SAN099-2299J

Interval Assignment for Volumes with Holes

Jason Shepherd (jfsheph@sandia.gov)
Parallel Computing Sciences Department
Sandia National Laboratories, Albuquerque, NM 87185

Steven Benzley
Civil and Environmental Engineering Department
Brigham Young University, Provo, UT 84604

Scott Mitchell[1]
Parallel Computing Sciences Department
Sandia National Laboratories, Albuquerque, NM 87185

Abstract

This paper presents a new technique for automatically detecting interval constraints for swept volumes with holes. The technique finds true volume constraints that are not necessarily imposed by the surfaces of the volume. A graphing algorithm finds independent, parallel paths of edges from source surfaces to target surfaces. The number of intervals on two paths between a given source and target surface must be equal; in general, the collection of paths determine a set of linear constraints. Linear programming techniques solve the interval assignment problem for the surface and volume constraints simultaneously.

Keywords: meshing, hexahedron, sweeping, linear program, interval assignment

[1] Scott Mitchell was supported by the Mathematical, Information and Computational Sciences Division of the U.S. Department of Energy, Office of Energy Research. Scott and Jason work at Sandia National Laboratories, operated for the U.S. DOE under contract No. DE-AL04-94AL8500. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the U.S. DOE.

DISCLAIMER

Introduction

The finite element method is a fundamental modeling technique with widespread use and growing popularity in the engineering community. The 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. 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 part of using the finite element method is the discretization of the model's geometry into finite elements, the process known as mesh generation.

The quality of the mesh affects the accuracy and efficiency of the finite element method. In practical cases, it is difficult to have perfectly shaped elements because many of the elements must be distorted to fit the geometry. The quality of the depends on the technique, or "scheme", selected to generate the mesh.

Mapping and submapping schemes are designed to place a structured mesh[2] on surface and volume geometry. The regular structure of the mapping and submapping meshes imposes constraints on the mesh boundary: two sets of edges are paired and the number of mesh edges, or "intervals", are constrained to be equal.

Surfaces are often connected to other surfaces by shared edges. The interval constraints imposed on an edge can propagate across sets of mapped or submapped surfaces, thereby affecting the meshes generated on distant surfaces.

The quality of a structured mesh depends on having roughly equally sized edges on its sides. An interval count, which enhances the global quality of the final mesh, is desired. The intervals assigned to the edges can be optimized through the use of linear programming

routines.[3,4,5] The linear program takes into account the constraints imposed on each of the edges of the volume by the surface meshing scheme and geometry, and attempts to find an optimal set of intervals which satisfy all of the constraints and produce a quality mesh.

Constraints other than those arising from the surfaces are also known to exist: in some cases, interval constraints also need to be propagated through the volume. This is the case on a swept volume with holes.

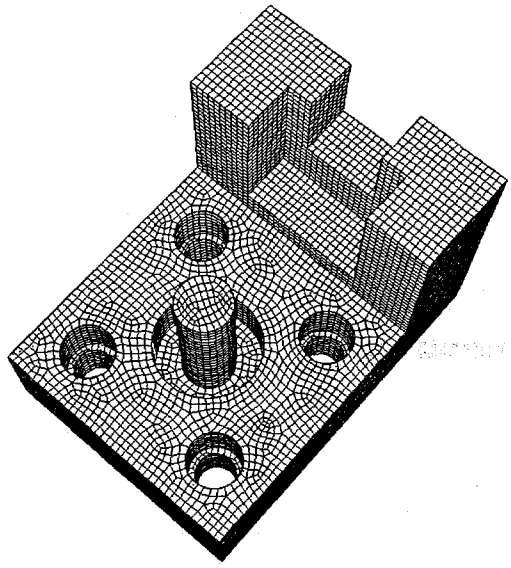


Figure 1 – Swept Volume with Mesh

A swept, or two and one-half dimensional, volume is a volume that has a topologically constant cross section along a single axis.[6] 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] An example of a swept mesh is given in Figure 1.

