Automatic Hexahedral Sweep Mesh Generation of Open Volumes

Nilanjan Mukherjee[*], Bhanu Peddi, Jean Cabello, and Michael Hancock

Meshing & Abstraction Group
Digital Simulation Solutions
Siemens PLM Software
SIEMENS
2000 Eastman Dr., Milford, Ohio 45150 USA
{mukherjee.nilanjan,bhanu.peddi,jean.cabello}@siemens.com,
hancock.michael@siemens.com

Summary. Open volumes with partial surface geometry are often encountered in the industry while modifying or morphing legacy meshes or while creating meshes in structural recesses or void regions. The paper describes an algorithm for auto-generating hexahedral meshes on these open volumes without the need for laborious CAD repairs. Based on the available input, the open volumes are first topologically categorized into five classes. From the free edges and vertices of the open volume, a new, temporary, mesher-native and lightweight void-topology is first created. The void-topological network is supplemented by an underlying void-geometry which is also an abstract, light-weight, mesher-native discrete data-set. With the help of these void vertices and edges the open or void-volume is topologically and geometrically sealed. The source mesh is first morphed on the target if the latter is present. Transfinite meshing is done on the wall faces. A 2½D sweep meshing approach is used to complete the hex mesh. Thus, during the mesh generation process, facetted geometry is automatically created that can be handed over to the CAD engine.

Keywords: open volume, hexahedral meshing, sweep, void topology, void geometry, transfinite.

1 Introduction

Multiple industrial finite element analysis models experience “Open Volumes”. Some of these could be voids or structural recesses that are packed with non-structural material. Typical examples include void spaces between the content and its container filled with bubble material in drop-box analyses; epoxy or resin filled

[*] Corresponding author.

zones around fibrous material in micromechanical analyses, explosive filled spaces between missile parts, bone marrow spaces in bone cavity or interstitial fluid zones in bioengineering analyses etc. Traditional approach for generating solid meshes in such void regions is to use CAD engines to create NURBS faces and stitch them in to make a watertight CAD solid body. Needless to say it is an inefficient and costly solution.

In this paper we attempt to solve the problem on the mesh generation side. The meshing engine developed at Siemens already creates an abstract lightweight representation of the input CAD or discrete geometry for the purpose of mesh generation. To this mesher-native abstract geometry network, we add new topology and geometry entities called "void-topology" and "void-geometry" respectively. Source face vertices and edges are first paired with those of the target face via topological comparison. Next, a free-edge and free-vertex detection system is set up to generate void topological edges and faces called v-TopoEdge and v-TopoFace respectively. Each v-TopoEdge and v-TopoFace can have underlying v-GeomEdge and v-GeomFace respectively. The geometry creation methods are discussed in details.

Once the void topology and void-geometry entities are added to the existing virtual geometry network, volume closure strategies can be used to seal the v-TopoFaces and make a watertight v-TopoVolume that can be hex-meshed. Depending on the available input data, the Open Volume is organized into 5 classes. Class specific closure strategies are used to create a watertight void-volume.

A traditional 2½D sweep meshing approach is used to complete the hex mesh from quadrilateral meshes initially generated on the volume-skin. Thus, in the process of hexahedral mesh generation for the open volume, we also create facetted geometry that can be handed over to the CAD engine to enable it to create a watertight CAD solid.

1.1 Related Work

There are many sweeping algorithms that can mesh various prismatic solids (2½D). The advances in these algorithms enabled the creation of hexahedral meshes with unique and distinct geometries that include non-planar, non-parallel and multiple source and target faces [1, 2]. Most of these algorithms use a 2-D structured or unstructured quadrilateral mesh as source mesh and then create a target mesh by mapping the source mesh [1,3]. Mapping is a popular tool for hex mesh generation, which is implemented to produce final structured hexahedral mesh [4]. Constraints on these hexahedral meshes determine the quality and visual appearance of these meshes. Therefore, a criterion is defined to produce a hexahedral swept mesh for a given geometry [5]. No work in open literature that deals with hexahedral meshing on incomplete or open CAD volumes could be found.

