INTERVAL MATCHING AND CONTROL FOR HEXAHEDRAL MESH GENERATION OF SWEPT VOLUMES

by

Jason F. Shepherd

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 Engineering

Brigham Young University

April 1999

ABSTRACT

INTERVAL MATCHING AND CONTROL FOR HEXAHEDRAL MESH GENERATION OF SWEPT VOLUMES

Jason F. Shepherd

Department of Civil Engineering

Master of Science

Surface meshing algorithms require certain relationships among the number of intervals on the curves that bound the surface. Assigning the number of intervals to all of the curves in the model such that all relationships are satisfied is called interval assignment. Volume meshing algorithms also require certain relationships among the numbers of intervals on each of the curves on the volume. These relationships are not always captured by the surface meshing requirements. This thesis presents a new technique for automatically identifying volume constraints. In this technique, volume constraints are grouped with surface constraints and are solved simultaneously.

A sweepable volume has source, target, and linking surfaces. The technique described in this thesis uses graph algorithms to identify independent, parallel sets of linking surfaces, and determine if they correspond to through-holes or blind-holes. For blind-holes, the algorithm generates constraints that prevent the hole from being too deep in interval parameter space and, thus, penetrating opposite target surfaces. For each linking set, the adjoining source and target surfaces are partially ordered by the structure of the linking set. A small set of representative paths for each linking set is found, and the representative paths for all linking sets are gathered and distilled by Gaussian elimination into a small set of constraints.

ACKNOWLEDGMENTS

In the process of completing work on this thesis, I have learned that the Master's thesis is in actuality the culmination of work from a large set of individuals, for which one gets to take most of the credit. This thesis has been no exception.

I am indebted to Dr. Steven Benzley and Sandia National Laboratories for the opportunity to be a part of this research, and also for help rendered by the CUBIT team, especially Scott Mitchell and David White. A special thanks goes to the BYU CUBIT team of Bob Kerr and Steve Jankovich.

My thanks also goes to Harold and Kathryn Smith for their help, time and support in my schooling, as well as, for letting me use their basement as my playground, and my mom and dad, Bruce and Sydnee Shepherd, for their overwhelming confidence in all of my endeavors.

I would also like to thank Bruce and Patti Brown for the time, attention, and toys they have loaned to me and my family, without which much of this work would have been more difficult.

Lastly, and most importantly, I would like to thank my wife, Stacey, for her patience, love, and support which have always been present in any adventure in which we may participate. I would not be where I am today without her...

Table of Contents

List of Figures

Chapter 1 - Introduction

The finite element method is a powerful engineering tool for analyzing models with complex geometry. However, the technology has also produced it's own set of obstacles. A fundamental step of the finite element method is breaking the geometry into smaller pieces known as finite elements. This step is also known as mesh generation and has proven to be one of the most time consuming tasks in the process. Many algorithms have been devised in an attempt to automate this process, but currently there is no single meshing technique that appears to fulfill all of the mesh generation requirements. Research into mesh generation continues to enhance these algorithms to make them more powerful and robust.

In many cases, the most time and labor intensive task of mesh generation is geometry and attribute preparation prior to meshing. This preparation step includes specifying the number of mesh intervals on each edge of a surface, or interval assignment. This thesis presents the development of an algorithm designed to further automate the process of assigning intervals to sweepable volumes.

The Finite Element Method

The finite element method is a fundamental modeling technique with widespread use and growing popularity in the engineering community. In use since the late 1960's, this technique uses a numerical approximation of partial differential equations to model

the effects of heat transfer, fluid flow, and stress for objects with complex geometry. A drawback of the finite element method is the significant amount of expertise and time required to complete a successful analysis. Much of the research into the finite element method focuses on further automating the process, which would allow personnel with less training to use the process, improve productivity, and achieve more accurate solutions.[1] The most time consuming and expertise intensive part of the finite element method is the discretization of the model's geometry into finite elements, the process known as mesh generation.

The fundamental concept of mesh generation is to approximate the complex object with a grid or collection of simpler objects. The set of elements used to approximate the geometric object is known as the mesh. The mathematical model developed from the finite element mesh allows the computation of physical parameters, such as stress, temperature, pressure, etc. The computed physical parameters of each of the elements can then be used to represent a continuous solution for the whole geometry.

The finite element method is an approximate method, and, therefore, the quality of the mesh can affect the results produced. In practical cases, it is difficult to have perfectly shaped elements because many of the elements must be distorted to fit the geometry. A good mesh will tend to minimize the amount of distortion found in the element. The quality of the mesh generated is largely a function of the technique, or scheme, selected to generate the mesh.

Mesh Generation by Mapping Techniques

Mapping algorithms provide a powerful mesh generation technique because they generate both a regular grid of elements and have a very small percentage of irregular nodes. The mesh shown in Figure 1.1 is an example of a mapped surface mesh (i.e. parameter space mapping).

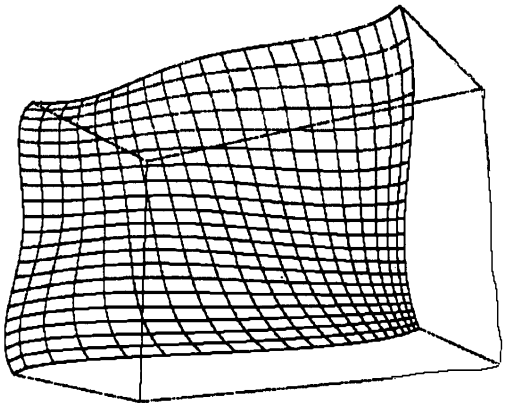


Figure 1.1 - Parametric Surface Mapping

Meshes generated by mapping algorithms have the following advantages[4]:

- Boundary Sensitivity: Mapping a region produces an all quadrilateral mesh that closely follows the shape of the boundary. Rows of elements end at geometric corners, intersect perpendicularly at large interior angles, and follow the contour of smooth boundary segments. Since well-shaped elements are usually critical near the boundary, this characteristic is of particular importance.
- Orientation Insensitive: Different orientations of the geometry do not result in different meshes. Mapped element approaches thus provide repeatability and consistency.
- Regular: A mapped mesh results in a very regular pattern of connectivity among the nodes. An interior mesh node is considered regular if it is connected to four elements. Irregular nodes can appear with mapping techniques, but they are few in number and can be placed away from the boundaries.

Most commercial mesh generation codes[15,16] employ mapping algorithms to generate all-quadrilateral meshes. The mapping techniques rely on picking the four logical corners of the surfaces, and insuring an equal interval counts on opposite sides. A grid can be placed on the surface to match the interval counts on each of the sides, thus producing the all-quadrilateral mesh.

Appropriate assignment of intervals between the four logical corners of a surface is required to produce an acceptable mesh. Improper interval assignment often leads to poor quality meshes, or the subsequent failure of the meshing algorithm. Therefore, automated interval assignment algorithms have been developed using linear programs to ensure interval feasibility across sets of surfaces.[9, 12, 14]

Mesh Generation via Submapping

Since only simple primitives such as rectangular areas and hexahedral regions can be meshed directly with a mapping algorithm, many geometric models do not lend themselves directly to mapping algorithms. Therefore, such geometric models must first be divided, or decomposed, into mappable sub-regions before they can be meshed. This decomposition of the geometry can be accomplished manually or virtually. One algorithm for automatic virtual decomposition and mapping is known as submapping.

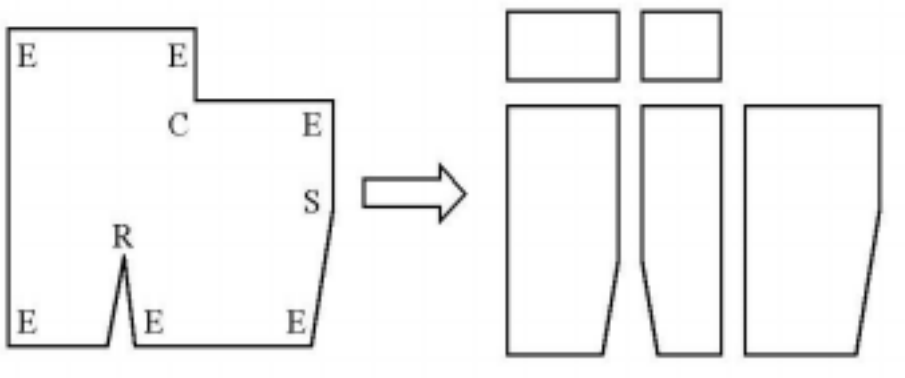


Figure 1.2 - Surface Decomposition by Submapping (R = reversal, C = corner, E = end, S = side)

A submapping algorithm attempts to identify the logical locations at which to divide, or decompose, the solid model by looking for "corners" in the geometry. To accomplish this, the interior angles on a surface are calculated and then classified as ends (~$\pi/2$), sides (~π), corners (~$3\pi/2$), or reversals (~2π). From these angle classifications appropriate division points between sub-volumes can be determined. If an angle is a corner or a reversal, a division must be made at that point. The process continues until all corners and reversals have been eliminated.[13] Figure 1.2 shows an example of a surface decomposition and Figures 1.3 and 1.4 show examples of a volume decomposition and a completed submapping mesh.

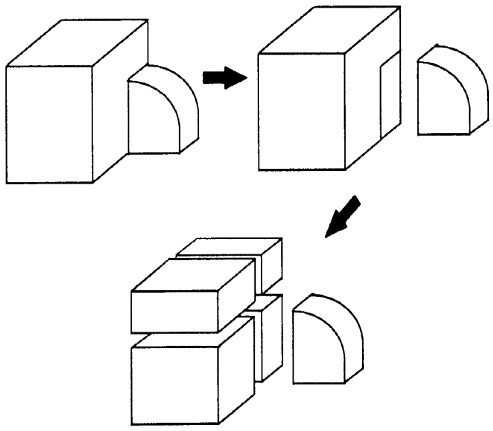


Figure 1.3 - Volume Decomposition by Submapping

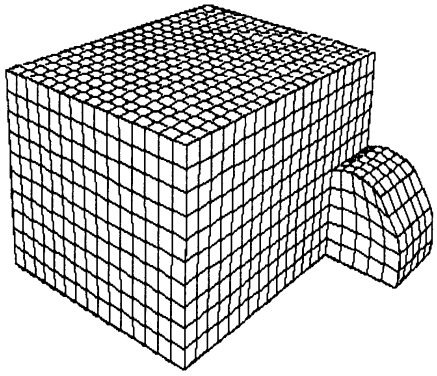


Figure 1.4 - Volume Meshing by Submapping

Mesh Generation via Sweeping

A volume is said to be sweepable if topologically equivalent source and target surfaces are connected by mappable or submappable linking surfaces. Such solids are also known as two and one-half dimensional volumes because a meshed source surface is easily projected layer by layer through the linking surfaces to the target surface. The mesh is not really created at the volume level; rather, it is created at the surface level and projected through the linking surfaces to fill the volume with hexahedra.

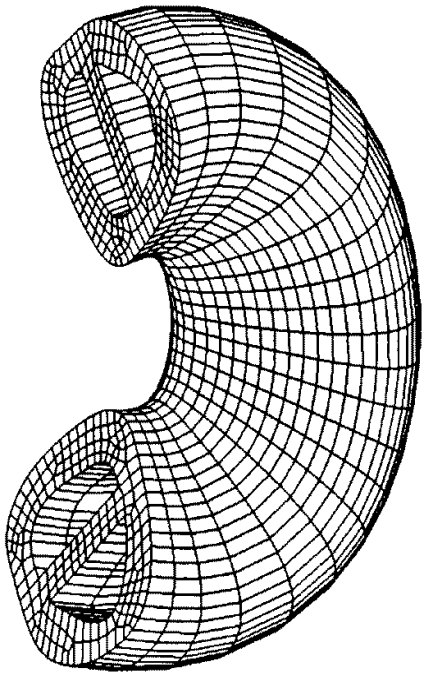


Figure 1.5 - One-to-One Sweep[10]

Sweepable volumes are often classified by the number of source surfaces and the number of target surfaces the volume possesses. These classifications are usually single source to single target, multiple source to single target, and multiple source to multiple target. An example of each of these types of volumes is shown in Figures 1.5, 1.6 and 1.7.

As can be seen in Figures 1.6 and 1.7, the complexity of the geometry, and thus the difficulty of producing an acceptable mesh, increases as the number of source and/or target surfaces increase.

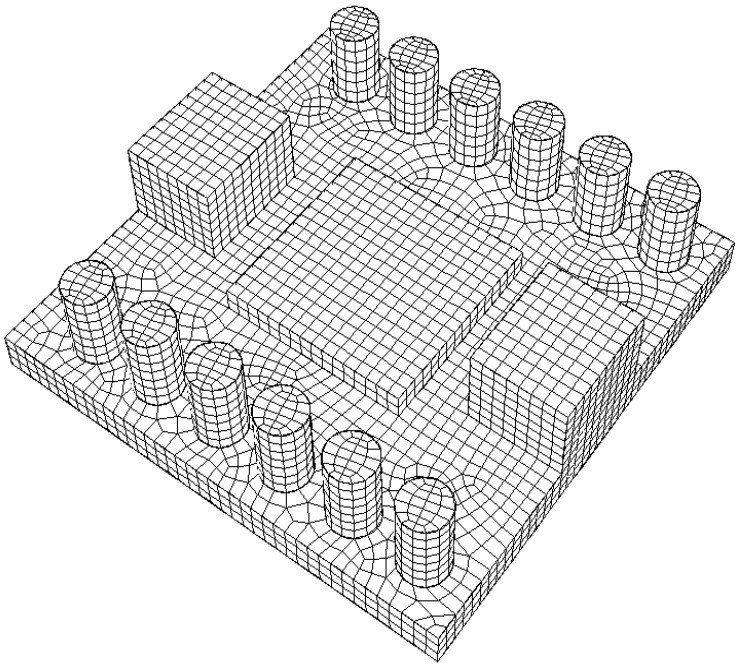


Figure 1.6 - Many-to-One Sweep

Until recently, multiple source to multiple target sweepable volumes have required manual decomposition before the meshing could be completed. Recent research has automated the process for this class of volumes[2, 8] however, because some interval constraints cannot be propagated across the surfaces, the process of assigning intervals is still a manual operation. Because of the inherent complexity of many of volumes, the interval assignment is often difficult and time-consuming.

