TRIANGULATION OF ARBITRARY POLYHEDRA TO SUPPORT AUTOMATIC MESH GENERATORS

B. Kaan Karamete*, Mark W. Beall and Mark S. Shephard
Scientific Computation Research Center
Rensselaer Polytechnic Institute
Troy, NY 12180 USA
*Corresponding Author: kaan@scorec.rpi.edu

An algorithm is presented for the triangulation of arbitrary non-convex polyhedral regions starting with a prescribed boundary triangulation matching existing mesh entities in the remainder of the domain. The algorithm is designed to circumvent the termination problems of volume meshing algorithms which manifest themselves in the inability to successfully create tetrahedra within small subdomains to be referred to herein as cavities. To address this need, a robust Delaunay algorithm with an efficient and termination guaranteed face recovery method is the most appropriate approach. The algorithm begins with Delaunay vertex insertion followed by a face recovery method that conserves the boundary of the cavity by utilizing local mesh modification operations such as edge split, collapse and swap and a new set of tools which we call complex splits. The local mesh modifications are performed in such a manner that each original surface triangulation is represented either as was, or as a concatenation of triangles. When done in this manner, it is always possible to split the matching mesh entities, ensuring that a compatible mesh is created. The algorithm is robust and has been tested against complex manifold and non-manifold cavities resulting in a valid mesh of the entire domain.

Key Words: Unstructured Mesh, Face Recovery, Delaunay Triangulation, Robustness.

1 Introduction

Unstructured mesh generation algorithms have been devised by many researchers [1, 2, 3, 4, 5]. Although these algorithms have provided promising results, there are still open issues and need for a fail safe algorithm to ensure successful termination of the meshing process. Failure to complete the meshing process typically relates to the inability to mesh one or more small sub-domains which we refer to as cavities. This problem has been studied by Hecht, George and Saltel [6]. They have proposed a method which requires the heuristic insertion of interior vertices (Steiner points) if the recovery of the faces can not be succeeded with local edge and faces swaps. Their work insists on recovering the cavity faces without changing the input surface connectivity. However, such procedures which rely on heuristic interior point placement have not yet proven to work in all cases. In this paper, an algorithm is presented which is robust and successful in meshing a cavity region such that it can always be connected to an exterior mesh.

The method presented here to mesh the cavities begins with the same base mesh generation method [6], i.e., Delaunay point insertions, due to its ability to easily mesh convex domains. The convex Delaunay triangulation of a bounding box that includes the cavity may not represent all the cavity boundary faces. The recovery of missing cavity boundary faces inside the Delaunay triangulation of the bounding box is the most crucial step of the algorithm. The procedure presented in this paper takes an alternative viewpoint of how to produce a valid set of tetrahedra within the cavity. In this approach, the definition of maintaining the required portions of the cavity boundary is relaxed to allow specific split operations based on the geometric intersections between the Delaunay bounding box mesh of the cavity and the missing cavity boundary faces. As is shown, adding this freedom to the problem specification allows us to always complete the tetrahedronization of the cavity. It is also shown that a controlled implementation of local mesh modifications [7], edge/face splits, edge collapse, edge swap and a new set of "complex splits" tool makes it possible to successfully complete the meshing operations. The required split operations are always possible and can be applied on both sides of the cavity geometry thus allowing the cavity mesh to be matched with the volume mesh in the remainder of the domain. Given that the fundamental goal of the cavity meshing algorithm is to be able to complete the triangulation of the entire domain, the local mesh modifications are performed in such a manner that each original surface triangulation is represented either as it was originally defined, or as a concatenation of other triangles. An overview of the method is described in Section 2 which covers the basic steps of the face recovery algorithm. Section 3 discusses the Delaunay meshing of the cavity bounding box. The face recovery algorithm is explained in Section 4. The results and

future goals are discussed in Section 5.

2 Overview of the method

The basic steps of the method are itemized below with the help of a 2D example shown in Figure 1. Although the algorithm would not be used in 2D, the example here is given to ease the explanation of the overall steps of the face (edge in the example) recovery algorithm. In Figure 1(a), it is shown a mesh, which will be referred heretofore as original volume mesh with an unmeshed cavity region. The goal is to have a volume mesh of the cavity geometry which represents the geometry of original cavity surface triangulation (triangular faces in 3D). Furthermore the cavity mesh must be matched with the outside volume mesh to complete the meshing process. The basic steps, that will be discussed in detail in Section 4, are as follows:

1. The cavity geometry is isolated and placed in a bounding box and a Delaunay mesh of the bounding box including the vertices of the cavity is generated as discussed in Section 3. In the Delaunay mesh of the bounding box of cavity, the original surface triangulation of the cavity geometry may not all be present, as depicted in Figure 1(b). There could be missing faces and missing edges of the cavity surface in the 3D case (only edges in 2D).