Swept volumes with holes pose an interesting problem to the interval assignment process.

Because the linking surfaces within the hole are not connected through any shared edges to the linking surfaces on the boundary of the volume, the constraints imposed on the edges of these two sets of surfaces are independent, ignoring volume constraints for the moment. Without the ability to match these two sets of intervals, the sweeping algorithm may fail. Previously, this matching had to be done explicitly by the user through a tedious manual process.

This paper presents a new graphing algorithm to formulate additional volume interval constraints which couple the intervals within a hole, to intervals on the outer surfaces of a swept volume. The volume and surface constraints are solved with a linear program.[8] Thus, swept volumes may be meshed more automatically.

Interval Assignment Background

The process of interval assignment and control is fundamental to obtaining automatic mapped and submapped meshes on surfaces. In this section, a brief introduction to linear programs is given followed by a brief discussion on their use in interval assignment and control.[3]

Introduction to Linear Programs

A linear program is an optimization problem in which the objective function and design constraints are linear functions of the design variables.[9] 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.

The general form of a linear program is to find an $x = (x_1, x_2,...,x_n)$ that optimizes (maximizes) the objective function subject to the specified constraints. Each constraint may be a greater-than inequality (>=), a less-than inequality (<=) or an equality (=), i.e.,

$$\textit{optimize:} \qquad z = c_1x_1 + c_1x_1 + ... + c_nx_n \qquad (1)$$

$$\textit{subject to:} \qquad a_{1,1}x_1 + a_{1,2}x_2 + ... + a_{1,n}x_n \ (<=, =, >=)\ b_1 \qquad (2)$$

$$a_{2,1}x_1 + a_{2,2}x_2 + ... + a_{2,n}x_n \ (<=, =, >=)\ b_2$$

$$... \qquad ... \qquad ...$$

$$a_{n,1}x_1 + a_{n,2}x_2 + ... + a_{n,n}x_n \ (<=, =, >=)\ b_n$$

A special feature of a linear program results from the fact that the derivatives of the objective function with respect to the design variables are constants. This implies that the extrema of the linear program are located on the boundary of the design space, and not in the interior of the design space. Because the constraint relations in a linear program are also linear functions, there are optimal designs that lie at the intersections of the constraint functions.

Linear Programs and Interval Assignment

The use of linear programming for interval control was first introduced by Tam and Armstrong[4] and later used by Whiteley.[5] 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; see 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.[3,4] The optimal solution to this linear program tends to change the intervals on very few curves, but a given curve's interval change could be quite large.

Later work by Mitchell[3] 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 series of linear programs. Each program fixes one or more deltas which currently determine the weighted maximum, M, then removes the variable. In this step the integer constraints are ignored, so the intervals are not necessarily integral, but are rounded to an integer at each pass. The second step solves the integer program with all of the integer design constraints using a branch and bound. In general, branch and bound can take a long time, but in this case the search space is limited by there being a feasible solution near the solution found in the first step.

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.[3]

Surface Interval Constraints

The constraint equations supplied to the linear program are derived from the meshing algorithms applied to the surface. Additionally, some intervals may be fixed by the user. The constraints derived from the surface meshing algorithms are described in this section.

Mapping Constraints

Surfaces with more that four boundary edges may be mapped. Mapping first finds four sequences of boundary edges forming four logical.[5] The constraint formulated for each mapping surface is that the sum of the intervals on opposing sets of edges must be equal.[11,12]

Submapping Constraints

To facilitate the constraint formulation for submappable surfaces, the surface is placed in an abstract i-j coordinate system. Starting at an arbitrary vertex on the surface and proceeding around the surface in a counter-clockwise direction, each edge is classified as being parallel to the [i] or [j] axis, and directed in the positive or negative direction, i.e., [+i], [-i], [+j], or [-j].[5,13]

The constraints for the submappable and mappable surfaces are similar. 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, or

$$\Sigma I_{[+i]} = \Sigma I_{[-i]} \qquad (3)$$

$$\Sigma I_{[+j]} = \Sigma I_{[-j]} \qquad (4)$$

There are additional constraints which enforce that the surface does not self-overlap in the i-j space. These constraints are added to the linear program dynamically, as they are violated. These constraints are similar to the sweep volume constraints.