2 Open Volumes and Their Taxonomy

The "Open Volume" is defined by the complete absence of a CAD body and is made up of an incomplete set of connected faces. These faces could either be NURBS faces or tessellated or facetted faces, or abstract geometry (ABSCAD) faces that are a lighter weight, locally healed and sometimes globally modified version of CAD geometry, often generated on legacy fem data. Since the hex mesh to be generated will use a 2½D sweep mesh generation algorithm, the user needs to specify a source face and a target face (if exists) and wall faces (if exist). According to the available data, the open volumes first need to be classified as shown in Table I. This classification will enable us to identify closure strategies for each case that need to be developed within the mesher to produce a sweep-meshable v-Volume (i.e. void-Volume).

CLASS II and III are differentiated by the fact that the former must have "wall-free" vertices. A "wall-free" vertex is defined as a vertex on the source or target face that is not connected to a wall face.

3 Source-Target Topology Matching

In an open volume the source face is not completely connected to the target face. Therefore an edge and vertex correspondence cannot readily be established for all topological entities. The missing relationships need to be discovered. Such discovery is always limited and thus need to be bound by speculated domain of geometric validity. Algorithm 1 describes the matching technique.

Table 1. Taxonomy of Open Volumes

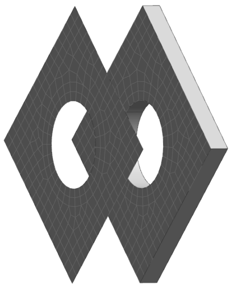
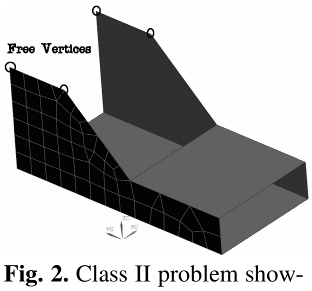
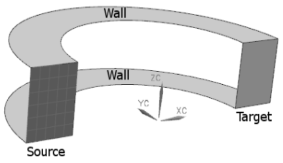
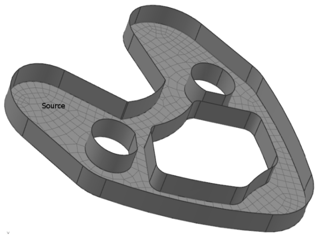
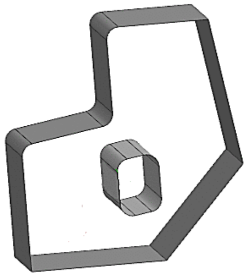
PROBLEM CLASS	Source Face	Target Face	Wall Faces	Examples
CLASS I	✓ (single/ multiple loop)	✓	None	 **Fig. 1.** Class I problem
CLASS II	✓ (single/ multiple loop) Face must have wall-free vertex	✓ Face must have wall-free vertex	✓ (some)	 **Fig. 2.** Class II problem showing wall-free vertices

Table 1. (*continued*)

CLASS III	✓ (single/ multiple loop)	✓	✓ (some)	 **Fig. 3.** Class III problem
CLASS IV	✓ (single/ multiple loop)	---	✓ (all)	 **Fig. 4.** Class IV problem (multiloop) depicting the missing target face
CLASS V (single/multiple loop)	---	---	✓ (all) single/ multiple loop	 **Fig. 5.** Class V problem

Algorithm 1. Source-Target Face Topological Match-Up

1. Query edge loops of the source and the target faces.
2. The source and the target faces need to be topologically similar (same number of loops and edges) but not necessarily be geometric twins. If the number of source face loops $\mathbf{n}_{\mathbf{SL}} \neq \mathbf{n}_{\mathbf{TL}}$, the number of target face loops, algorithm does not continue to the next step.
3. Both source vertices and target vertices location on each loop are identified.
4. Loop Matching is set based on distance between vertices. Thus,
   i. Each source vertex distance to target vertices are measured and where this distance is minimized for a source vertex, the respective target vertex is identified as a match.
   ii. All the target vertices are cycled to identify if a match of source vertex was identified, and in the case where, any of the target vertices have not

been matched or have been matched more than once with source vertices, this is marked as a non-unique map.