2. One way to recover the geometry of the original cavity is to perform specific split operations at the intersections of Delaunay mesh entities and missing cavity entities as described in Section 4.1. The volume mesh of the cavity will be referred heretofore simply as cavity mesh. A single edge split operation on cavity mesh is needed to recover a missing edge as depicted in Figure 1(c). There is actually one more step which uses collapse operations to represent each missing face with least number of faces in the cavity mesh. However, the collapse step can not be seen in the simple 2D example of Figure 1. The collapses are discussed in detail in Section 4.3.

3. After the recovery of the surface triangulation (includes the geometry of the original cavity triangulation in terms of face concatenations) via the application of split and collapse operations, the tetrahedra of the cavity mesh outside the cavity are deleted as depicted in Figure 1(d). The deletion requires the classification of cavity mesh entities which is discussed in Section 4.2.

4. To complete the meshing process, cavity mesh must be merged into the volume mesh. A successful merge operation requires an identical match between the

cavity boundary of the original volume mesh and the cavity mesh. Two different approaches are developed for this purpose; normal splits followed by planar edge swaps and "complex splits" which are performed on the original volume mesh and explained in Section 4.5 and Section 4.6, respectively. It will be shown that the second approach is a guaranteed way to match up the boundaries for merging process. This is shown in Figure 1(e).

5. Finally, the cavity mesh and the volume mesh can be merged, resulting in a valid mesh of the entire domain as depicted in Figure 1(f). Topological operations needed in merging process is explained in Section 4.7.

Although the high level steps of the algorithm in three-dimensions are the same, the operations needed to obtain the geometry of the cavity and split the volume mesh outside the cavity to match the cavity mesh boundary such that a valid mesh is obtained requires careful consideration. A more detailed description of the above steps is given in Section 4.

3 Delaunay Vertex Insertions

The boundary vertices of the Cavity are inserted into a bounding box by a robust incremental Delaunay vertex insertion method [3]. The bounding box of the cavity is found and enlarged cubically by a factor of 1.5. The vertices at the corners of the bounding box are marked to help in deleting the undesired outside regions later in the algorithm (See Section 4.2). The initial bounding box triangulation is based on a six tetrahedra template. Each vertex of the cavity boundary is inserted into the convex bounding box triangulation by first finding the tetrahedron containing the vertex. The vertex is localized by traveling through face (triangle) connected regions (tetrahedra). To speed up the searching process, the bounding box is divided into a rectangular prism grid or bins [2]. This division is saved in a separate data structure where each 3D rectangular cell contains a vertex. Initially, none of the bins contain a vertex. However, as the vertices are inserted, by simply mapping the coordinate of the vertex into the integer indices of the bin structure, the bin of the vertex can be found [2]. If there is already a vertex stored in the bin, one of the connected regions of the vertex are taken as the seed for the searching process to find where exactly the new vertex is localized. From this region, the regions whose circumspheres enclosing the vertex are found [1]. Before the creation of Delaunay cavity from these regions, the Delaunay cavity should be ensured to be point convex with respect to the vertex. The faces of Delaunay cavity which would form invalid regions are deleted from the

cavity by shrinking the cavity. If there exists another vertex inside the cavity other than the vertex to be inserted then one of the regions connected to this extra vertex is also deleted from the cavity and the emptiness of the Delaunay cavity is verified [8]. The entities in the Delaunay cavity are deleted and the vertex is connected to each Delaunay cavity face to form a region. The Delaunay cavity faces should all be so oriented that the face normals would show interior or outside of the cavity which is also a requirement for a consistent convexity check and for the generation of a valid set of mesh regions. The bounding box meshing of two example cavities by Delaunay vertex insertions are depicted in Figure 2 and Figure 3.