Other Constraints

Other meshing schemes can also supply constraints to the edges on a surface. For example, paving,[14] an advancing-front meshing algorithm, requires that the sum of the intervals on all the

curves be even.[3] These constraints are relevant for the source and target surfaces. For additional information on surface intervals constraints, readers are referred to [3], [5] and [13].

Introduction to Volume Interval Constraints

Recall that surface constraints propagate across contiguous surfaces. This section considers volume interval constraints that are not captured by this propagation.

Sweepable Volumes and Volume Interval Constraints

Sweepable volumes are 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 "one-to-one sweep"; a volume with more than one source and only one target is known as a "many-to-one sweep", and a volume with many sources and many targets is known as a "many-to-many sweep".

Until recently, many-to-many sweeps were not possible without decomposing the volume first. But now, they can be done through a series of source and target surface projections and subsequent surface imprinting through sweep layers.[15,16] Each of the sweep layers is represented by a single interval along the linking surfaces. In order to prevent excessive skewing or warping of the imprint, it is important that any different paths within each layer have roughly equivalent lengths.

If the source and target surfaces are removed from the volume, the remaining linking surfaces are connected to each other in distinct sets through their shared edges. Sets of connected linking surfaces indicate through or blind holes, or the outer shell of a volume. Because the interval constraints are propagated through shared edges across mappable or submappable surfaces,

the disconnectedness of the linking surfaces means incompatible intervals may be assigned between holes.

The remedy to this problem is to formulate constraint equations that couple the interval constraints between the disconnected sets of linking surfaces to each other, called "volume interval constraints."

The procedure used to formulate the volume interval constraints 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 from a source surface to a target surface for each disconnected set of linking surfaces, and constrain the sum of the intervals on each edge path to be equal. This effectively couples the interval assignment on the two chains together. The procedure for finding the edge paths and formulating the volume interval constraint equations is demonstrated in the next section.

The Volume Interval Assignment Algorithm

The volume interval assignment routine begins with an arbitrary, possibly many-to-many 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 disconnected sets of linking surfaces and supply additional interval constraints which will couple the interval assignment on the two sets of surfaces together.

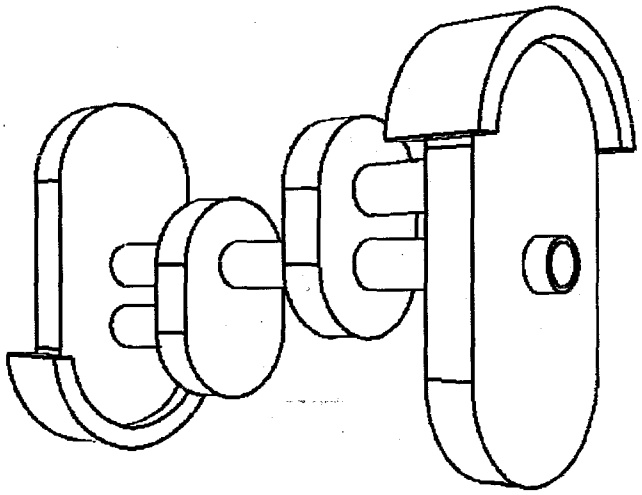


Figure 2 – Example Cam Shaft

Initial Graph Creation

To accomplish the goal of the algorithm, a graph is created using the volume geometry. A graph is simply 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.[17,18]

There are three types of graph vertices. Initially, our graph is a 1-1 abstraction of the volume's vertex-edge topology: each of the volume's vertices has a corresponding "sweep vertex", and each of the volume's edges has a corresponding "sweep edge" connected to two sweep vertices. Besides sweep vertices, "sub-vertices" and a "super vertices" are collections of sweep vertices that have been merged together. Both sub-vertices and super vertices contain all the connectivity information of each of the sweep vertices they contain. Sub-vertices and super vertices will be discussed in greater detail in the next section.