5. In the case of a non-unique map, the distances to source vertices from each target vertex are measured and where this distance is minimized for target vertex, the respective source vertex is identified as a match.
6. Loops are now matched based on the vertices and their respective loop directions are then identified. If direction of loops is mismatched, a flip direction flag is turned on for the match.
7. A validation check is performed based on the mean distances.

   i. First mean and standard deviation of matched loops are calculated.
   ii. Identify if at least 50% of these matches fall under 2 standard deviations.
   iii. If the validation fails, then outliers are identified and are used to identify better matches.

The problem, at hand, being ill-defined, obviously the proposed algorithm has its limitations. If auto-matching fails, the user is prompted to manually match the source and target face loops.

4 v-Topology Creation

The traditional recourse to the problem described above is to take the geometry to a CAD product and manually create and stitch-in CAD faces so as to make a closed solid body. However, in the present meshing approach this tedious and manual step is completely eliminated. In order to achieve a closed sweep-meshable body, the void spaces need to be first identified and closed in a meaningful manner. This is done by creating a new kind of abstract, light-weight, temporary void-topology. The creation methods reside within the mesher. We will, from here on, refer to void-topology as “v-Topology” and void-geometry as “v-Geometry”.

The void-volume, to be called v-TopoVolume from here on, is mostly constructed with v-Topology. Each v-Topological entity will have underlying v-Geometry that are simplical geometric representations. The surface mesh generated during sweep meshing could probably be used as the best representation of the “constructed” geometry. For an open volume portrayed in Fig. 6a, the topology tree and its geometric affiliates are described in Fig.7. The TopoFace/TopoEdge(s) define the topology of the given faces, while the “v-Topo” entities refer to the void faces and edges constructed to seal the open volume and convert it to a topologically closed v-TopoVolume as shown in Fig. 6b. v-TopoEdges are always constructed between wall-free vertices.

It is important to remember again that each v-TopoEdge/v-TopoFace may have underlying v-GeomEdge/v-GeomFace which are short-lived, lightweight, discrete geometry that may sometimes be used during mesh generation. The algorithms for v-TopoEdge and v-TopoFace construction are described below in sequence–

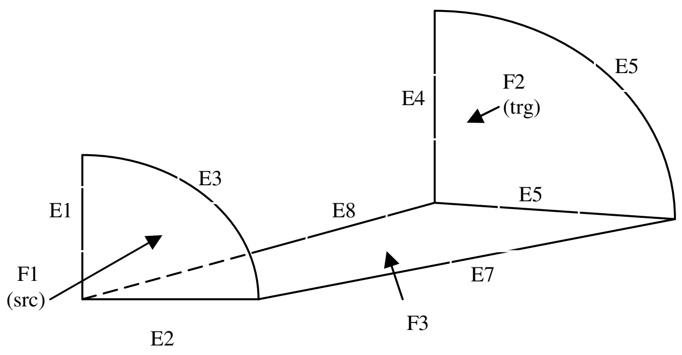


Fig. 6a. A typical open volume with a source, a target face and one wall face

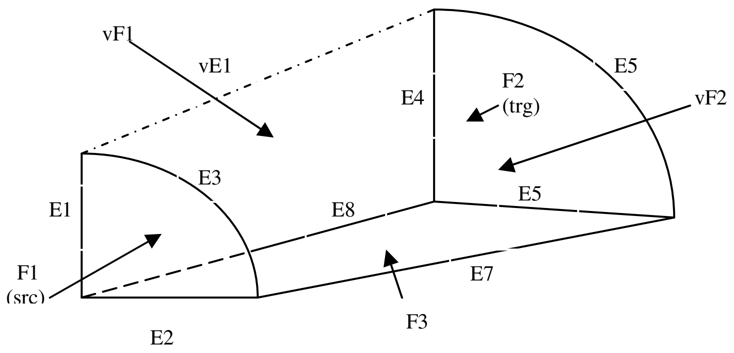


Fig. 6b. The open volume with closed with void topology – one void-edge and 2 void-faces are added to the void-volume