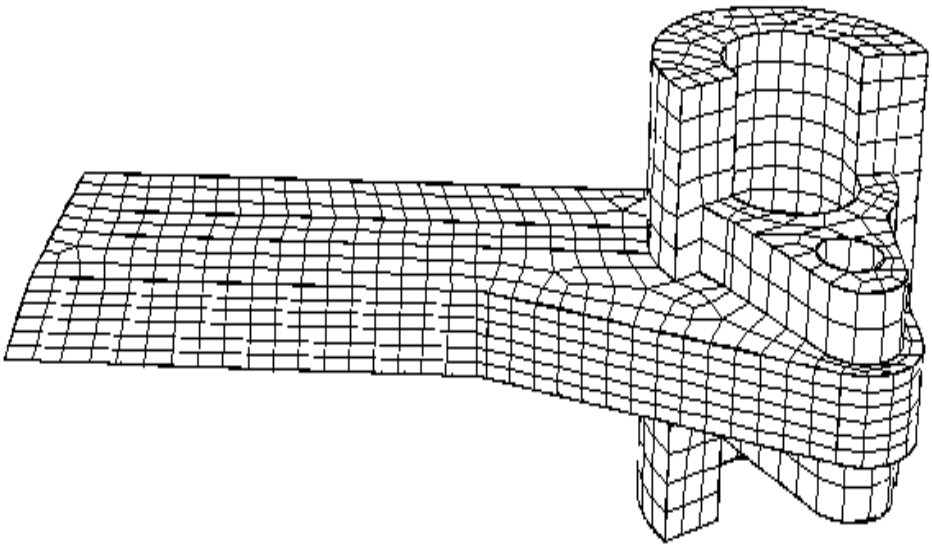


Figure 1.7 - Many-to-Many Sweep

Surface Interval Assignment

The mapping and submapping meshing schemes rely heavily on proper corner picking and edge interval assignment to be successful. However, because the surfaces on a volume or set of volumes are all interconnected through the curves, the interval assignment for the surfaces must be done simultaneously to achieve feasible results. With the use of linear programs, and interval linear programs, an optimized interval assignment for each curve on the volume can be obtained simultaneously.[9, 12, 14] The constraint equations for these linear programs can be derived from each surface's meshing scheme and geometry. For example, mapping and submapping require an even number of intervals bounding the surface, and the interval counts on opposite sets of edges must be equal.

Once the constraint equations for each surface have been formulated, the linear program attempts to produce an optimized interval assignment for each edge. If a solution is found then the new intervals can be assigned to each edge.

Volume Interval Assignment

This thesis presents an enhancement to the interval assignment algorithm described above where only surface constraints were considered in the interval assignment algorithm. Here volume constraints are added to the linear program to guarantee proper interval assignment to a greater population of cases.

Volume constraints are required on many submappable and sweepable volumes.[9] For a simple example where volume constraints are required, consider the solid shown in Figure 1.8, with a through hole extending from the source surface to the target surface. If the intervals along the edges of the through hole are not equal to the intervals along an outer edge path from source to target, then this volume is no longer sweepable. Similar constraints for volumes with non-through holes are also required.

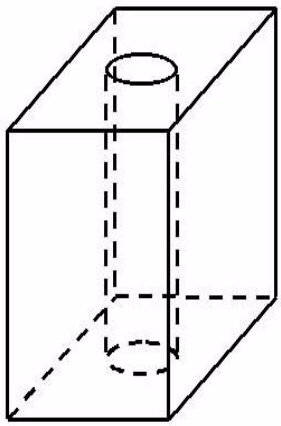
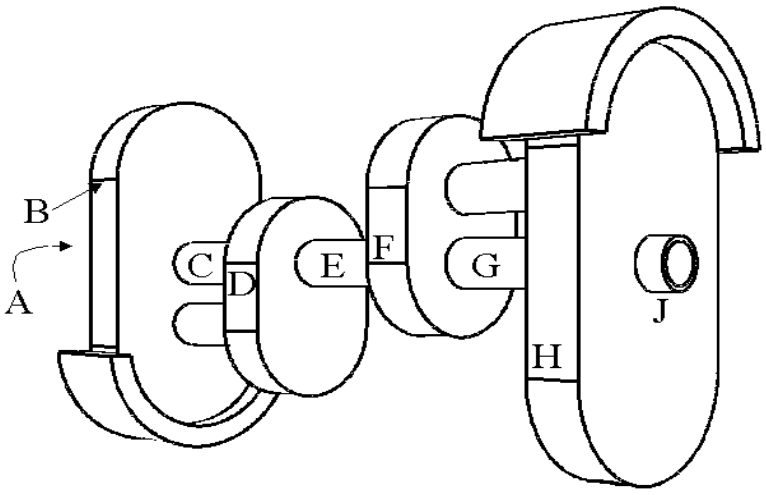


Figure 1.8 - Sweepable Rectangular Tube

As the complexity of the volume geometry increases, the formulation of the volume interval constraints becomes more difficult. The volume shown in Figure 1.9 is an example of a complex sweepable volume. The volume interval constraint that is needed to ensure the sweepability of the volume is also given in Figure 1.9. The formulation of this constraint will be demonstrated in Chapter 4.



$$I_A + I_B + I_C + I_D + I_E + I_F + I_G + I_H + I_J = I_K$$

Figure 1.9 - Many-to-many sweepable volume showing the necessary volume interval constraints to ensure sweepability.
Note: K is an edge along the hole through the volume.
(I_A = the number of intervals on edge A, etc.)

In many cases, the formulation of the volume constraints can become tedious, time-consuming and frustrating. This thesis presents an algorithm which would automate this procedure. The remainder of the material in this thesis will be presented as follows:

- Chapter Two - review of literature in interval assignment algorithms and automatic hexahedral mesh generation via sweeping algorithms

- Chapter Three - presentation of the volume interval assignment algorithm
- Chapter Four - examples of several volumes to which the algorithm has been applied
- Chapter Five - review developments made by this thesis, as well as a proposal for areas of future research.

Chapter 2 - Interval Assignment Background

This chapter presents a review of recent developments in sweeping and interval assignment algorithms used to generate hexahedral meshes in a volume. The focus is on linear programs used for interval assignment, surface interval constraints on sweepable volumes, and enhancements to sweeping algorithms. The specific details of algorithms or enhancements will be based on their implementation in CUBIT[3], a mixed hexahedral and tetrahedral mesh generation tool kit developed by Sandia National Laboratories.

This chapter is broken up into three main sections. First, a brief discussion of linear programs used for interval assignment is provided. Next, a description of interval constraint formulation procedures for mappable and submappable surfaces is given. Finally, developments made to sweeping algorithms to enhance the capability are considered.

Linear Programs and Interval Assignment

A linear program is an optimization problem in which the objective function and design constraints are linear functions of the design variables[17]. For interval assignment, the design constraints are derived from the surface geometry and meshing scheme, and the design variables are the intervals on the curves of the surfaces.

Introduction to Linear Programs

The term linear programming describes a particular class of mathematical extremization problems in which both the objective function and the constraint relations are linear functions[19]. The general form of a linear program can be stated mathematically as: find x = (x1, x2,…,xn)t so as to optimize (either maximize or minimize) the objective function subject to the specified constraints. Each constraint may be a greater-than inequality (<=), a less-than inequality (>=) or an equality (=). Linear programs have the following format:

$$
\begin{aligned}
&\textit{optimize:} && z = c_1x_1 + c_1x_1 + \ldots + c_nx_n \\
&\textit{subject to:} && a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n \ (<=, =, >=)\ b_1 \\
& && a_{2,1}x_1 + a_{2,2}x_2 + \ldots + a_{2,n}x_n \ (<=, =, >=)\ b_2 \\
& && \ldots \qquad\qquad \ldots \qquad\qquad \ldots \\
& && a_{n,1}x_1 + a_{n,2}x_2 + \ldots + a_{n,n}x_n \ (<=, =, >=)\ b_n \\
& && x_1, x_2, \ldots, x_n >= 0
\end{aligned}
$$

A special feature of a linear program results from the fact that any derivatives of the objective function with respect to the design variables are constants which are not necessarily zero. This implies that any extrema of the linear program must be located on the boundary of the design space, and not to the interior of the design space. Because the constraint relations in a linear program are also linear functions, an optimal design must lie at the intersection of two or more constraint functions.

Because large number of variables and constraints are often associated with a linear program, several methods have been designed which will reliably and efficiently move from one extreme point to the next until a solution is found which both optimizes the

objective function and satisfies all of the design constraints. The most popular of these methods are the simplex methods[17].

An integer linear program is a linear program which has the additional constraint of restricting all of the design variables to integer values. Integer programs are combinatorial problems, which are generally more difficult to solve than regular linear programming problems. Cutting-plane and branch and bound algorithms exist for the solution of such problems[9].

Linear Programs and Interval Assignment

The use of linear programming for interval control was first introduced by Tam and Armstrong[12] and later implemented into CUBIT by Mark Whitely[14]. The design variables in the linear program are the actual intervals on the curves of the volumes or sets of volumes. Constraint equations are formulated based on the surface geometry and meshing scheme. The formulation of these constraint equations will be discussed in the next section.

Tam and Armstrong proposed an objective function which minimized the sum of the weighted differences between the goal intervals and the intervals assigned by the linear program[9, 12]. The optimal solution to this linear program was often to change the intervals on very few curves, with the resulting changes often being quite large.

Later work by Scott Mitchell[9] altered the objective function to minimize the lexicographic vector of weighted differences between the goal intervals and the assigned intervals. This is accomplished by adding two extra “delta” variables to each curve which are used to compute the positive and negative value between the assigned interval and the

goal interval. The deltas are weighted inversely proportional to the goal, and an additional variable, M, is used to compute the maximum of the weighted deltas. The algorithm then has two steps.

The first step solves a linear program for the constraints without the integer programming constraints. This results in a solution for the intervals where the solution is not necessarily an integer. The new non-integer interval values are then rounded to the nearest integer value. The second step is to use these nearly feasible values and solve the integer program with all of the design constraints. This is accomplished using a branch and bound technique. The solution is expected to be near the solution found in the first step, so to reduce the amount of time required to complete the procedure, the depth of the branch and bound search is limited.

The resulting interval assignment has a relative change in intervals which is small, rather than the resulting number of curves with interval assignment changes being small. This technique gives interval assignments which have very high fidelity to the goal intervals, spreading out the changes over multiple curves, thus, reducing mesh distortion[9].

Surface Interval Constraints

The constraint equations supplied to the linear program are derived from the meshing algorithms applied to the surface, as well as firmness constraints on edges supplied when the intervals are initially assigned. The constraints derived from the surface meshing algorithms are described in this section.

Surface Mapping Constraints

The surface mapping methods presented in this section are based on standard mapping procedures. In general, these mapping algorithms work well and yield high quality meshes in regions which have roughly parallel opposing sides. The most efficient form of a mapping algorithm does not have a limit on the number of surface boundary curves. The technique is to find four sets of edges on the surface, or the four logical sides, which form a logical quadrilateral for the surface.[14]

To form a logical quadrilateral for the surface, four vertices must be found which form the best corners of the quadrilateral. The four corner vertices are found using a corner picking algorithm which compares the set of interior surface angles and selects the four vertices which are nearest to a perfect right angle (i.e. 90 degrees).[9, 14]
Once the logical corners of the surface are found, the sets of edges between these four vertices form the four logical sides of the quadrilateral. The constraint set for the surface can now be formulated. For a mapping surface, the constraint is that the sum of the intervals on opposite sets of edges are required to be equal, as shown in Figures 2.1 and 2.2.