The initial bounding box enlargement to initiate Delaunay vertex insertions affects the number of missing faces that must be recovered. There is however no simple selection of the bounding box enlargement ratio which will result in the minimization of the missing faces. The optimum ratio changes with the shape of cavity and distribution of its vertices. The success of the entire algorithm also depends on the successful implementation of Delaunay insertions of the cavity boundary vertices. Although many Delaunay triangulation algorithms are published in the literature, a very few of them respects the validity check of the Delaunay reconnections [3, 8]. In this study, the visibility and the emptiness of the Delaunay cavity is checked and modified if it is needed by shrinking the cavity before the reconnection process. These checks are crucial for the robustness of Delaunay triangulation. Correction of Delaunay cavity is required to prevent the formation of invalid regions which may result from the numerical sensitivity of Delaunay triangulations [8]. The Delaunay cavity shown in Figure 4(a) has all of its vertices on a sphere. The cavity is formed from the regions whose circumspheres (the circumspheres are all the same in this case) contain the vertex to be inserted at the center of the sphere. Some of the cavity faces are not visible to this vertex. The reconnection process may result in invalid regions if the cavity is not made visible to the vertex. Therefore, the visibility is conserved by shrinking the cavity as depicted in Figure 4(b).

In this study, we have decided to use the perpendicular distance of any vertex of the tetrahedron to its opposite face for our validity and visibility check (distance can be signed). If the perpendicular distance is less than a preset tolerance value which is made to adjust itself with the changing model scales, then the region is counted to be invalid. This choice of validity has also been used in every local mesh modification operation to preserve the consistency of the successive operations. Also, we note that randomizing the order of vertex insertions speeds up the Delaunay meshing by a factor of 1.5-2.5.

4 Face Recovery Algorithm

The original cavity faces may not all be present in the Delaunay mesh of the bounding box of the cavity and thus must be recovered to have a triangulation of the cavity geometry. The faces which do not appear in Delaunay triangulation is called missing faces. The face recovery algorithm is applied for missing faces and summarized below. The items below are also depicted schematically for a missing face in Figure 5.

1. Edge and face splits on the cavity mesh are performed to represent a missing face by a group of edge and face sets which has links to the missing face. The Delaunay property of the cavity mesh is no longer applicable after the first split operation. The splits are performed at the intersection locations of the missing faces on the cavity mesh with mesh edges and faces in the Delaunay mesh of the cavity bounding box. This step is applied for each missing face as explained in Section 4.1.

2. The missing faces are now representable by a collection of groups of edge and face sets in the cavity mesh. When this collection is matched with the original set of cavity faces (including non-missing, embedded and non-manifold faces), the cavity mesh is classified as explained in Section 4.2.

3. The region between the cavity interior and its enlarged bounding box boundary is now separable. The mesh regions outside of the cavity interior is deleted from the cavity mesh.

4. Edge collapses are performed on the cavity mesh to represent a missing face by minimum number of edge and face sets (the ideal is to have only three edges and one face for each). The collapses are confined inside the boundary of each missing face as explained in Section 4.3.

5. If the edge collapses have not yet eliminated all the extra faces and edges of the original surface triangulation, the original mesh outside the cavity must be modified to be able to merge it with the cavity mesh. We have developed two different algorithms for this purpose. The first, algorithm 1, as is explained below, using normal splits followed by edge swaps, produces better meshes than the second algorithm but is not provably robust if the missing cavity faces are connected to interior regions where the swap may not always be possible. The second one, algorithm 2, using "complex splits", which may change the connectivity of the neighboring interior regions on the other side of the missing cavity faces, will always work and is used in the current software implementation of the face recovery algorithm.

a. Algorithm 1:

   1. The extra vertices of the cavity mesh are inserted onto the original volume mesh (original mesh with unmeshed cavity) by edge and face splits as discussed in Section 4.4.
   2. Although the split locations are the same, the resulting splits on the boundary faces of the cavity and the volume mesh would be different since the existing tetrahedra on either side of the cavity boundary are not the same. Therefore, the boundary of the two meshes; the cavity and the volume may not necessarily match. However, the only difference is the edge orientations inside the original planar boundary of each missing face represented by both meshes. Therefore, planar edge swaps are attempted on the boundary of the volume mesh to have a matching boundary triangulation. If the edge to be swapped is connected to a set of tetrahedra whose swap configuration is not possible due to geometrical constraints of a general 3D swap, this procedure can not be used to complete the process of matching the mesh. For the cavities whose boundary is not a part of a pre-triangulated volume mesh, this process terminates since it's been proven that edge recovery by swaps is guaranteed on planar faces [6, 9] and face recovery on planar facets are identical to its recovery of edges as explained in Section 4.5.