An example of a sweepable volume is shown in Figure 2. This volume will be used throughout the remainder of the paper to demonstrate how the volume interval constraints will be formulated.

Final Graph Creation

When surfaces have more than one loop, the volume edges do not form paths which connect all of the sweep vertices to all other sweep vertices in the volume, that is, the graph is not connected. To alleviate this problem and to simplify later searches, a 'collapse' of each source and target surface takes place. This 'collapse' coalesces each of the sweep vertices on a single source or target surface into a single 'super' vertex in the graph. A super vertex retains the connectivity knowledge of each individual sweep vertex on the source/target surface: if two sweep vertices are connected by a sweep edge, then super vertices containing them are also connected by a sweep edge. The collapsed graph for the example volume is shown in Figure 3.

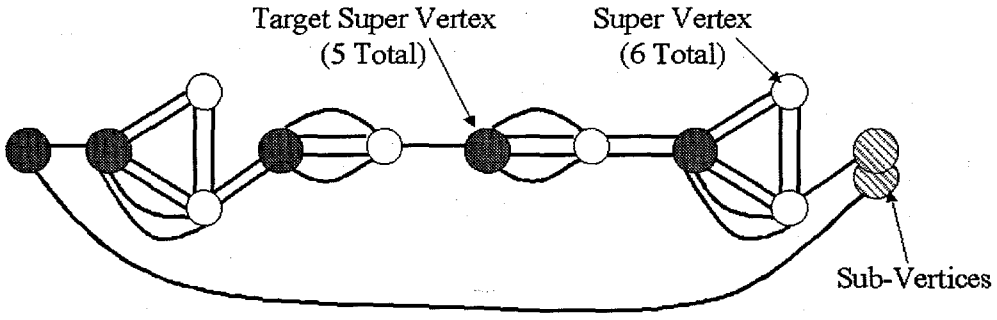


Figure 3 – Final Graph for the Cam Shaft Example

Searching the Graph

With final graph setup completed, edge paths from a source surface to a target surface can easily be found using a breadth-first search algorithm.[17,18] Recall that the goal of the algorithm is to find independent paths between a source and target surface. The branching points of these paths are super vertices corresponding to source surfaces touching two or more disconnected sets of linking surfaces. Hence, only source surfaces with disconnected sets of linking surfaces need to be searched. The current implementation assumes that each instance of disconnectedness corresponds

to a case of multiple loops of edges on the source surface(s), as shown in Figure 4. The loops of curves are found by first collating the source or target surfaces with shared curves. If the remaining curves on the surface(s) form more than one loop, then this surface becomes a candidate for the formulation of the additional volume interval constraints. A more general solution is described in the 'Future Areas of Research' section under 'Path Initialization'.

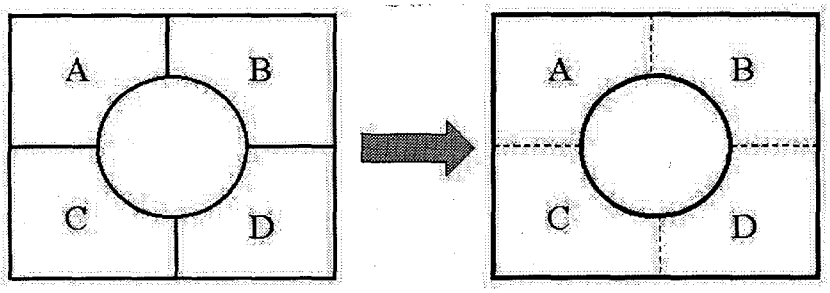


Figure 4 – Combining Edges on Surfaces A, B, C, & D to Form Two Edge Loops

From each loop of curves on the source surface 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 of curves on the source surface(s) can be thought of as a sub-vertex within the collapsed surface(s) super-vertex. Each sub-vertex contains only the connectivity information that can be derived from the sweep vertices contained in the loop of curves.