Algorithm 2: Construction of v-TopoEdges

```
1.  After topological match-up of the source and target faces, their geometric vertices are thus paired.
2.  Once the pairing happens, cycle the source vertices
    {
     4a. If the **i-th** source vertex is a wall-free vertex
       {
         4aa. Get the **i-th** target vertex
         4ab. Check if it is a wall-free vertex
                If not, continue cycling
         4ac. If yes, create a v-TopoEdge between these vertices
         4ad. Create a v-GeomEdge which is a straight line between the two vertices.
         4ae. Link the v-Topo & v-GeomEdge and flag it as a wall-edge.
       }
    }
```

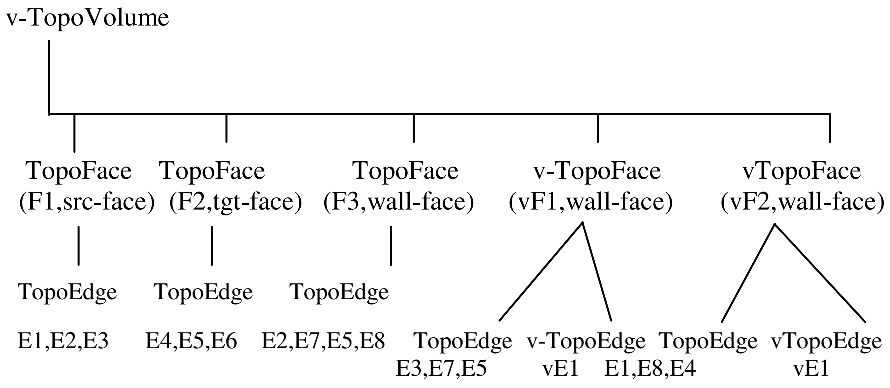


Fig. 7. v-Topology tree for the open volume described in Fig.6a

Once the v-TopoEdges & their v-GeomEdges are constructed, they are stored in a workspace with the other Topo and Geom entities. The v-TopoVolume, which is still open at this point, is reevaluated and a free-edge list ($\mathbf{F}_{\mathbf{el}}$) is created. Algorithm 3 is used at this point to create the v-TopoFaces.

Algorithm 3: Construction of v-TopoFaces

```
1.  While $\mathbf{F}_{\mathbf{el}}$ is not empty
    {
2.    Start a new free-edge list - $\mathbf{F}_{\mathbf{eln}}$
3.    Look for a free-edge on the source, add it to $\mathbf{F}_{\mathbf{eln}}$
4.    While $\mathbf{F}_{\mathbf{eln}}$ does not form a closed loop
      {
5.      Use edge-connectivity to walk the neighbor free-edges.
        All v-TopoEdges are free edges and figure in $\mathbf{F}_{\mathbf{el}}$ initially. They need to
    be checked at this step to see if they are still free, i.e., they are not hooked up
    to v-TopoFaces yet.
6.      Select a neighbor free edge. Add it to $\mathbf{F}_{\mathbf{eln}}$ . Go to step 5.
      }
7.    $\mathbf{F}_{\mathbf{eln}}$ forms a closed loop. Use these free TopoEdges and v-TopoEdges to
    construct a v-TopoFace. At this point the face is a pure topological face with
    no geometry. Geometry data can be created later for the v-TopoFace if de-
    sired. (Ideally, some geometry is always created whether it is needed by
    meshing or not. This is to ensure that the volume generated is always both to-
    pologically and geometrically closed.)
8.    The TopoEdges in $\mathbf{F}_{\mathbf{eln}}$ are removed from $\mathbf{F}_{\mathbf{el}}$.
9.    The v-TopoEdges in $\mathbf{F}_{\mathbf{eln}}$ are re-evaluated and if not free anymore they are
    removed from $\mathbf{F}_{\mathbf{el}}$.
10.   $\mathbf{F}_{\mathbf{eln}}$ is emptied. Go back to step 2.
    }
```

5 v-Geometry Creation

Once the v-Topology entities are established their underlying v-Geometry need to be created. All v-TopoEdges need v-Geometry because all edges are seeded during meshing. Since these edges connect two vertices in void space, there is no intermediate path-data to predict their shape. Thus all v-TopoEdges are assumed to be linear. v-TopoFaces, however, may not necessarily need any geometry for mesh generation, especially when the v-TopoFace is a wall face. All wall faces need 2D transfinite meshes in order to produce a swept hexahedral solid mesh. A 2D transfinite mesh can be generated in 3D space using traditional TFIs. However, to seal the v-Volume both topologically and geometrically we generate v-Geometry on all faces irrespective of the needs of the mesher. Three different methods of geometry creation are used. Furthermore, the v-Geometry created in the process of mesh generation can be handed back to the CAD engine for geometry creation.