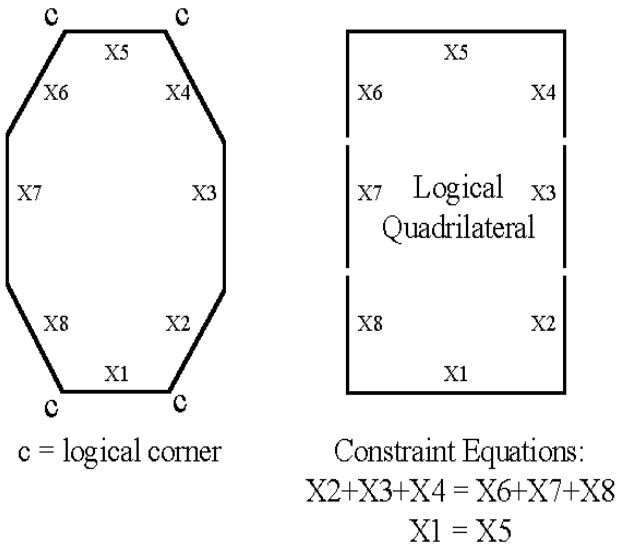


Figure 2.1 - Surface Mapping Interval Constraints

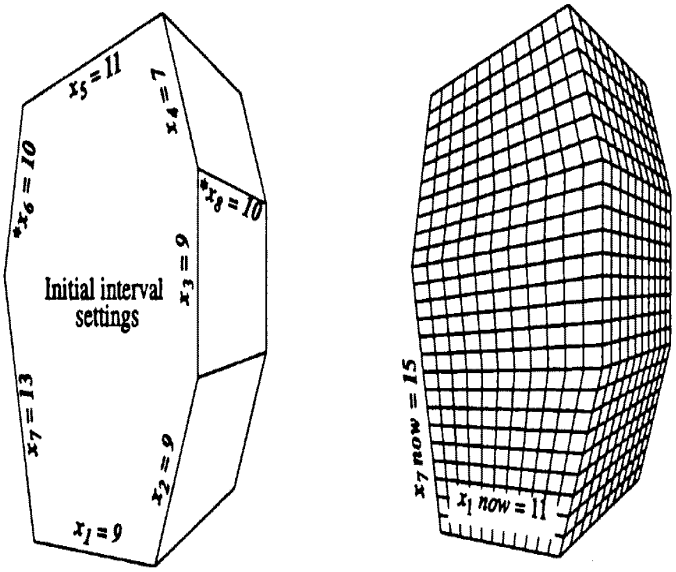


Figure 2.2 - Surface Mapping Interval Assignment and Mesh

Surface Submapping Constraints

A surface which is submappable is similar to a mappable surface, except for the addition of vertices with interior surface angles which are nearly 270 and/or 360 degrees. At these vertex locations, the surface must be decomposed into mappable sub-regions. The resulting mesh is a well-formed grid.[13, 14]

To facilitate interval assignment, a submappable surface can be placed in a local i-j coordinate system. Each edge on the surface is given a classification into a local i-j plane. Starting at an arbitrary vertex on the surface and proceeding around the surface in a counter-clockwise direction, each edge is classified as [+i], [-i], [+j], or [-j].[13, 14] The classification is accomplished by calculating the interior angle between consecutive curves and assigning the proper direction based on the calculated angle. An example of this process is shown in Figure 2.3.

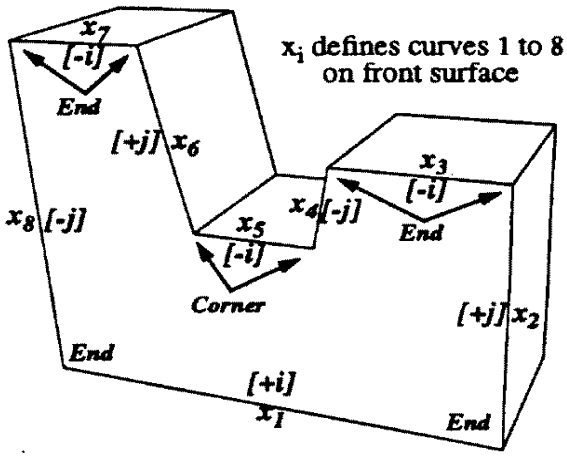


Figure 2.3 - Surface Submapping Vertex Traversal

This method of traversing the surface boundary curves for the purpose of defining their position in the local i-j coordinate system, provides the basis for the formulation of

interval constraints through the use of an “unfolded” surface geometry method.[14] The unfolded surface geometry can be used to formulate the constraint equations for each surface.

The unfolded geometry is formed by taking all curves classified as [+i] and grouping them into one side of the unfolded geometry, and similarly, the opposite side is the collection of [-i] edges. This procedure is followed for the [j] edges, with the resulting unfolded geometry as shown in Figure 2.4.

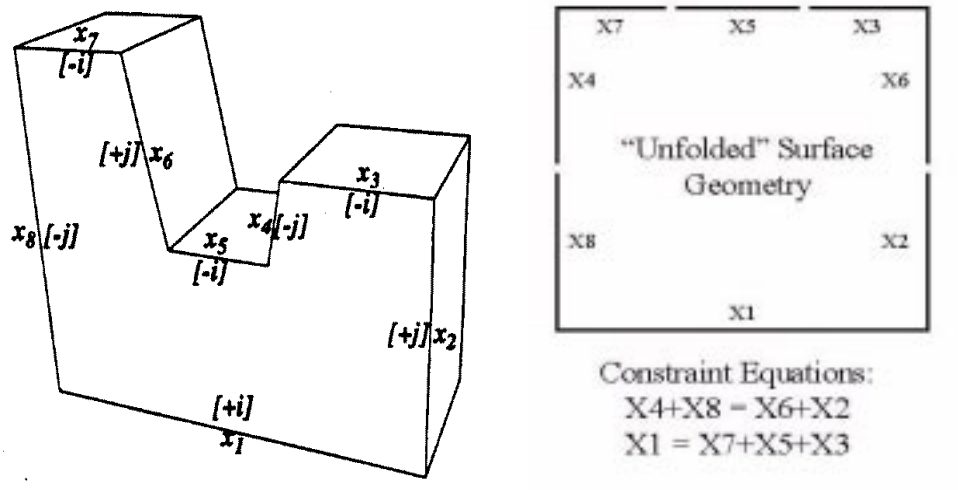


Figure 2.4 - Surface Submapping Edge Parameterization and Constraint Equations Using the “Unfolded” Surface Geometry

The constraints for the submappable surface are then formed in the same way the constraints for a mapped surface were formed. That is, the sum of the intervals on the [+i] edges must be equal to the sum of the intervals on the [-i] edges, and similarly for the [+j] and [-j] edges.

Other Constraints

Other meshing schemes can also supply constraints to the edges on a surface. For example, paving, an advancing-front meshing algorithm, requires that the sum of the intervals on all the curves be even.[9] Although these constraints rarely affect the interval assignment on a sweepable volume, which mainly deals with the mapping and submapping constraints, they must also be considered in the linear program.

The "firmness" of an interval count on each of the edges can also supply constraints to the linear program. Intervals may be specified as being hard (cannot be changed) or soft (can be modified slightly).[9, 14] An interval which is specified as hard supplies the additional equality constraint to the linear program, while soft constraints set goals for an optimal solution.

Sets of surfaces

Mappable and submappable surfaces often have boundary curves which are shared with other surfaces. Due to these shared boundary curves, the interval constraints from one surface are partially propagated to the next surface through the shared boundary curves. An example a series of surfaces is shown in Figure 2.5. Note that if all of the surfaces are assumed to be mappable and the intervals on edge "A" are hard set at a given interval count, then the intervals on edges "B" through "K" are propagated through the surfaces and must be equal to "A".

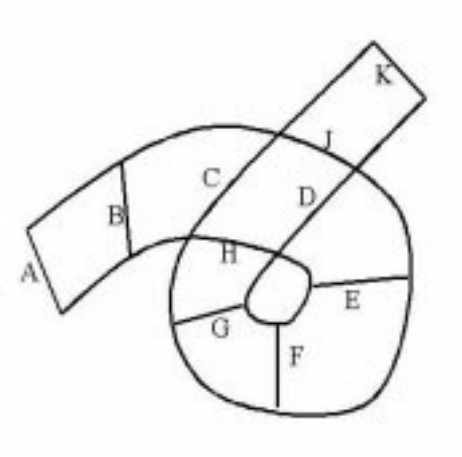


Figure 2.5 - Interval Constraints Imposed by Connecting Surfaces

These interconnected surfaces can be thought of as “chains” of surfaces. The interconnected nature of the interval constraints on these surface chains is important for submappable and sweepable volumes which contain holes. If the intervals on these chains are not constrained together, an infeasible interval assignment may result.

Sweeping Algorithms and Volume Constraints

Sweepable Volumes

A sweepable, or two and one-half dimensional, volume is a volume that has a topologically constant cross section along a single axis.[10] There are several approaches to mesh generation via sweeping, but common to all is the idea of identifying surfaces on a volume to serve as “sources” and “targets”, and a complementary set to serve as “linking sides”. The source surface(s) is meshed, and then swept along the linking sides towards a target surface. This is feasible provided the linking surfaces are meshed with a mapping or submapping algorithm.[7]

Sweepable volumes are often classified by the number of source surfaces and the number of target surfaces. A volume with one source surface and one target surface is known as a single source to single target sweep. Volumes with more than one source and only one target are known as many-to-one sweeps, and volumes with many sources and many targets are known as many-to-many sweeps.

Until recently, many-to-many sweeps were not possible without decomposing the volume first. Recent developments have enabled this class of volumes to be swept automatically[2, 8]. This is accomplished through a series of source and target surface projections and subsequent surface imprinting through sweep layers. Each of the sweep layers is represented by a single interval along the linking surfaces. It is crucial that the interval assignment on the linking surfaces be correct in order to guarantee proper imprinting on the correct surface and the eventual success of the meshing scheme. Due to the inherent complexity of the many-to-many type volumes, interval assignment can be a tedious and time-consuming process, even with the aid of surface constraints and a linear program. To ease this process, volume interval constraints need to be added to the linear program.

Volume Interval Constraints

The linking surfaces on a sweepable volume are often interconnected to each other through shared edges. Sets of surfaces connected through shared edges can be thought of as "chains" of surfaces. Interval assignment and constraints are propagated around a chain of linking surfaces through the interconnected edges. When all of the interval constraints are met for each chain of linking surfaces, this chain can be guaranteed to be sweepable, with respect to interval assignment.

A problem arises, however, when more than one chain of linking surfaces is connected to a source or target surface on a volume. These cases are normally found on volumes with holes. Because there are no shared curves connecting the surface chains within the hole to the surface chains on the exterior of the volume, the interval constraints are not propagated among the chains. The result is that many times the intervals within the hole do not match the intervals on the exterior of the volume. This improper interval assignment eventually leads to the failure of the sweeping algorithm, and, subsequently, a time intensive process to remedy the interval assignment problem. It is this problem that this thesis is to address.

The remedy to this problem is to formulate constraint equations which couple the interval constraints between the surface chains to each other. These new "volume interval constraints" can be supplied to the linear program in addition to the surface interval constraints. The surface and volume constraints can then be solved simultaneously in the linear program. The remainder of this thesis focuses on a new algorithm, implemented in CUBIT, which formulates and supplies these additional volume interval constraints.

Chapter 3 - Volume Interval Assignment

In this chapter the volume interval assignment algorithm for sweepable volumes will be presented. This algorithm automates the process of assigning intervals for holes in a sweepable volume.

Before describing the volume interval assignment process, a brief description of graphs will be given by way of introduction to the algorithm. Following these descriptions, the general algorithm will be described with detail into each step of the process. The description of the algorithm will be based on its implementation in CUBIT[3], a mixed hexahedral and tetrahedral mesh generation tool kit developed by Sandia National Laboratories.

Introduction to Graphing Algorithms and Definitions

The volume interval assignment routine uses a graphing algorithm to identify paths of edges from a source surface to a target surface on a sweepable volume. The goal is to find at least one edge path per linking surface chain and constrain the sum of the intervals on each independent chain's edge path to be equal. This effectively couples the interval assignment on the two chains together. A brief introduction to graphing algorithms and the terms used in conjunction with this research will be given in this section.

What is a graph?

A graph consists of a set of objects called vertices and another set known as edges, such that each edge is identified with an unordered pair of vertices.[6] The most common representation of a graph is by means of a diagram, in which the vertices are represented as points and each edge as a line segment connecting it's corresponding vertices. Figures 3.1 and 3.2 are examples of graphs.

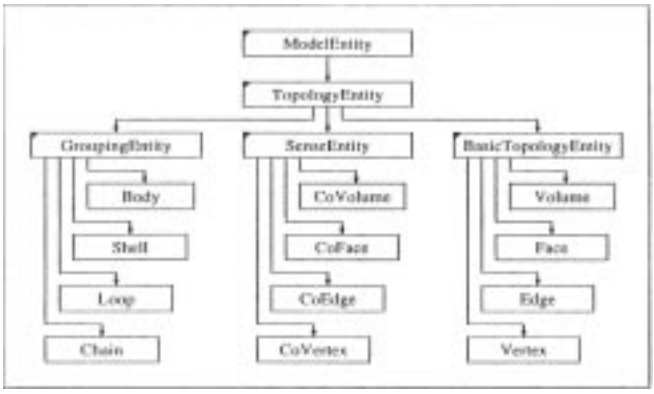


Figure 3.1 & 3.2 - Example Graphs

Because of a graph's inherent simplicity, graph theory has a very wide range of applications in engineering; physical, social, and biological sciences; in linguistics; and in numerous other areas. A graph can be used to represent almost any physical situation involving discrete objects and a relationship among them.

By supplying additional data or constraints to the vertices and/or edges in the graph, the graph can be used to solve flow problems for city water systems, electrical circuits, shortest path problems, etc. In addition, the graph structure is ideal for some data storage and data searching algorithms, lending itself nicely to many applications associated with computers and computer programming.

In this work, the vertices in the graph are known as "sweep vertices." The edges in the graph correspond to either a curve or a periodic surface on the volume. A collection of sweep vertices is known as a "super vertex."

The Volume Interval Assignment Algorithm

The volume interval assignment routine begins with an arbitrary sweepable volume on which the source surface(s) and target surface(s) have been designated. The goal of the routine is to detect instances of independent and parallel sets of linking surface chains and supply interval constraints to couple the interval assignment on the surface chains.