Searches are initiated from these sub-vertices, and this is there only use. Sweep vertices of the initial super vertex not in the initial sub-vertex are "blacked out" to prevent the search from going through them. This ensures that the search cannot loop back on itself, and also ensures different edge paths from the source surface to the target surface for each sub-vertex.

The search continues for each sub-vertex until the first, or specified, target super-vertex is encountered. At this point, the search ends successfully, and an ordered list of sweep vertices is

obtained leading from the source surface to the target surface. This ordered list of sweep vertices can then be translated to a set of geometric edges representing an edge path from the source surface 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 inside the blind hole to be less than the intervals outside 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 5.

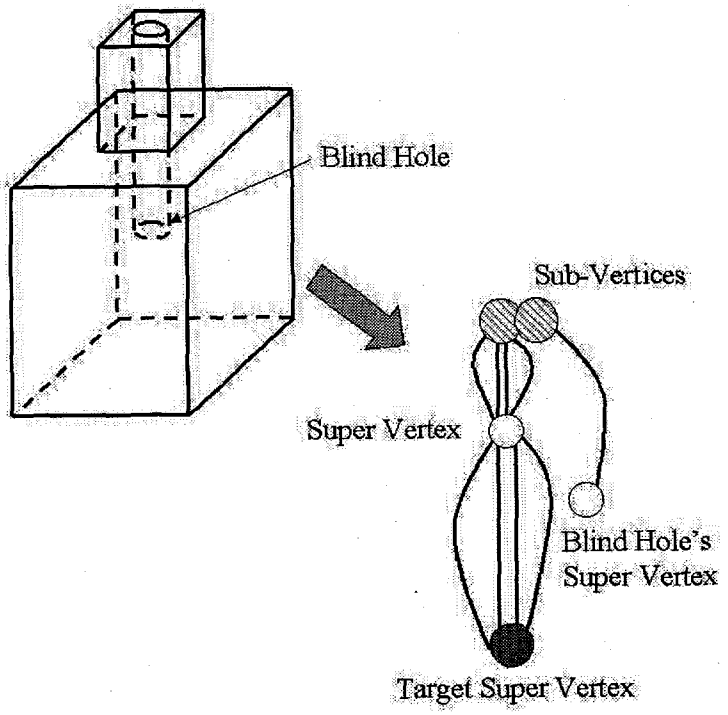


Figure 5 – Graph with a Blind Hole

Figure 6 shows this volume with the surfaces unfolded to aid in visualizing the constraint equations. The surface and volume interval constraint equations for this volume are:

Surface Constraints:

$$I_{13} = I_{14} = I_{18} = I_{20} \quad (5)$$

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

$$I_{26} = I_{27}$$

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

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

Volume Constraint:

$$I_{26} < I_1 + I_{13} \quad (6)$$

Where: I_n = the number of intervals on curve n

The additional volume constraint shown effectively couples the surface interval constraints to the exterior surfaces.