5.1 2D Meshing on a Newell Domain

Each v-TopoFace has a facetted boundary data that presents itself as loops of facet edges. Each facet edge is linear and is defined by 2 3D points. Thus the v-TopoFace can be represented as a 3D point-loop. The following algorithm is used to fill the v-TopoFace.

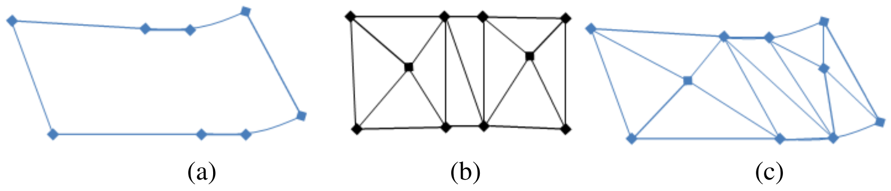


Fig. 8. A 3D v-TopoFace (a) and its corresponding 2D Newell Domain (b) which is meshed and transformed back in 3D to yield the facetted face (c).

Algorithm 4: v-TopoFace geometry creation using meshing on a Newell Domain

1. The 3D polygonal loops (shown in Figure 8a) represented by facet points of the edges that make the boundary of the v-TopoFace are transformed to a 2D plane or a Newell domain [6] which is any 2D plane where the projected 3D area is maximized following a least-square approach.
2. A transformation relationship is established between the 3D and 2D domains expressed by the vector normal to this 2D plane.
3. Inner/Outer loops are identified based on their 2D locations. The 2D domain is used to generate a triangular mesh between them using any 2D meshing algorithm. All interior nodes and elements are transformed to 3D space. This 3D mesh represents the facetted v-TopoFace.

This freshly constructed v-TopoFace can now be flattened and meshed using procedures explained by Beatty & Mukherjee [7]. A limitation of this method of v-TopoFace creation is the limitation of the Newell domain. If the original 3D loops are too curved, the loops can overlap and self-intersect in 2D. In such a scenario we revert to the method described in 5.3.

5.2 v-TopoFace Defined by 2-Loops

For class I problems, source and target face exist but are disconnected (i.e. no wall faces exist between the two) and a surface dependency has been established either automatically or manually, there are some special cases that need special treatment.

One such case is when we have only one free vertex that exists on both source and target (i.e. a loop that consist of one closed edge). In such case, the wall face has only one wall edge defined by the straight line between the source and target vertex. The virtual wall face is not map meshable because we cannot define four logical sides. Another case as described in Figure 9, requires special treatment is two concentric cylinders, one being the source face and the other one the target face with the missing wall faces represented by the top and bottom annular discs. Again, the wall faces (annular discs) are not map meshable as no four logical sides exist.

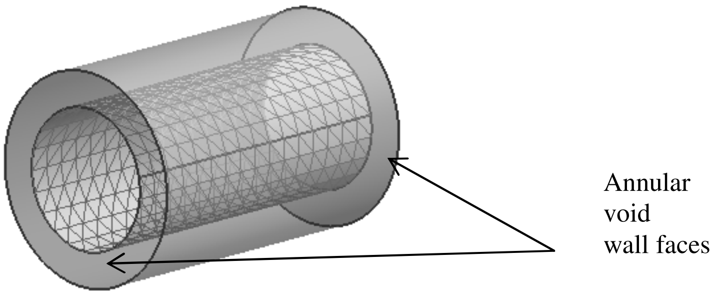


Fig. 9. A Class I v-TopoVolume with annular wall v-TopoFaces

These special cases can be handled in many different ways. In our approach, we directly created a structured mesh in 3D. Algorithm 5 describes the approach for geometry creation for the annular wall face.