Initial Graph Creation

To accomplish the goal of the algorithm, a volume must be designated as sweepable with the source and target surfaces given. A graph can then be created from the volume geometry. Each vertex on the volume becomes a sweep vertex in the newly

created graph. Edges in the graph are represented by the curves or periodic surfaces between the vertices on the volume. An example of a sweepable volume is shown in Figure 3.3 with source surfaces surfaces S1 and S2 and the target surface T1.

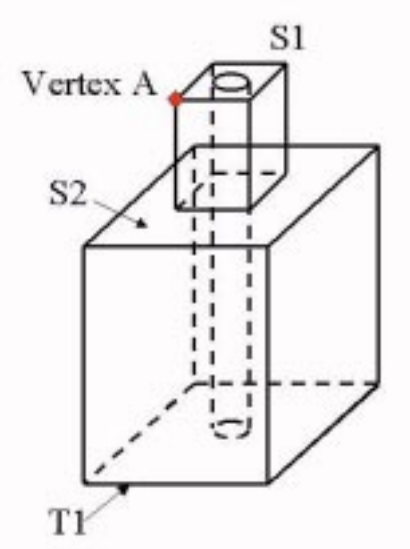


Figure 3.3 - Example Sweepable Volume

As can be seen in Figure 3.3, the connectivity of the vertices through the volume's curves is not always enough to determine an edge path from a source surface to a target surface. This is seen readily at vertex "A" in which there is not an edge path from the vertex to the target surface. In other words, all edge paths from vertex "A" loop back to source surface S1 and vertex "A" without ever encountering a vertex on a target surface.

Final Graph Creation

To alleviate this problem, a collapse of the source and target surfaces takes place, where each of the vertices on a source or target surface is coalesced into one vertex. This collapse creates a new vertex type in the graph called a "super vertex". A super vertex is a

collection of sweep vertices. A super vertex contains the connectivity knowledge of each individual sweep vertex on the source/target surface, thereby expanding the search capacity of the graph across the surface. This collapse of the source and target surfaces, and formation of the super vertices is shown in Figure 3.4.

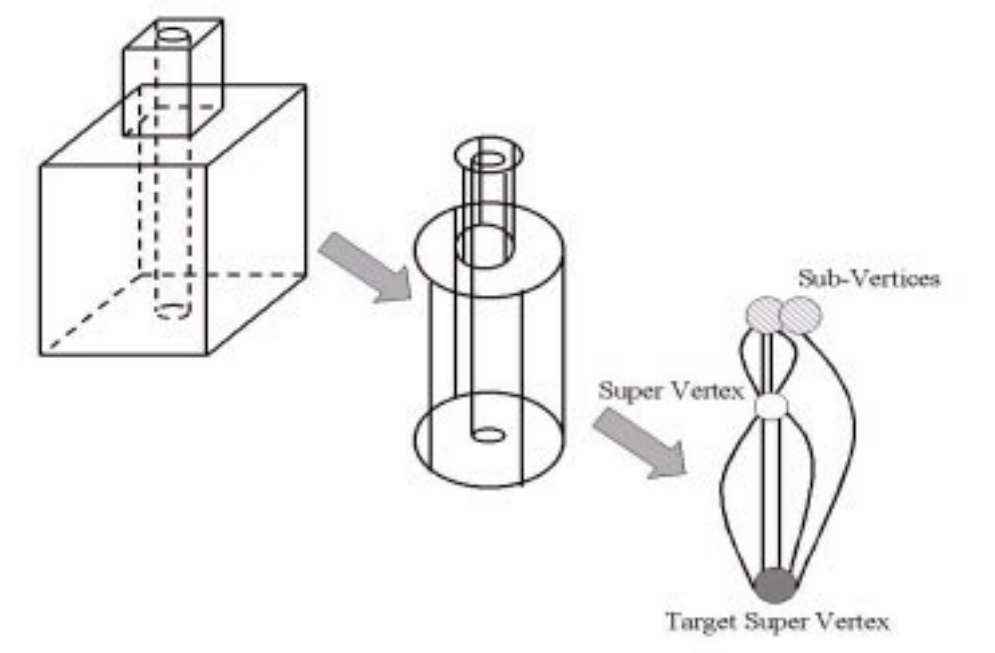


Figure 3.4 - Source and Target Collapse to Form Super Vertices

Searching the Graph

Once the initial and final graph setups have been completed, the parallel and independent surface chains can be searched to find edge paths from a source surface to a target surface. Each edge within an each edge path will later be assigned a weight to be used in formulating the constraint equations.

Where to Search

To reduce the total number of constraint equations written to the interval linear program, only source surfaces with independent and parallel sets of the linking surface chains need to be searched. It is necessary to find the instances of the independent, parallel linking surface chains, and find one representative edge path from the source surface to the target surface per chain.

An assumption has been made with regards to the multiple chain detection. The assumption is that each instance of multiple linking surface chains attached to a source surface or set of source surfaces corresponds to multiple edge loops on the source surface(s), as shown in Figure 3.5. The edge loops are found by first collating the source or target surfaces with shared edges. If the remaining edges on the surface(s) form more than one edge loop, then this surface becomes a candidate for further searching and volume interval constraint formulation.

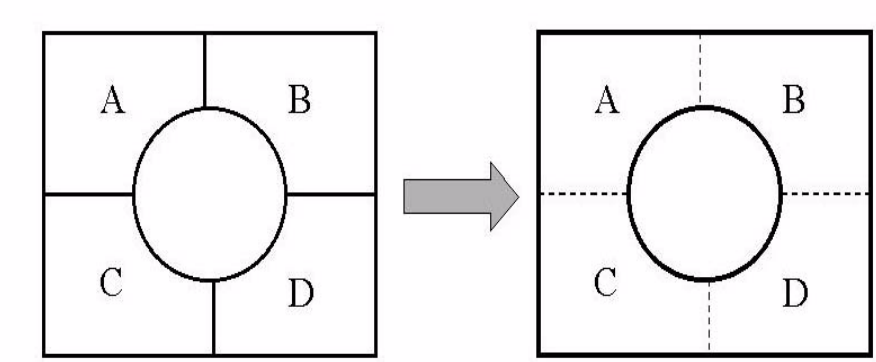


Figure 3.5 - Combining Edges on Surfaces A, B, C, & D
to Form Two Edge Loops

From each loop a search will be initiated that will lead to a representative path of edges from the source surface(s) to the target surface. Each loop on the source surface(s) can be thought of as a sub-vertex within the collapsed surface(s) super-vertex. Searches will normally be initiated from these sub-vertices.

The Breadth-First Search

The graph searching algorithm used in the volume interval assignment algorithm is a variant of a breadth-first search. The breadth-first search algorithm was chosen for two reasons. First, the shortest path between the start vertex and any other vertex within the graph is always returned[5]. This is advantageous because the shortest path represents the fewest edges which must be constrained against each other. This results in an optimal linear program which can have added benefits in speed for larger models. Second, the breadth-first search generates a breadth-first tree which can be used to translate an ordered list of vertices to edges to be used in formulating the constraint equations.

The search algorithm works by taking a starting vertex and systematically exploring each of the edges to "discover" every vertex that is reachable from the starting vertex. The shortest distance, or fewest number of edges, from the starting vertex to all reachable vertices is also computed. The breadth first search is so named because it expands the frontier between discovered and undiscovered vertices uniformly across the breadth of the frontier. That is, the search discovers all vertices at distance (k) from the starting vertex before discovering any vertices at a distance (k+1). [5]

To keep track of progress, a color is given to every vertex indicate the state of the vertex in the search. The three states in the search process correspond to the three colors:

white, gray, or black. All vertices start out white and later, and as they are searched, become gray, then black. A vertex is discovered the first time it is encountered during the search, at which time it becomes gray, then black. Gray and black vertices, therefore, have been discovered. As the graph search progresses, the white vertices represent the frontier of undiscovered vertices. [5]

The breadth first search constructs a breadth-first tree, initially containing only the starting vertex. Whenever a white vertex is discovered in the list of connected vertices, the white vertex is added to the tree, and the color is changed to gray. Because only white vertices are added to the tree, a vertex can be discovered at most once, and therefore each vertex, or descendant, in the tree can have only one parent relative to the starting vertex in the tree.

The search progresses level by level by adding white vertices to the tree below the gray vertex being searched. Once the desired final vertex is found, the search algorithm is stopped and the resulting tree can be searched from the final vertex to the start vertex using the parent-descendant relationship within the tree. An example of a graph search is shown step-by-step in Figure 3.6.

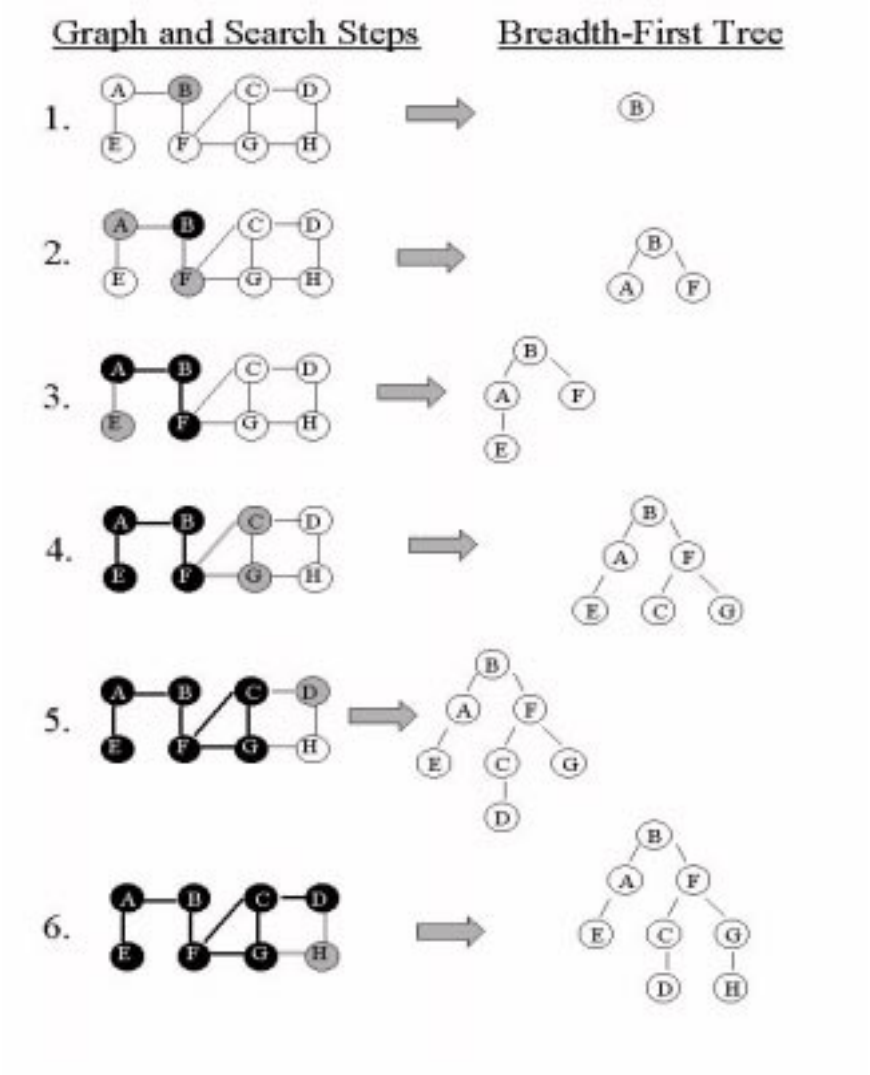


Figure 3.6 - The Breadth-First Search

Search Completion

For a sweepable volume, a search begins at a sub-vertex found on a source surface with more than one edge loop, as described earlier. Prior to beginning the search, each vertex within the super-vertex has the color set to black, with the exception of the sub-vertex. This ensures that the search cannot loop back on itself. This step also ensures a set of unique edge paths from the source surface to the target surface, that is no two edge

paths from differing sub-vertices on the source surface will contain exactly the same edges.

The search continues in the breadth-first fashion until the first target super-vertex, or a specified target super-vertex, is encountered. At this point, the search ends successfully, and an ordered list of vertices is obtained from the breadth-first tree. This ordered list of vertices can then be translated to a set of edges between each of the vertices in the list. This set of edges represents an edge path from the sub-vertex to the target surface.

Blind Holes

A blind hole in a volume is a hole that does not pass completely through the volume. A graph search of a blind hole creates an interesting result. Because the super-vertex is blacked out prior to the search, and because the search is completed successfully only when the target super-vertex is found, a search within a blind hole can never be completed. Using this knowledge, blind holes can be detected using the graphing algorithm.

The constraint equation for a blind hole edge path is to set the intervals within the blind hole to be less than the intervals without the blind hole. The search is allowed to continue until the blind hole's bottom is found. The search is terminated at this point and the breadth-first tree is traversed to obtain the ordered vertex list within the blind hole. This vertex list is flagged as belonging to a blind hole rather than a through hole, and the corresponding constraint equation can be setup with a less-than inequality (<=), rather than a equality (=). An example of a volume containing a blind hole and the corresponding graph is shown in figure 3.7.