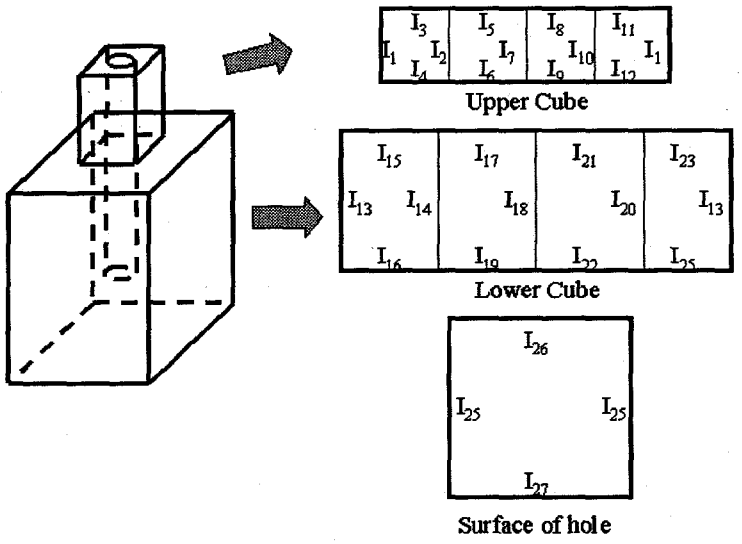


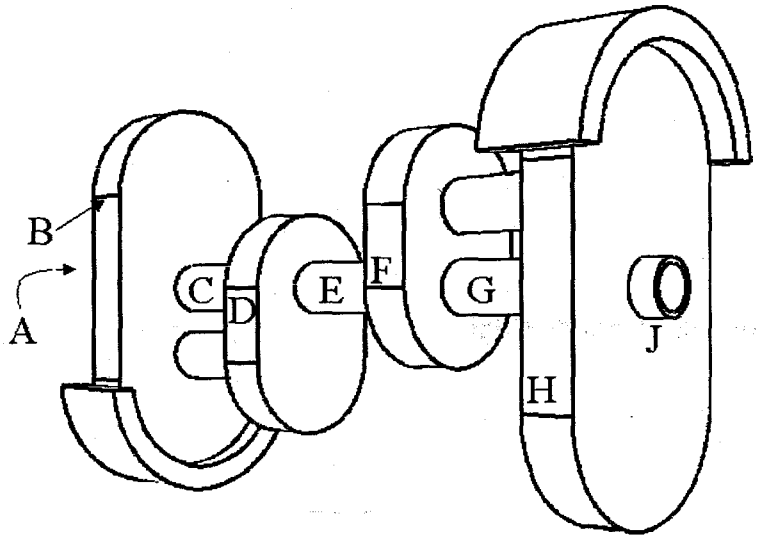
Figure 6 – Constraint Equation Map

Constraint Formulation

With all edge paths from a source surface to a target surface defined, the constraint can be formulated which couple all of the edge paths together. This is accomplished by first attaching a weight, {-1, 0, 1}, to each of the edges, and constraining the sum of the weighted intervals for each edge path to be equal to every other edge path.

The weight associated with each edge is found by systematically traversing the edge list and determining the direction of traversal for the edge. Any edge being traversed in the sweep direction is given a weight of one, opposite the sweep direction is given a weight of negative one, and perpendicular to the sweep direction is given a weight of zero.

Once all edges in the edge list have a weight assigned to them, the constraint equations can be formulated. The sum of the weighted intervals for each edge path is set equal to every other edge path from a given source to a given target surface. Constraints for volumes with blind holes are formulated in a similar manner, using a less-than inequality (<=) instead of an equality constraint (=). An example of the constraint equation formulated for the example sweepable volume is given in Figure 7.



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

Figure 7 – Volume with the Additional Volume Interval Constraint
Note: K is an edge extending through the interior hole.

Interval Assignment

Before adding the constraints to the linear program, they are preprocessed. First, they are written in matrix form. A Gaussian elimination step reduces the constraints to a minimal spanning set of constraints. The reduced constraints are then added to the linear program.[3,5]

The additional volume constraints in the linear program enable the surface and volume constraints to be solved simultaneously. This technique reduces the amount of user time required

to mesh models composed of sweepable volumes with holes; where previously a user often had to manually identify constraints and set intervals before these volumes would successfully mesh.

Other Examples

Figures 8 and 9 display other examples of meshed volumes to which this algorithm has been applied.