Algorithm 5: Geometry creation for annular wall v-TopoFace

1. We have a v-TopoFace that consists of two loops. One loop is the source loop made up of source edges and the other one is the target loop made up of the corresponding target edges. We assume that the number of layer between the source and target is user input.
2. The surface dependency data is also input and consists of a map between the source pair (edge, direction) and the target pair (edge, direction). For each source edge and its direction in the source loop, the corresponding

target edge and its direction in the target loop is defined. Boundary nodes are created on the source edges and their corresponding boundary nodes on the target edges are stored in a map during the boundary node generation.

3. Once the node map between source nodes and target nodes is established, internal nodes along the node line starting at the source node and ending at its corresponding target node are created using a linear interpolation. The node line gets subdivided into **nL** layer segments.
4. Moving counter clock wise along the edges of the source loop, rows of quad elements are created by connecting newly created internal nodes along the node line to previous ones. Finally a structured mesh is generated and stored as the mesh on the virtual wall face.
5. The method described previously applies to not just annular wall faces but to any two loops between a source and a target face that have no wall faces. It is the only recourse when the loop has only one edge. However, when the loop has more n edges [**n**>1]. One can decide to use this method or create **n** v-TopoFaces using Algorithm 6.

5.3 v-TopoFaces with High Boundary Curvature

If a wall v-TopoFace boundary has a large wrapping angle the Newell domain becomes unreliable. Instead, 3D TFI is used to create the final quad mesh on the face before its v-Geometry is determined. Then, the transfinite mesh is used to generate its v-Geometry.

Algorithm 6: Geometry creation for curved v-TopoFaces

1. Use Algorithm 4 to generate face facets even if they overlap/intersect.
2. This v-GeomFace is not used to mesh the wall-face. A bilinear 3D TFI is used to create the quad mesh on the geometry-less face.
3. This final quad mesh is now transformed to facet data. First new facet points are created at interior node locations. All boundary node locations are swapped with facet points closest to them.
4. At this point all nodes have corresponding facet points. Next, all boundary facet (edge interior points only) points are connected to an interior facet location closest to them to form new facets on the boundary.
5. This facet-node relationship is used to create new triangular and quadrangular facets on the face. The quads are next triangulized. This will result in a valid all-tria facet mesh which is used to build the v-GeomFace for this wall v-TopoFace.

6 Volume Closure Strategies

The v-Topology and its underlying v-Geometry are abstract, temporary, lightweight, mesher-native geometry representations that are necessary elements required to create a closed void-Volume for hex meshing. These void topologies and geometries, however, cannot be created according to a generic algorithm. The closure strategies are functions of their class signatures.

As the first step in this process, the mesh on the source face is morphed on to the target following our earlier algorithm [8]. In the process, a map of the source and target nodes (N-Map) and vertices (V-Map) are created.

6.1 Closure Strategy for Class I v-Volumes

In case of Class I problems, there are no wall faces. This scenario does not need a sweep meshing technique to generate the hexahedral mesh. It's a case of pure translation where linear nodelines can be created between the source and target meshes using the N-Map. These nodelines can then be discretized further according to the desired number of layers; auto-generation of the hex mesh is trivial.

6.2 Closure Strategy for Class II v-Volumes

In case of Class II problems, the v-Topology and v-Geometry are needed to build the mesh. Algorithm 8 describes the steps involved.

Algorithm 8: Volume Closure – Class II Topology

1. Use V-Map or N-Map to build vTopoEdges (**E**$_v$)
2. Construct v-TopoVolume (**V**$_v$) with **m** available TopoFaces (**F**)

$$V_v = \cup\ (F_s, F_t) + \cup_{i=1}^{m} F_i \tag{1}$$

3. Check if v-TopoVolume is watertight
4. Find the free edges (a mix of TopoEdges and vTopoEdges)
5. Construct the wall v-TopoFaces (no v-Geometry is present initially)
6. One v-TopoFace (**Fv**) is created per free loop.
7. Use Algorithm 4 or Algorithm 6 to generate the facets needed to create v-Geometry
8. Add all of the **n** v-TopoFaces to v-TopoVolume as wall faces

$$V_v = \cup\ (F_s, F_t) + \cup_{i=1}^{m} F_i + \cup_{i=1}^{n} F_{vi} \tag{2}$$

The volume is now watertight.