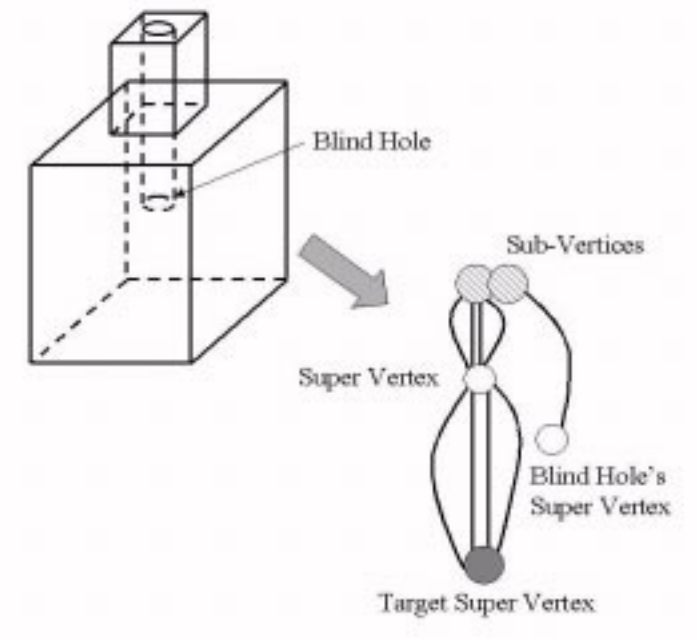


Figure 3.7 - Graph with a Blind hole

Vertex to Curve Translation

The ordered vertex list must be translated to a set of edges representing an edge path from the source surface to the target surface. Due to the collapse of the source and target surfaces to form the super vertices, each set of consecutive vertices in the ordered vertex list does not necessarily share a corresponding edge. Therefore, the translation also consists of finding edges from a vertex to a source/target surface and from a source/target surface to a vertex.

Constraint Formulation

Once the ordered list vertices has been translated to edges, the constraint formulation for each edge path can begin. This is accomplished by attaching a weight, termed a sweep weight, to each of the edges in the list.

The weight associated with each edge is found by systematically traversing the edge list and determining the direction of traversal for the edge. An edge which is being traversed from a source towards a target is given a weight of one, while an edge being traversed from a target to a source is given a weight of -1. An edge which is found to be parallel to the source or target surface is given a weight of zero.

The direction of traversal is found by using the previous edge and weight in the edge list. If the edge is the first edge in the list, the angle is calculated between the source surface's normal vector and the vector represented by the edge. If the calculated angle is approximately 180 degrees, the edge is being traversed from the source to the target, and is assigned an edge weight of one.

The direction of traversal for the edges following the first edge in the list can be found by using the previous edges direction of traversal and comparing it to the current edges direction of traversal. If they are found to be in approximately the same direction, the previous edge's sweep weight is also assigned to the current edge's sweep weight. Variations in the direction of traversal from the previous edge to the current edge changes the weight accordingly to either a zero or a negative one, depending on the calculated angle between the two edges.

Once all edges in the edge list have a weight assigned to them, the constraint equations can be formulated. The sum of the products of the weight and the edge's interval count for each path correspond to one side of a constraint equation. Two corresponding edge paths from the same source surface are constrained as being "equal to" for a through hole or "less than" for a blind hole. Figure 3.8 shows an example of the constraint equation formulated for the example sweepable volume.

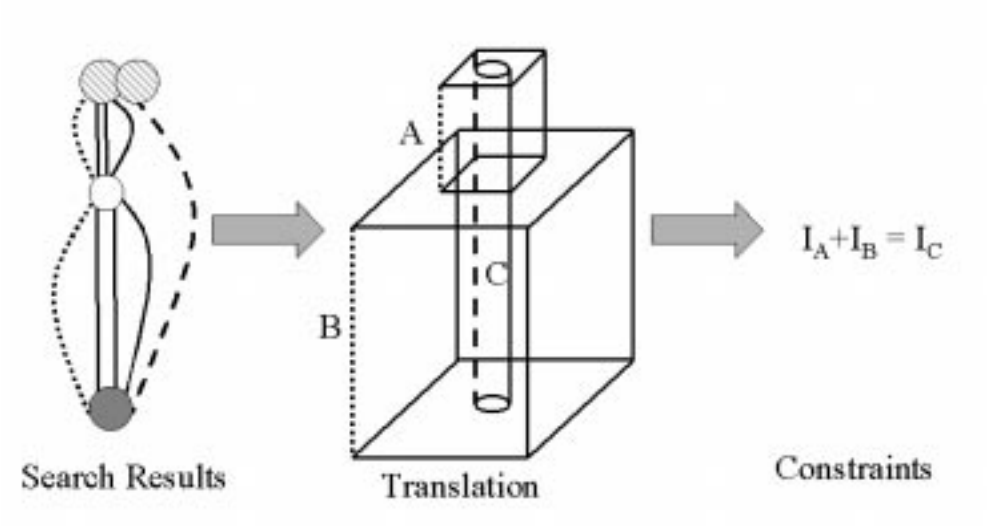
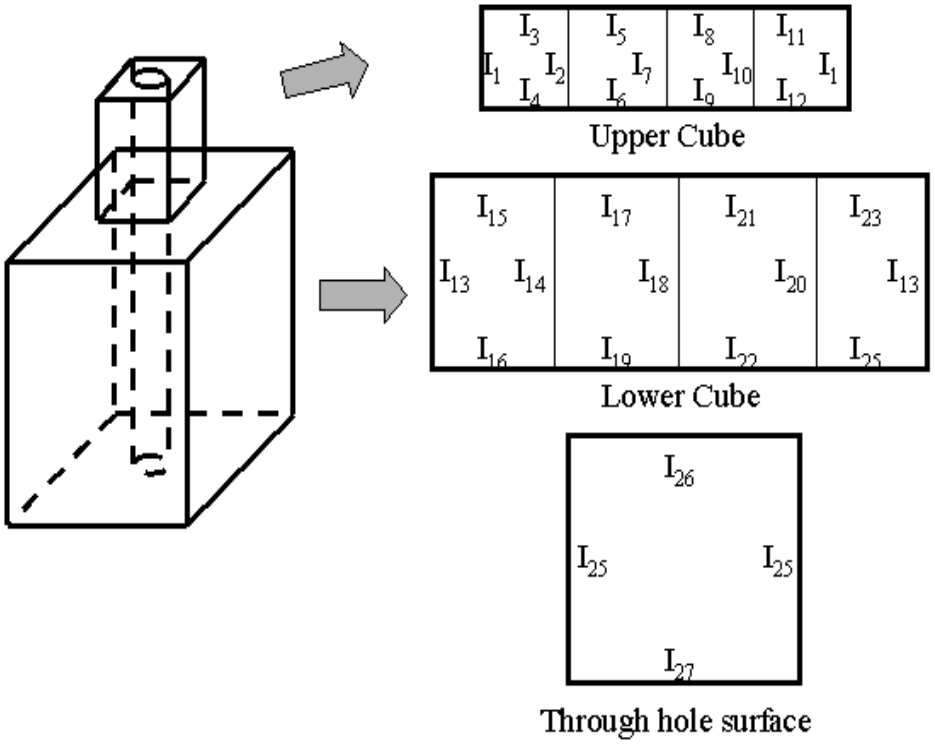


Figure 3.8 - Search, Translation and Volume Interval Constraint Equations

Interval Assignment

The formulated constraint equations are supplied to the same interval linear program which was developed for the surface interval assignment. Using the same linear program for the surfaces and volumes, the interval assignment on each edge can be optimized for all constraints on the volume, not just the surfaces.[9, 14]

The constraint equations are stored in the linear program as a matrix. The matrix representation of these equations is advantageous because matrix operations can be performed on the constraint equations to simplify the optimization of the intervals. The volume interval constraint equations have a Gaussian elimination step which is performed to simplify the constraint equations and reduce any ambiguity that may occur in the process of forming the constraints.



Minimize: $Z = D_1 + D_2 + D_3 + D_4 + D_5 + D_6 + D_7 + D_8 + D_9 + D_{10} + D_{11} + D_{12} + D_{13} + D_{14} + D_{15} + D_{16} + D_{17} + D_{18} + D_{19} + D_{20} + D_{21} + D_{22} + D_{23} + D_{24} + D_{25} + D_{26} + D_{27}$

Subject to: (Minimum Edge Constraints)

$I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9, I_{10}, I_{11}, I_{12}, I_{13}, I_{14}, I_{15}, I_{16}, I_{17}, I_{18}, I_{19}, I_{20}, I_{21}, I_{22}, I_{23}, I_{24}, I_{25}, I_{26}, I_{27} >= 1$

(Linking Sides Surface Constraints)

$I_{13} = I_{14} = I_{18} = I_{20}$

$I_1 = I_2 = I_7 = I_{10}$

$I_{26} = I_{27}$

$I_{15} = I_{16}, I_{17} = I_{19}, I_{21} = I_{22}, I_{23} = I_{25}$

$I_3 = I_4, I_5 = I_6, I_8 = I_9, I_{11} = I_{12}$

(Additional Volume Constraint)

$I_{26} = I_1 + I_{13}$

Where: D_n = the difference between the interval initially placed on curve n and the interval assigned to curve n by the linear program

I_n = the number of intervals on curve n

Figure 3.9 - Linear Program

Once the Gaussian elimination procedure has distilled the set of volume constraints as much as possible, the constraints are added to the system constraint matrix. The linear program for the example volume in this chapter is shown in Figure 3.9. Figure 3.9 also displays the volume as an unfolded set of surfaces to act as a map for interpreting the linear program. The volume constraints are added to the linear program enabling the surface and volume constraints to be solved simultaneously.

Once the linear program has been solved, an integer linear program must also be solved to ensure that all of the system constraints are met and that all intervals are integer numbers, as discussed in Chapter 2.

Chapter 4 - Example Problems

This chapter presents some examples of volumes and the results produced from the volume interval matching algorithm. The examples used for demonstration in this chapter have been chosen for their ease of showing the steps of the algorithm.
Each demonstration will progress through each step of the algorithm describing the items taking place at each step. The steps of the algorithm are:

1. Initial graph formation
2. Final graph formation
3. Searching the Graph
4. Vertex to curve translation
5. Edge weight assignment
6. Constraint equation formulation and addition of the constraint equations into the linear program.

Other examples of meshed geometries to which the algorithm has been applied will be shown at the end of the chapter.

Many-to-One Example with Blind and Through Holes

The first example is a many-to-one sweep with a through hole and a blind hole. The geometry is shown in Figure 4.1.

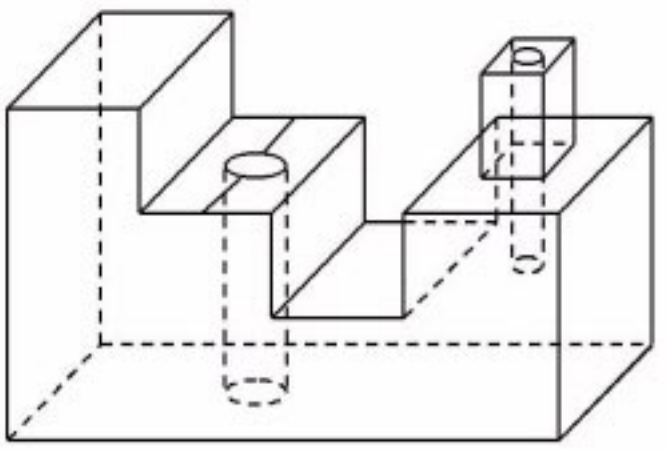


Figure 4.1 - Many-to-One Example Volume

Initial Graph Creation

The initial graph for this geometry is created from the vertices of the geometry. A sweep vertex is formed for each vertex on the volume. Each sweep vertex is given knowledge of any vertex to which it is connected through an edge in the geometry. Each sweep vertex can be accessed through the geometry vertex. The connected vertices are stored in a list of vertices on each of the sweep vertices.

Vertices which are connected through periodic surfaces must also have knowledge of each other. This is accomplished by simply adding these vertices to the vertex list stored in the sweep vertex.

With the connectivity data stored on each sweep vertex, some geometry searching can be accomplished. However, due to incomplete connectivity across surfaces, not all searches from a vertex on a source surface would eventually reach a target surface. Therefore, a collapse of the source and target surfaces is affected to complete the necessary connectivity.

Final Graph Creation

A new data structure is created by the collapse of the source and target surfaces known as a super vertex. The super vertex retains the connectivity knowledge of each of the sweep vertices within the super vertex. This is important due to the differentiation which must take place when a surface is to be searched.

Just prior to collapse, a query of the source and target surfaces is initiated to determine if any of the source or target surfaces share curves with any of the other source or target surfaces. If any of these surfaces share edges, then the two super vertices for each of these surfaces must be combined.

To limit the number of searches which are required to take place, the number of edge loops for each surface or set of surfaces is determined. The assumption is that only surface sets with multiple edge loops will have multiple chains of linking surfaces attached to them. For each edge loop on the surface(s), a sub-vertex will be formed within the super vertex. A sub-vertex is the base of each breadth-first search tree during the subsequent searching step. The collapse, along with the formation of the super vertices and sub-vertices, is shown in Figure 4.2 for the example geometry.