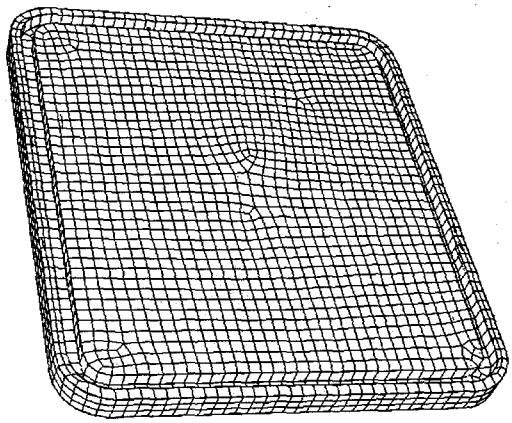


Figure 8 – Example swept volume with a blind hole where volume constraints are needed

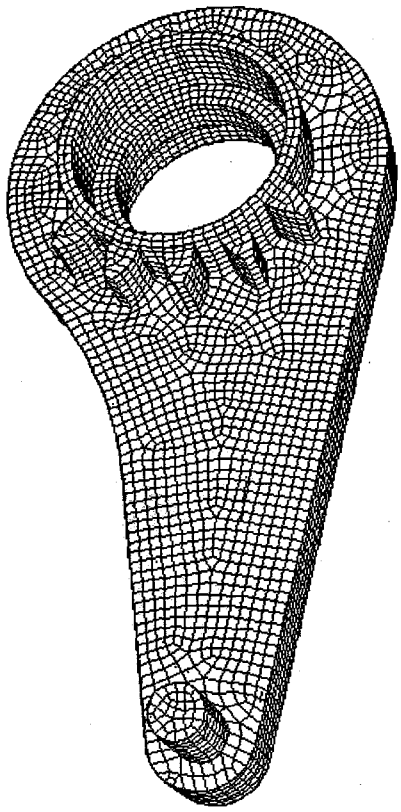


Figure 9 – Example swept volume where volume constraints are needed

Summary

A sweepable volume has source, target, and linking surfaces. Connected sets of linking surfaces define blind-holes, through-holes, and 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 paper uses 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 terms of intervals 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. In addition to the constraints imposed by the structured ordering of the surfaces, volume constraints are also needed to maintain the ordering throughout the volume. Representative edge paths from each linking set are found and constraints are formulated between each set of edge paths. The representative paths for all linking sets are gathered and distilled by Gaussian elimination into a small set of constraints.

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

Path Initialization

The implementation of volume interval assignment makes the assumption that only surfaces with multiple edge loops touch independent and parallel edge. In some cases, this is not necessarily true. Consider the volume shown in Figure 10, where a source has a single edge loop, but contains multiple chains of linking surfaces. The formulation of constraints for this case can be solved for the many-to-one type sweeps by searching from the source to the target, and again from the target to the source. However, this procedure will not work on some many-to-many type sweeps.

One possible solution to the problem is to find all cycles in the final graph. The sum of the weighted edges in any cycle in the graph must sum to be zero. However, this process does not work for blind holes.

A second solution to the problem uses the geometric edges source and target surfaces to determine changes in traversal direction. As each loop of edges is traversed on the source/target surface(s) each time the dihedral angle of the edge and the two connected surfaces changes

dramatically from the previous edge, a new path of edges from the source to the target is computed. Therefore, a larger number of constraints are generated for each source/target combination in an attempt to capture all of the independent, parallel paths of edges. While this solution may work for an additional set of volumes, it is not a complete solution to the problem

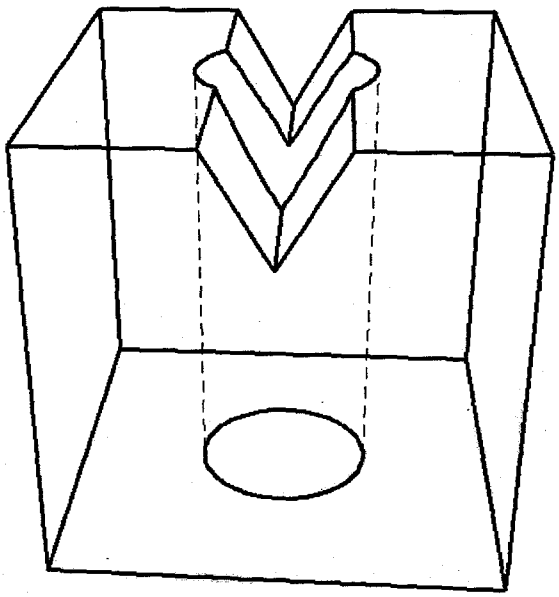


Figure 10 – Path Initialization Problem

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 11 shows the possible cases of blind holes in many-to-many volume sweeps.