6.3 Closure Strategy for Class III v-Volumes

In case of Class III problems, wall-free vertices are not involved, so there is no need for the N/V-Maps. Algorithm 9 describes the steps involved.

Algorithm 9: Volume Closure – Class III Topology

1. Construct v-TopoVolume (**V**$_v$) with available **m** TopoFaces as in eqn. (1).
2. Check if the volume is watertight
3. Find free edges (TopoEdges)

4. Build free loops (single or multiple) to create the v-TopoFaces – one per free loop as in eqn. (1).
5. Use Algorithm 4, Algorithm 5 or Algorithm 6 to build v-Geometry for the v-TopoFaces
6. Add all of the **n** v-TopoFaces to v-TopoVolume as wall faces as described by eqn. (2).

The volume is now watertight.

6.4 Closure Strategy for Class IV v-Volumes

In case of Class IV problems, wall v-TopoFaces, wall-free vertices are not involved. Neither are the N/V-Maps as the target face is missing. Algorithm 10 describes the steps involved.

Algorithm 10: Volume Closure – Class IV Topology

1. Construct v-TopoVolume ($\mathbf{V_v}$) with the available TopoFaces, i.e. the source face ($\mathbf{F_s}$) and the **m** wall faces.

$$V_v = \bigcup_{i=1}^{m} (F_s, F_i) \tag{3}$$

2. Find free edges $\mathbf{E_v}$ (TopoEdges)
1. Build free loops (single or multiple)
2. Use all free loops to build a v-TopoFace ($\mathbf{F_t}$) that represents the target face.
3. Build v-Geometry for the v-TopoFace using Algorithm 4
4. Add v-TopoFace to v-TopoVolume as target face

$$V_v = \bigcup_{i=1}^{m} (F_s, F_i, F_t) \tag{4}$$

The volume is now watertight.

6.5 Closure Strategy for Class V v-Volumes

In case of Class V problems, wall-free vertices are not involved as both the target and the source are void. Algorithm 11 describes the steps involved.

Algorithm 11: Volume Closure – Class V Topology

1. Construct the v-TopoVolume ($\mathbf{V_v}$) with **m** available TopoFaces (**F**)

$$V_v = \bigcup_{i=1}^{m} F_i \tag{5}$$

2. Check if the v-TopoVolume is watertight
3. Find **p** free edges $\mathbf{E_v}$ (TopoEdges)
4. Build free loops (must be an even number of loops)
5. Store the point-loop data in a spatial hash or a 3D Cartesian grid
6. Use the hash/grid to group the loops into two distinct subsets separated by a sweeping distance

7. Build 2 v-TopoFaces ($\mathbf{F_s}$, $\mathbf{F_t}$), one with each loop subset

$$F_s = \bigcup_{i=1}^{p/2} E_{vi} \tag{6}$$

$$F_t = \bigcup_{i=\frac{p}{2}+1}^{p} E_{vi} \tag{7}$$

8. Use Algorithm 4 to build v-GeomFace for each v-TopoFace
9. Add one v-TopoFace to v-TopoVolume as source face
10. Add other v-TopoFace to v-TopoVolume as target face

$$V_v = \bigcup_{i=1}^{m} (F_i,\ F_s,\ F_t) \tag{8}$$

The volume is now watertight.

7 Hexahedral Sweep Mesh Generation

CSALF-Q, a combined recursive subdivision and loop-paving auto quad mesh generation algorithm [8,9] is used to generate the mesh on the source face. The wall faces are meshed using transfinite meshing methods [10]. Once the 2D meshes on the surface (except for the target face) are done, the hexahedral mesh is generated by a 2½D sweep meshing approach. The source mesh is advanced through the volume primarily by means of an affine transformation [11]. Where the affine transformation does not exist, the nodes on the cross-section are advanced according to local normal vectors and smoothed according to various earlier reported techniques [11]. The last layer is finally projected onto the target face and smoothed using a variational smoother [12].

8 Results

The algorithm family described in the above sections are used to create the hex mesh for examples listed in Table 1 in section 2. The results are displayed in Table 2. The final meshes for open volumes of Class I-V are shown in Fig.10 (a-e).