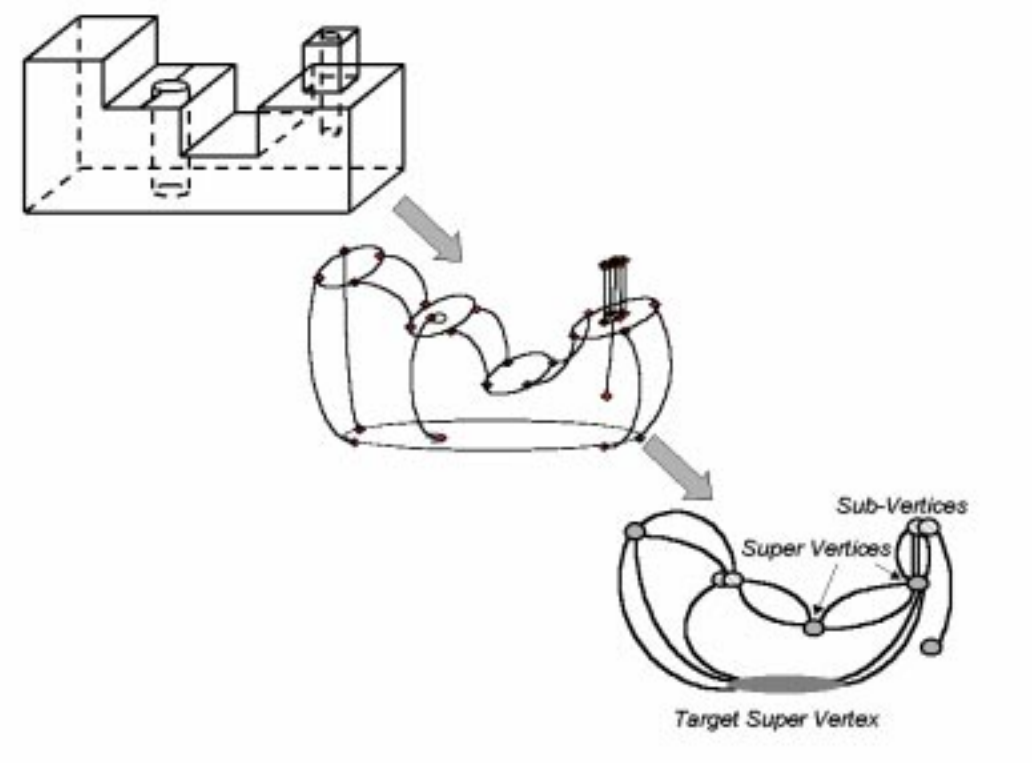


Figure 4.2 - Source and Target Surface Collapse to Form Super Vertices and the Final Graph

Graph Search

With the final graph formed, the next step is to search from each sub-vertex to find the shortest list of vertices from a sub-vertex to the target super-vertex. A breadth-first search is used to accomplish this step. To prevent identical paths of curves from being returned by the search, all of the sub-vertices on the super vertex, with the exception of the sub-vertex being searched, are blacked out prior to starting the search. This guarantees a unique edge path per sub-vertex, and also ensures that each chain of linking surfaces will be represented by an edge path in the constraint equations which are formulated later.

Blind holes present an interesting situation in the search. A successful search is completed upon encountering the target super-vertex, but because all of the sub-vertices

have been blacked out prior to searching, any search within a blind hole results in failure of the search algorithm. The subsequent failure is used to detect the blind holes and the breadth-first tree is used to find a partial path of curves representing the blind hole. The partial path is flagged as being a blind hole and the constraint equations are set up accordingly. The search results for the example volume are shown in Figure 4.3, where each set of dashed lines represents an independent and parallel edge path from the sub-vertex to the target super vertex.

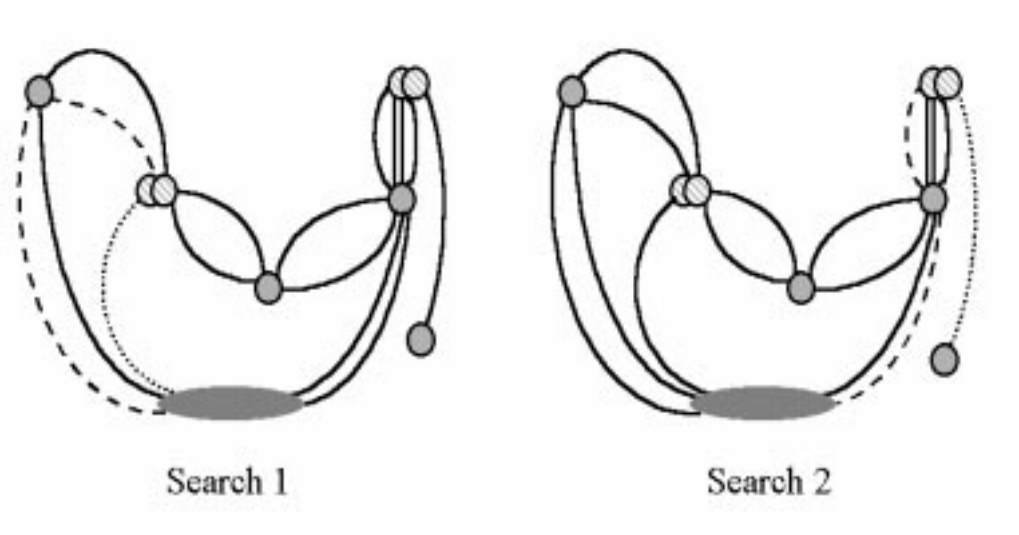


Figure 4.3 - Search Results for Two Sets of Sub-Vertices

Vertex-to-Curve Translation

An ordered list of vertices from the sub-vertex to the target super vertex results from the search procedure. This ordered list of vertices must be translated to a list of corresponding curves representing an edge path for each chain of linking surfaces attached to the source surface(s). The translation process requires finding the corresponding curve or periodic surface between two vertices in the ordered list. The vertices which are

returned in the ordered list are the geometry vertices, not the sweep or super vertices. Therefore, the translation algorithm is required to find a corresponding curve between two geometry vertices, a geometry vertex and a super vertex, or between two super vertices. The original volume with the translated edges is shown in Figure 4.4.

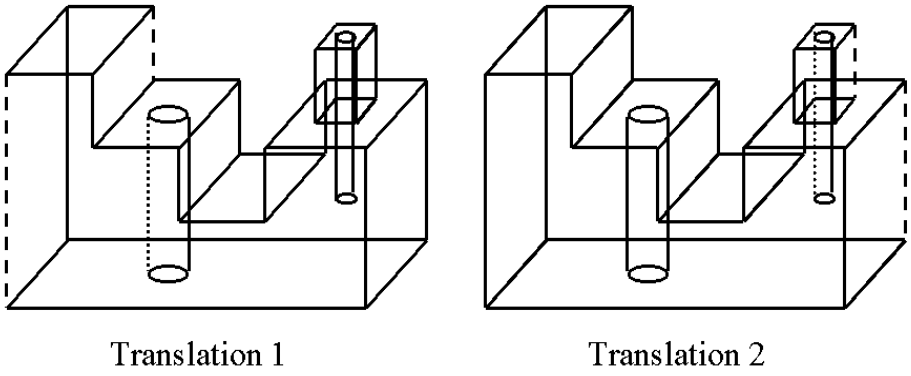


Figure 4.4 - Search Results Translated Back to Original Volume

Edge Weight Assignment

A weight will be multiplied by the curves interval count and the sum of the product of all the weights and interval counts for all of the curves in the translated list corresponds to one side of a constraint equation.

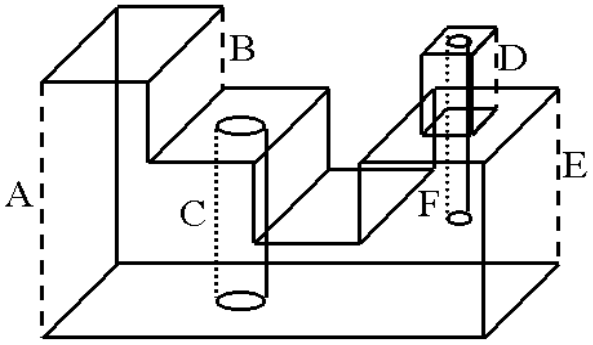
The weight on each edge is either +/-1 or zero corresponding to the sweep direction for the volume. A curve which is being traversed in a direction from the source to a target is given a weight of one. A curve being traversed opposite the sweep direction is assigned a weight of negative one, and perpendicular to the sweep direction is given a weight of zero.

The edge weight and sweep direction is determined from a local angle calculation between a given curve in the curve path list and the previous curve in the same list. If the curve is the first curve in the list, the weight is determined by calculating the interior angle between the first curve and the source surface normal at the curves location on the source surface. The surface normal or previous curve sweep weight is always known and the sweep weights can be found by simply stepping through the list in order until the entire list of curves has been assigned weight.

Constraint Formulation and the Linear Program

Each super vertex with sub-vertices will contain a set of corresponding constraint equations which needs to be added to the linear program. Every path from the super vertex must be constrained against every other path on the super vertex. This is accomplished by summing the product of the weight and the interval count on each edge within a path, and then either setting two paths equal to each other for a through hole, or setting one side less than the other for blind holes. Every path is constrained against at least one other path on the super vertex.

The newly formulated constraint equations are added to the linear program's constraint matrix through a specially written interface for volume interval assignment. Once all constraints have been written to the constraint matrix, the constraints are distilled through a Gaussian elimination procedure. This step reduces the overall size of the constraint matrix, thereby speeding the solution of the linear and the integer linear programs. The example volume with the final set of volume interval constraints is shown in Figure 4.5.



Constraint Equations:

$I_C = I_A - I_B$

$I_F <= I_E + I_D$

Figure 4.5 - Original Volume with Volume Interval Constraint Equation
(I_A = number of intervals on curve A, etc.)

Many-to-Many Sweep Example

The second example in this chapter demonstrates the capability of the algorithm to handle all types of sweepable volumes. The example shown in this section is a many-to-many sweepable volume, and the procedure is shown in a step-by-step manner. Figure 4.6 is the example volume used in this section.

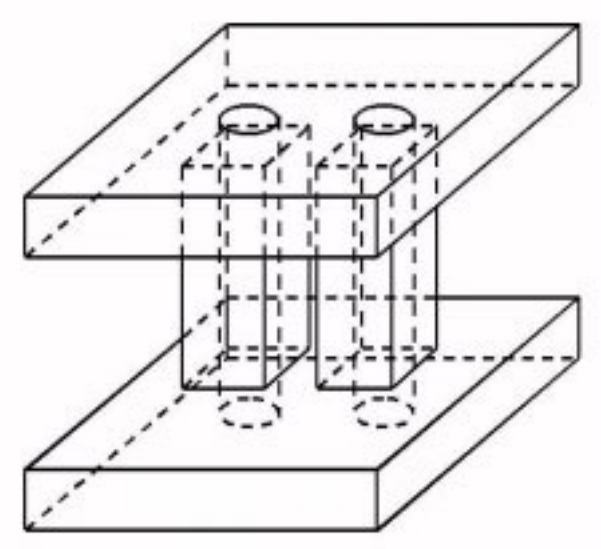


Figure 4.6 - Many-to-Many Example Volume

Initial and Final Graph Creation

Graph creation for the many-to-many example is the same as the previous example. All vertices are first assigned a sweep vertex and connectivity information is then transferred to the super vertex during the collapse stage.

Once all sweep vertices have been defined and set, the source and target surfaces are combined, as necessary, and any instances of multiple edge loops are found. For each edge loop, a sub-vertex is formed, and for each source/target surface or set of surfaces, a super vertex is formed by the collapse of the sweep vertices on the surface(s). This process is shown in Figure 4.7.

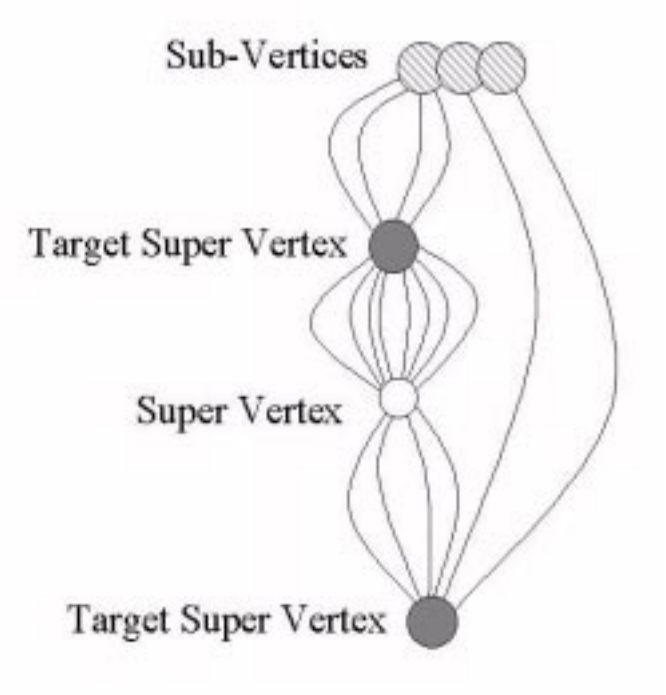


Figure 4.7 - Final Graph

Graph Search

The graph search procedure is altered slightly for the many-to-many case. The corresponding curve paths from each sub-vertex must correlate by having similar ending target super vertices in order for the constraint equations to be correct. Therefore, a knowledge of the target super vertex at which the search is terminated is added to the source super vertex. All searches which begin at the given source super vertex must end at the same target super vertex so that the paths can be related in the constraint equations. The search progresses in the same manner as for the many-to-one example, except that the search is completed successfully upon reaching the specified target super vertex. Each super vertex may then have a different target super vertex as the final destination of the search. Figure 4.8 shows the result of a search from each of the super vertices with sub-vertices.