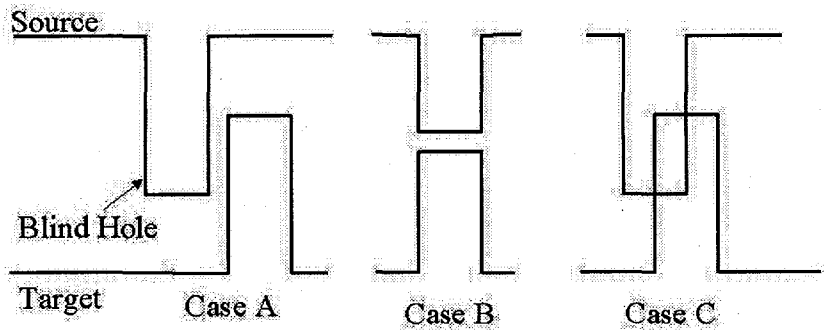


Figure 11 – Blind Hole Cases in Many-to-Many Sweeps. A – No Potential Overlap, B – Full Potential Overlap, C – Partial Potential Overlap

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.[16] 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.

References

1. Bathe, K., "Current Directions in Meshing," Mechanical Engineering, July 1998, pp. 70-71.
2. Blacker, T. D., et al., 1988, "Automated Quadrilateral Mesh Generation: A Knowledge System Approach", ASME Paper No. 88-WA/CIE-4.
3. Mitchell, S., "High Fidelity Interval Assignment," Proceedings 6th International Meshing Roundtable `97, Sandia National Laboratories, pp. 33-44.
4. 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.
5. Whiteley, M., An Automated Mesh Control Procedure for Generalized Mapped Meshing, Published Masters Thesis at Brigham Young University, 1995.
6. 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.
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. Shepherd, Jason F., Interval Matching and Control for Hexahedral Mesh Generation of Swept Volumes, Published Masters Thesis at Brigham Young University, 1999.
9. Ignizio, J. P., Cavalier, T. M., Linear Programming, Prentice Hall, New Jersey, 1994.

10. Wolfe, C. S. Linear Programming Algorithms, Prentice-Hall, Virginia, 1985.
11. Whiteley, M., White, D., Benzley, S.E., and Blacker, T.D., "Two and Three-Quarter Dimensional Meshing Facilitators," Engineering with Computers, 12, pp.144-154, 1996.
12. 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.
13. White, D., Automatic, Quadrilateral and Hexahedral Meshing of Pseudo-Cartesian Geometries Using Virtual Subdivision, Published Masters Thesis at Brigham Young University, 1996.
14. Blacker, T., Stephenson, Michael B., "Paving: A New Approach to Automated Quadrilateral Mesh Generation, " International Journal for Numerical Methods in Engineering, vol. 32, pp. 811-847, 1991.
15. Blacker, T., "The Cooper tool," Proceedings of 5th International Meshing Roundtable `96, Sandia National Lab., 1996, pp. 13-29, 1996.
16. Mingwu, L., Automatic Hexahedral Mesh Generation by Generalized Multiple Source to Multiple Target Sweeping, Published Doctoral Dissertation at Brigham Young University, 1998.
17. Cormen, T. H., Leiserson, C. E. and Rivest, R. L., Introduction to Algorithms, New York: McGraw-Hill, 1990.
18. Deo, N., Graph Theory with Application to Engineering, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1974.