Table 2. Final Hexahedral Meshes on Open Volumes

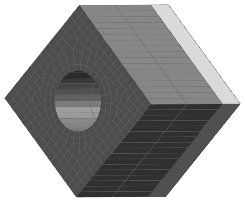
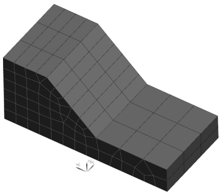
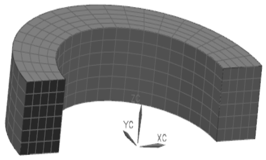
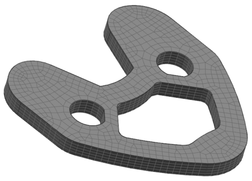
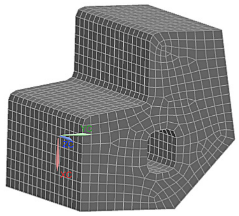
CLASS	Algorithm (V-Topo)	Algorithm (V-Geometry)	Algorithm (Closed Volume)	Figures
I	II,III	IV,VI	VII	 **Fig. 10a.** Hex mesh for a Class I v-TopoVolume

Table 2. (*continued*)

II	II,III	IV,V,VI	VIII	 **Fig. 10b.** Hex mesh for a Class II problem showing wall-free vertices
III	II,III	IV, VI	IX	 **Fig. 10c.** Hex mesh for a Class III v-TopoVolume
IV	II,III	IV, VI	X	 **Fig. 10d.** Hex mesh for a Class IV v-TopoVolume
V	II, III	IV,VI	XI	 **Fig. 10e.** Hex mesh for a Class V v-TopoVolume

References

1. Jankovich, S.R., Benzley, S.E., Shepherd, J.F., Mitchell, S.A.: The Graft Tool: An All-Hexahedral Transition Algorithm for Creating a Multi-Directional Swept Volume Mesh. In: Proceedings, 8th International Meshing Roundtable, pp. 387–392 (1999)

2. Mingwu, L., Benzley, S., Sjaardema, G., Tautges, T.: A Multiple Source and Target Sweeping Method for Generating All Hexahedral Finite Element Meshes. In: Proceedings, 5th International Meshing RoundTable, pp. 217–225 (1996)
3. Lin, H., Liao, H., Deng, C.: Filling Triangular Mesh Model with All-Hex Mesh by Volume Subdivision Fitting. Technical Report, TR ZJUCAD, 002, State Key Lab. Of CAD&CG (2012)
4. Shepherd, J., Johnson, C.: Hexahedral mesh generation constraints. Engineering with Computers 24(3), 195–213 (2008)
5. Delfino, A., Stergiopulos, N., Moore, J.E., Meister, J.J.: Residual strain effects on the stress field in a thick wall finite element model of the human carotid bifurcation. Journal of Biomechanics 30, 777–786 (1996)
6. Sunday, D.: Fast Polygon Area and Newell Normal Computation. In: Barzel, R. (ed.) Graphics Tools. AKPeters, MA (2005)
7. Beatty, K., Mukherjee, N.: Flattening 3D Triangulations for Quality Surface Mesh Generation. In: Proceedings, 17th International Meshing Roundtable, pp. 125–139 (2008)
8. Vurputoor, R., Mukherjee, N., Cabello, J., Hancock, M.: A Mesh Morphing Technique for Geometricaly Dissimilar Tessellated Surfaces. In: Proceedings, 16th International Meshing Roundtable, pp. 315–334. Springer (2007)
9. Mukherjee, N.: CSALF-Q - A Bricolage Algorithm for Anisotropic Quad-Mesh Generation. In: Proceedings, 20th International Meshing Roundtable, pp. 489–510 (2011)
10. Beatty, K., Mukherjee, N.: A Transfinite Meshing Approach for Body-In-White Analyses. In: Proceedings, 19th International Meshing Roundtable, pp. 49–66 (2010)
11. Knupp, P.: Applications of Mesh Smoothing: Copy, Morph, and Sweep on Unstructured Quadrilateral Meshes. International Journal for Numerical Methods in Engineering 45, 37–45 (1999)
12. Mukherjee, N.: A hybrid, variational 3D smoother for orphaned shell meshes. In: Proceedings, 11th Int. Meshing Roundtable, pp. 379–390 (2002)