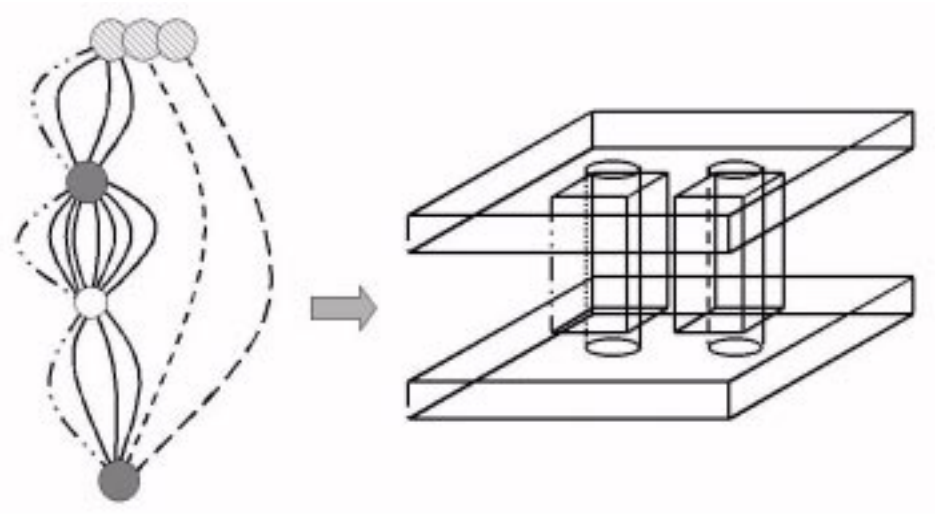


Figure 4.8 - Graph Search Results

Vertex-to-Curve Translation and Edge Weight Determination

Vertex-to-curve translation and edge weight determination is similar for both the many-to-one and the many-to-many type sweeps.

Constraint Formulation and the Linear Program

The constraints formed in this step are also similar to the many-to-one type sweep case. The final volume interval constraints for the example volume are shown in Figure 4.9.

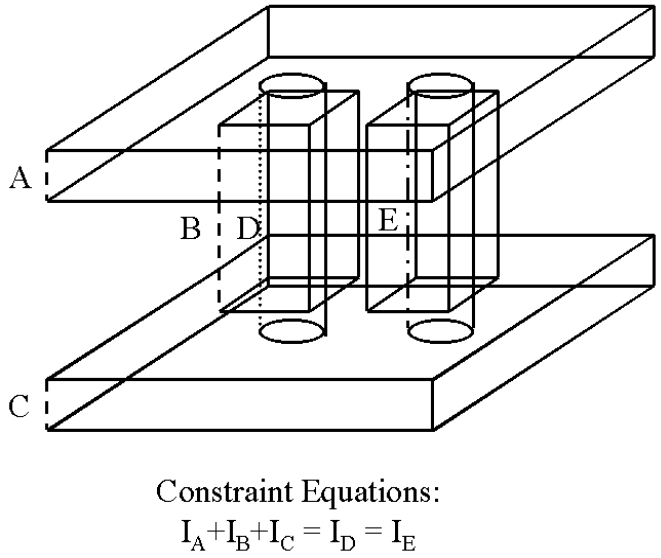


Figure 4.9 - Volume with Volume Interval Constraint Equations

Example Cam Shaft

The final example in this chapter demonstrates the capability of the algorithm to handle fairly complex types of sweepable volumes. The example shown in this section is a many-to-many sweepable volume. The procedure for generating the volume interval constraints will be shown in a step-by-step manner. These constraints will be added to a linear program in addition to the constraints imposed by the surfaces' geometry and meshing schemes. Figure 4.10 is the example volume used in this section.

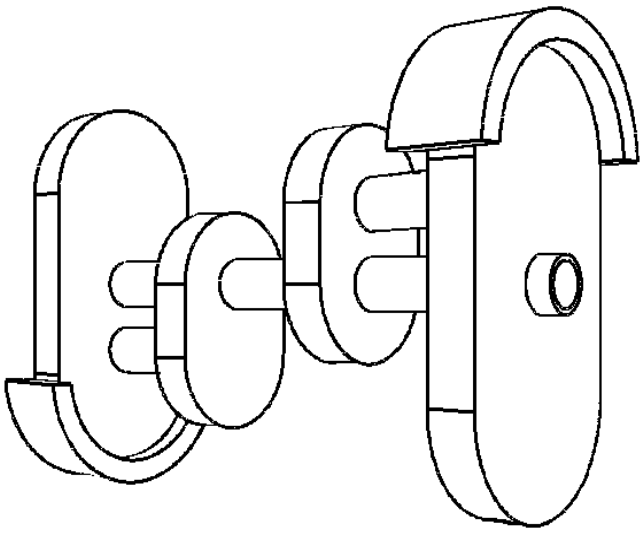


Figure 4.10 - Cam Shaft

Initial and Final Graph Creation

Graph creation for the cam shaft is the same as the previous example. All vertices are first assigned a sweep vertex and connectivity information is transferred to the sweep vertex. At this stage, the super vertices can be formed by collapsing the source and target surfaces. The final graph for this example is shown in Figure 4.11.

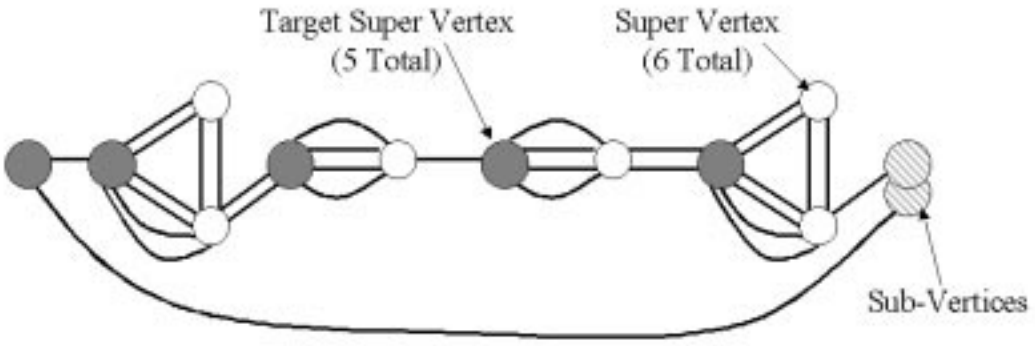


Figure 4.11 - Final Graph for the Cam Shaft

Graph Search

The graph search procedure is the same as for the previous many-to-many case. All search paths beginning at a given source super vertex must end at the same target super vertex so that the paths can be related in the constraint equations. The search progresses in the same manner as for the many-to-one example, except that the search is completed successfully upon reaching the specified target super vertex. Each super vertex may then have a different target super vertex as the final destination of the search. Figure 4.12 shows the result of the search from the sub-vertex for the cam shaft example.

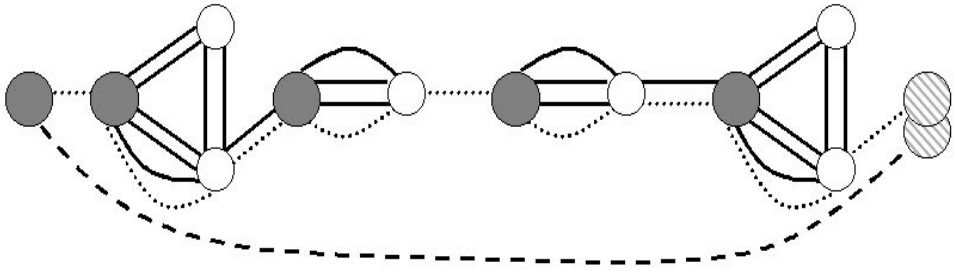
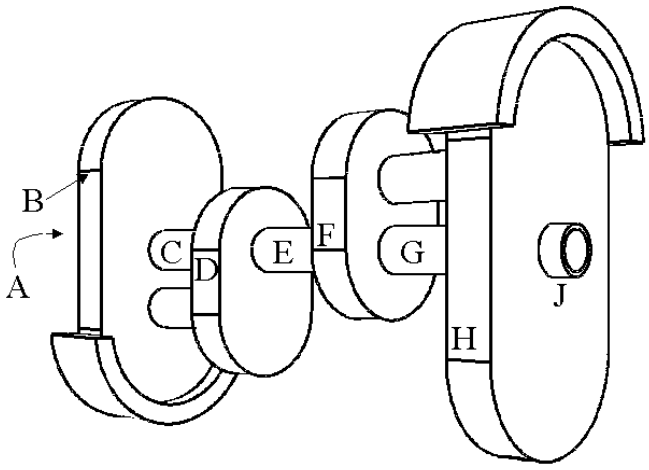


Figure 4.12 - Graph Search Results

Constraint Formulation and the Linear Program

The constraints are formed after translation of the search results and edge weights are determined. The constraint is the sum of the edge weight multiplied by the interval count on each edge for each path from the source to the target. The final volume interval constraints for the example volume are shown in Figure 4.13. These constraints will be supplied to the linear program in addition to the surface constraints. The addition of the volume constraints effectively couples the surface constraints for the through hole to the outer surfaces of the volume.



$$I_A + I_B + I_C + I_D + I_E + I_F + I_G + I_H + I_J = I_K$$

Figure 4.13 - Volume with Volume Interval Constraint Equations
Note: K is an edge extending through the interior hole.
(I_n = the number of intervals on edge n.)

Other Examples

Figures 4.14 through 4.17 display examples of meshed volumes to which this algorithm has been applied.

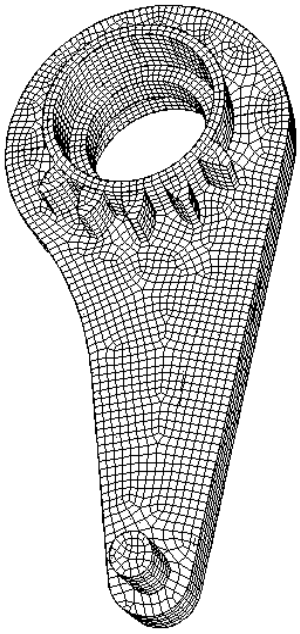


Figure 4.14 - Example Volume with Through Hole

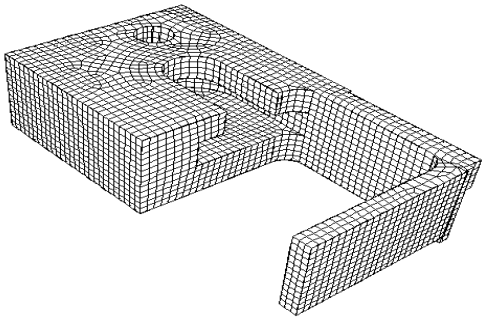


Figure 4.15 - Example Volume with a Through Hole

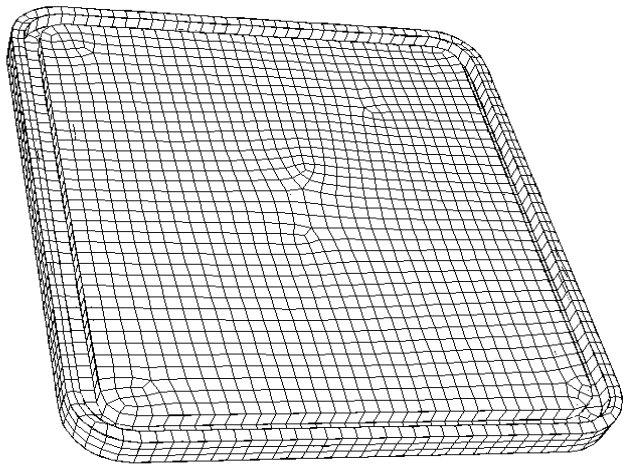


Figure 4.16 - Example Volume with a Blind Hole

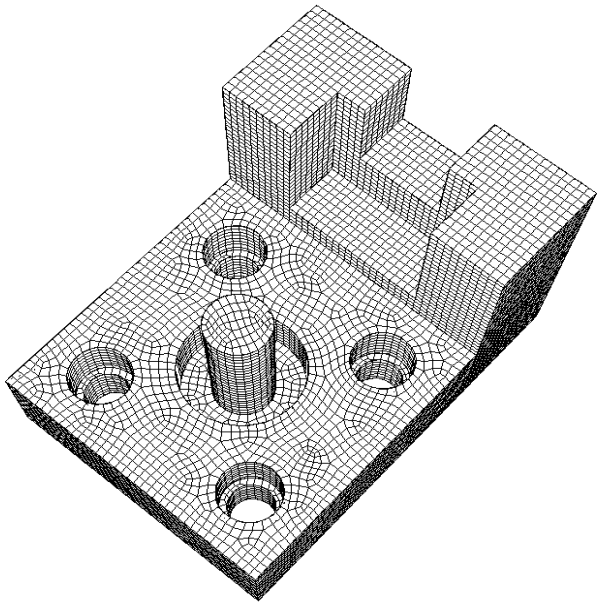


Figure 4.17 - Example Volume with a Blind Hole and Through Holes

Chapter 5 - Conclusion

This chapter presents a brief summary of the research and algorithms developed in this thesis. Finally, several possible areas of future enhancements and research topics that were not covered in this thesis will be discussed.

Summary

Surface meshing algorithms require certain relationships between the number of mesh edges (intervals) on the curves bounding a surface. Assigning the number of intervals to all of the curves in the model such that all relationships are satisfied is called interval assignment. Volume meshing algorithms also require certain relationships between numbers of intervals that are not always captured by the surface meshing requirements. For example, sweeping a hollow cylindrical solid requires that the numbers of intervals between the top and bottom annuluses are the same for the inner and outer cylinder walls.

This thesis presented a new technique for automatically identifying volume interval constraints. Volume interval constraints were grouped with surface interval constraints and solved simultaneously. This technique reduced the amount of user time required to mesh models composed of sweepable volumes with holes; previously a user often had to manually identify constraints and set intervals before these volumes would successfully mesh.

