be created on the surface.

The work herein also explained how the dual could be created from an existing surface mesh. The chords and junctions in the dual can then be manipulated independently from one another. Since the chords and junctions can be manipulated independent of other chords and junctions, the mesh is easily editable or altered.

Finally, this thesis explained the beginning of global connectivity of a surface mesh and surface dual. Splice chords from multiple surfaces can be connected to one
another to make global chords. With the information stored in a dual junction, global and local self-intersections and global and local self-tangencies can be found. Knowing where these self-intersections and self-tangencies reside, they can be removed and then algorithms such as whisker weaving can proceed with the potential of forming high quality, all hexahedral meshes.

8.2 Future Research

1. Extend surface splicing to work on multiple surfaces simultaneously.

Currently, surface splicing can only be done on one surface at a time. This process is somewhat tedious because of much user interaction. Presently the user must create all the components of the dual manually. Future work will change this so that the computer can generate the dual automatically without an existing surface mesh. Also, the work could be done on multiple surfaces at a time, while global and local self-intersections are being tracked.

2. Continue research in determining the surface mesh constraints necessary for producing high quality volume meshes via algorithms utilizing the dual.

Even with the work presented in this paper, only rudimentary information exists on how the surface mesh constrains the dual. The work presented here makes the visualization of the dual on a surface readily available. This facilitates the understanding of how the surface mesh and its dual constrain the volume mesh. Presently, algorithms such as whisker weaving do not handle self-intersections adequately. There are likely
some more constraints on the surface mesh and its dual that would help to make a volume mesh better.

3. Automatic removal of self-intersections on a surface

Currently, self-intersections are easily found on a surface, but then the question remains what is the best way to remove the self-intersection. Figure 8.1 shows a local self-intersection. Figures 8.2 and 8.3 show a local self-intersection can be removed.

Figure 8.1-Local self-intersection

Figure 8.2-Self-intersection changed to two loops.

Figure 8.3-Local self-intersection, after removal
In the case of a self-intersection, the chord can either be changed into two separate chords, it can be untangled to make one chord, or it can be removed. Depending on the constraints that the user needs to meet on the surface, the mesh could be changed to any of the options. Research needs to be done in understanding the best removal methods and how to help the algorithm make that decision. A global self-intersection has even more options of how to change it since it spans across multiple surfaces.

4. Smooth dual chords to take out areas of high curvature.

Dual chords create the highest quality surface meshes when they have no areas of high curvature. Research needs to be done in smoothing the dual representation of a surface mesh. If the dual of a mesh could be smoothed before a mesh is created it would give a higher chance of generating high quality all quadrilateral meshes.
GLOSSARY

Centroid: The intersection of two dual chords. For every centroid in the dual, there exists one quadrilateral element or one hexahedral element. A centroid is the dual of a quadrilateral element or a hexahedral element.

Dual Chord: A chord that connects opposite edges of quadrilateral elements or opposite faces of hexahedral elements. A chord is dual to a row of quadrilateral elements or a row of hexahedral elements.

Dual Junction: A dual junction is the intersection of two splice chords or a control point on a splice chord. A dual junction can have two splice chord segments joining into it or four splice chord segments joining into it. It is very similar to a centroid (defined in chapter 2). The name junction is used since a junction does not always define a centroid of a face, but is only defined by an intersection of two to four splice chord segments. A dual junction knows if it sits on the interior of a surface or a boundary. A dual junction with four chord segments connecting into it is dual to a quadrilateral face on the surface mesh.

Global Self-Intersection: The self-crossing of a dual chord that has been created by traversing opposite edges across multiple surfaces.

Global Self-Tangency: A single row of elements that belong to the same global dual chord and at least one edge on two adjacent elements is shared.
Local Self-Tangency: A single row of elements that belong to the same local dual chord and at least one edge on two adjacent elements is shared.

Local Self-Intersection: The self-crossing of a dual chord that has been created only by traversing opposite edges on one surface.

Splice Chord: A collection of one or more splice chord segments. It is used to define a row of quadrilaterals in a mesh. A splice chord can start at an edge and then terminate at another edge (two endpoints) or it can loop back and close on itself (zero endpoints). A splice chord must contain at least one splice chord segment. Very similar to a dual chord.

Splice Chord Segment: A splice chord segment is one of the pieces on a splice chord that is between two junctions with no junctions in between the two defining junctions. A group of ordered segments defines a splice chord.

Super Splice Chord: A chord that keeps track of multiple splice chords across multiple surfaces. A super splice chord is used to track global connectivity of a mesh.

2-cell: An n-sided polygon with sides consisting of chord segments. For every 2-cell on the surface there exists a single node in the mesh. A 2-cell is the dual of a node.
REFERENCES


3 Borden, Michael, Master's Thesis in Progress, Brigham Young University


15 Shepherd, J.F., Interval Matching and Control for Hexahedral Mesh Generation of Swept Volumes, Published Master’s Thesis at Brigham Young University, 1999.

16 Shepherd, J.F., Personal Communication October 23, 2001