A sweepable volume has source, target, and linking surfaces. Each maximal edge-connected set of linking surfaces defines a blind-hole, a through-hole, or the outer shell of the volume. Note the outer shell is topologically equivalent to a through-hole. Within a linking set, the numbers of intervals between source and target surfaces are already favorably constrained by the surface mapping constraints. However, between two linking sets the numbers of intervals may need to be explicitly constrained for the volume.

The procedure described in this thesis used graph algorithms to identify linking sets, and determine if they correspond to through-holes or blind-holes. For blind-holes, the algorithm generates constraints that prevent the hole from being too deep in interval parameter space and penetrating opposite target surfaces. For each linking set, the adjoining source and target surfaces are partially ordered by the structure of the linking set. Representative chains of curves capture this partial ordering; the level of a surface at the end of a chain must be equal to the level of the surface at the beginning of the chain plus the number of intervals assigned to the chain. A small set of representative paths for each linking set is found. Note that not all source/target pairs generate a path. The representative paths for all linking sets are gathered and distilled by Gaussian elimination into a small set of constraints.

Interval assignment has other considerations besides meshing scheme constraints: a user sets the number of intervals on individual curves, and designates them as hard-set (cannot be modified) or soft-set (merely a goal). Note that in some cases there is no interval assignment solution. The interval assignment constraints and goals are solved by a series of (integer) linear programs. The resulting numbers of intervals are assigned to each curve in the model, and subsequently meshing the surfaces and volumes will not change these numbers.

Future Areas of Research

While many areas of possible research in volume interval assignment have been covered in this thesis, a few areas still remain. This section will discuss five areas, which are:

1. Resolution of path initialization problems
2. Edge parameterization
3. Corner picking globalization
4. Volume interval assignment for submappable volumes
5. Blind hole interval constraint formulation for many-to-many sweeps

Path Initialization

The original research for volume interval assignment made the assumption that only surfaces with multiple edge loops would contain independent and parallel edge paths leading from the source surface(s) to the target. In some cases, this is not necessarily true. Consider the volume shown in Figure 5.1, where a source has a single edge loop, but contains multiple chains of linking surfaces.

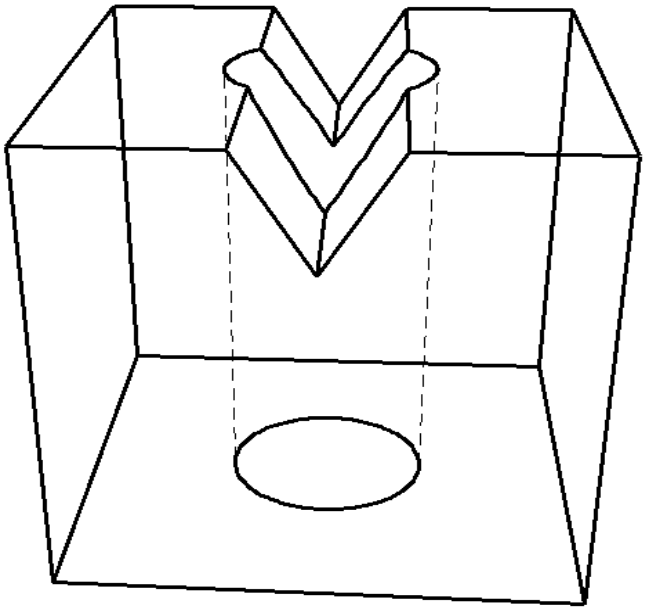


Figure 5.1 - Path Initialization Problem

A solution to the problem shown in Figure 5.1 is to search for the edge paths both from the source surface(s) to the target surface, and then from the target surface to the source surface(s). This solves the problem for the multiple source to one target classification; however, there is still potential for a small set of volumes in the multiple source to multiple target classification to have an incomplete set of constraints to verify the sweepability of the volume.

Edge Parameterization

The constraint equations from the volume interval assignment algorithm depend on the weight assigned to each edge in the edge path. The weights are assigned by determining if the edge is parallel, perpendicular, or parallel but opposite to the volume sweep direction. The sweep direction is calculated locally by using the angles between

successive edges in the edge path. Angles between successive edges are calculated using an interior angle of two vectors formed from the edges. For some cases, this is not necessarily indicative of the sweep direction for the volume. One possible solution to this problem is to parameterize the edges of the volume globally using the surface vertex angle and assigning the edge to an [i-j] space, where the [i] direction is always the sweep direction. This, however, is not an easy problem and brings us to the next area of future research, global corner picking.

Global Corner Picking

The corner picking algorithm for surfaces often guarantees success for the surface locally, but in some instances, can cause an infeasible or unsolveable set of constraints within the interval assignment linear program. An example of this type of infeasibility is shown in Figure 5.2. If vertex “A” is chosen as a corner for the linking surface, the volume is no longer sweepable with the source and target surface specified, and, subsequently, the edge parameterization for the volume will also be incorrect. An algorithm is needed which will pick the corners to ensure feasibility of the global problem for the interval assignment algorithm. Such a corner picking algorithm could also be used to verify the sweepability of a volume, given the source and target surfaces.

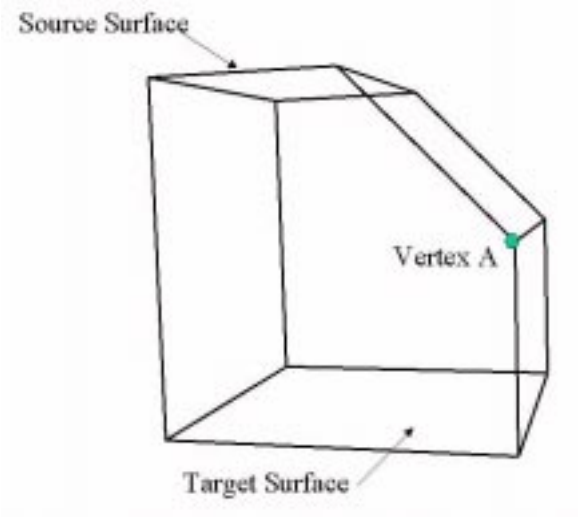


Figure 5.2 - Global Corner Picking Problem

Volume Interval Assignment for Submappable Volumes

Submappable volumes can, to some extent, be thought of as a specialized sub-set of sweepable volumes. Volume interval assignment for a submappable volume could then be handled similarly to the algorithm for volume interval assignment for sweepable volumes.

A submappable volume can essentially be swept in three directions. By ensuring that the volume interval constraints are formulated and supplied to the linear program in each of the three sweep directions, submappability with respect to interval assignment can be assured using the same algorithm described by this thesis.

Blind Holes and Many-to-Many Sweeps

Multiple blind holes on a multiple source to multiple target sweepable volume present an interesting problem for interval assignment. Not only are the intervals constrained within the blind hole to be less than the intervals along the outer surface of the

volume, but there is also the problem of potential overlap for blind holes protruding from a source and blind holes protruding from target surfaces. Figure 5.3 shows the possible cases of blind holes in many-to-many volume sweeps.

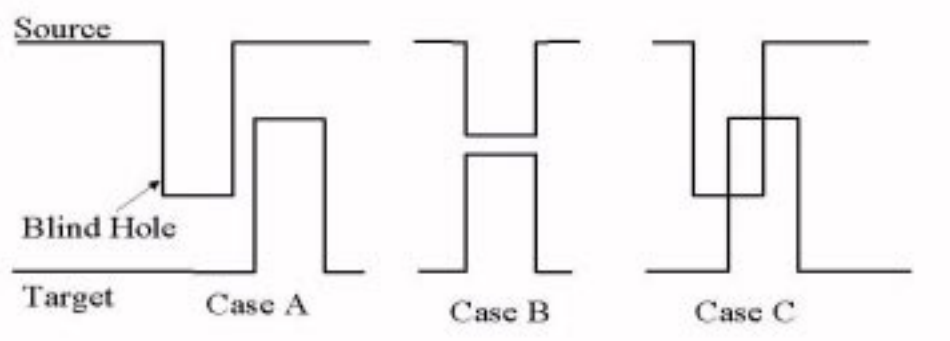


Figure 5.3 - Blind Hole Cases in Many-to-Many Sweeps.
Case A: No Potential Overlap;
Case B: Full Potential Overlap;
Case C: Partial Potential Overlap

Potential solutions to this problem must handle the detection of overlap between the blind hole's bottom and the geometry boundary. One idea has been to use the intersection detection algorithm described by Mingwu Lai in his work on multiple source to multiple target sweeping[8]. However, the algorithm assumes a proper interval assignment prior to the intersection detection. Thus the problem becomes circular, and the difficulty of the problem increases.

Glossary

Chain of Surfaces - A set of surfaces connected to each other by shared edges. Chains of surfaces usually form closed loops and are usually found as the outer skin of a volume or a set of surfaces comprising a through hole. In a sweepable volume, the linking surfaces will form at least one chain around the volume. Multiple chains of linking surfaces are found on sweepable volumes containing holes.

Graph - A graph consists of a set of objects known as vertices, and another set known as edges, such that each edge is identified with an unordered pair of vertices. Graphs are usually represented by means of a diagram.

Linking Surface - The surfaces which connect the source and target surfaces on a sweepable volume. Linking surfaces must be mappable or submappable.

Source - The surface or set of surfaces on a sweepable volume upon which the mesh originates. The source surface must be topologically equivalent to the target surface.

Sub-Vertex - A vertex in the graph derived from a super vertex and containing only part of the total connectivity data for a super vertex. Sub-vertices are formed from individual loops of edges on a source surface, or set of source surfaces. Subsequent searches of the sweepable volume graph usually originate from a sub-vertex.

Super Vertex - A collection of sweep vertices containing all connectivity data of each sweep vertex. The super vertex is formed by collapsing all sweep vertices on a source or target surfaces into a single vertex.

Sweep Vertex - On a sweepable volume, a sweep vertex is a vertex in the graph of the volume. A sweep vertex contains the connectivity data to other sweep vertices which forms the graph. The connectivity data is simply the edges in the graph to which this vertex is connected.

Sweepable Volume - A volume is said to be sweepable if topologically equivalent source and target surfaces are connected by mappable or submappable linking surfaces. The source is meshed first, and then copied, or “swept”, layer by layer along the linking surfaces.

Target - The surface or set of surfaces on a sweepable volume upon which the mesh terminates. The target surface must be topologically equivalent to the source surface or surfaces.

Works Cited

1. Bathe, K., "Current Directions in Meshing," *Mechanical Engineering,* July 1998, pp. 70-71.
2. Blacker, T., "The Cooper tool," *Proceedings of 5th International Meshing Roundtable '96,* Sandia National Lab., 1996, pp. 13-29, 1996.
3. Blacker, T. D., et al., 1994, *CUBIT Mesh Generation Environment Users Manual Vol. 1,* SAND94-1100, Sandia National Laboratories, Albuquerque, NM.
4. Blacker, T. D., et al., 1988, "Automated Quadrilateral Mesh Generation: A Knowledge System Approach", ASME Paper No. 88-WA/CIE-4.
5. Cormen, T. H., Leiserson, C. E. and Rivest, R. L., *Introduction to Algorithms,* New York: McGraw-Hill, 1990.
6. Deo, N., *Graph Theory with Application to Engineering,* Prentice-Hall, Inc., Englewood Cliffs, N. J., 1974.
7. Knupp, P. M., "Next-Generation Sweep Tool: A Method for Generation All-Hex Meshes on Two-and-One-Half Dimensional Geometries," *Proceedings 7th International Meshing Roundtable '97,* Sandia National Laboratories, pp. 505-513.
8. Mingwu, L., *Automatic Hexahedral Mesh Generation by Generalized Multiple Source to Multiple Target Sweeping,* Published Doctoral Dissertation at Brigham Young University, 1998.
9. Mitchell, S., "High Fidelity Interval Assignment," *Proceedings 6th International Meshing Roundtable '97,* Sandia National Laboratories, pp. 33-44.
10. Staten, M. L., Canann, S. A. and Owen, S. J., "BMsweep: Locating Interior Nodes During Sweeping," *Proceedings 7th International Meshing Roundtable '98,* Sandia National Laboratories, pp. 7-18.
11. Stephenson, M. B. and Blacker T. D., "Using Conjoint Meshing Primitives to Generate Quadrilateral and Hexahedral Elements in Irregular Regions," *Computers in Engineering,* Book No. G0502B, 1989, pp. 163-172.
12. Tam, T. K. H. and Armstrong, C. G., "Finite Element Mesh Control by Integer Programming," *International Journal for Numerical Methods in Engineering,* vol. 36, pp. 2581-2605, 1993.
13. White, D., *Automatic, Quadrilateral and Hexahedral Meshing of Pseudo-Cartesian Geometries Using Virtual Subdivision,* Published Masters Thesis at Brigham Young University, 1996.

14. Whiteley, M., *An Automated Mesh Control Procedure for Generalized Mapped Meshing*, Published Masters Thesis at Brigham Young University, 1995.

15. I-Deas User's Guide, -Structural Dynamic Research Corporation.

16. Pro/Engineer, A Solid Modeling Engine of Parametric Technology Corporation.

17. Ignizio, J. P., Cavalier, T. M., *Linear Programming*, Prentice Hall, New Jersey, 1994.

18. Whiteley, M., White, D., Benzley, S.E., and Blacker, T.D., "Two and Three-QuarterDimensional Meshing Facilitators," *Engineering with Computers*, 12, pp.144-154, 1996.

19. Wolfe, C. S. *Linear Programming Algorithms"*, Prentice-Hall, Virginia, 1985.