b. Algorithm 2: In the "complex split" operation, the extra mesh entities of the cavity mesh on the missing face of the original cavity triangulation are copied onto the corresponding face of the volume mesh outside the cavity without carrying the split processes to the higher order entities. For instance, two new edges are created per split location on an edge without connecting the split point to the neighboring faces. The face splits are also treated in the same manner without affecting the neighboring regions. This type of reconnection creates an invalid mesh temporarily. The invalidities are resolved by carrying out a reconnection process over the higher order entities using a convex triangulation algorithm which is discussed in Section 4.6. The final boundary of the volume mesh is guaranteed to be conforming with cavity mesh after "complex splits" since the operations are biased on maintaining an exact copy of the target configuration on the boundary of the cavity mesh.

6. The cavity mesh can now be merged inside the volume mesh thru its cavity boundary classification information. Since the boundary of the cavity and the volume meshes are identical, the stitching of cavity mesh to the master volume

mesh is topologically possible as explained in Section 4.7.

4.1 Splits on Cavity Mesh

The intersection sets between the mesh entities of the bounding box i.e., cavity mesh and the missing faces of the original cavity surface triangulation are computed. First of all, the missing edges of the missing faces are intersected with the cavity mesh faces in the neighborhood of each missing edge. The mesh faces can be intersected by these missing edges in their face interiors or along their edges. Depending upon the location of the intersection point, either an edge or a face split operation is done over the cavity mesh. This process is repeated for every missing face. At the end, the missing edges are represented by a collection of sets of edges on the cavity mesh. In other words, missing edges are recovered by a series of edges on the cavity mesh. Each edge set has a link to its missing edge in the cavity mesh. This process is shown in Figure 6.

It is important to note that the recovery of edges is not enough to recover the face. Therefore, face-edge intersections between the missing face and the edges of cavity mesh are computed, respectively. Edge splits are performed at the intersection locations on the cavity mesh. This process is repeated for every missing face as depicted in Figure 7. At the end, each missing face is represented by a set of faces on the cavity mesh. Each split vertex v_{Fi} on the cavity mesh has links with its master missing face F_i in the original surface triangulation. Each vertex of the original cavity surface triangulation, V_i is also linked with a vertex v_i in the cavity mesh. Similarly, each edge of a missing face, E_i is linked with one or more edges, e_{ij} of the cavity mesh. These links can be seen in Figure 8. The splits on cavity mesh are shown for two example cavities in Figure 9 and Figure 10.

The above algorithm is given in pseudo-code form as below:

Input: Missing faces list $MFaces$
For each missing face M_j^2 of $MFaces$ **do**
 For each edge M_i^1 of M_j^2 **do**
  If an edgelist has already been attached to M_i^1 **Then**
   Skip to another missing edge M_i^1.
  Get the cavity mesh region Set R around the first vertex M_i^0 of M_i^j.
  For each region M_k^3 of Set R **do**
   For each face M_n^2 of M_k^3 **do**
    Find Edge(M_i^1)-Face(M_n^2) intersection.
    If there is an intersection **Then**
     Locate intersection point position on the face M_n^2.
     If location is on the edge E of the face M_n^2 **Then**

```
      Split the Edge E and save the resulting vertex V.
    Else
    If location is in the face M_n^2 Then
      Split the Face M_n^2 and save the resulting vertex V.
    Attach the edge E1 composed of vertices M_i^0 and V
    to the missing edge M_i^1.
    Replace M_i^0 with V.
  EndIf
  Else
   No intersection is found; that means the edge is recovered.
   Skip to another missing edge M_i^1.
  EndElse
 EndDo face M_n^2 loop
EndDo R loop
EndDo missing edge M_i^1 loop
Attach all edge sets which have links to the edges of
the missing face to the misssing face itself. Call this list S.
For each cavity edge M_i^1 in S do
 Get the cavity mesh region set R around M_i^1.
 For each region M_k^3 of set R do
  For each edge M_n^1 of M_k^3 do
   If M_n^1 is an element of S Then
    Skip to another edge M_n^1.
   Find Edge M_n^1-Missing Face M_j^2 intersection.
   If there is an intersection Then
    Split the edge M_n^1 and save the resulting vertex V.
    Append the new edges set N around vertex V to S.
    Attach V to the missing face M_j^2.
   EndIf
  EndDo edge M_n^1 loop
 EndDo R loop
EndDo S loop
EndDo Missing faces loop
```

The robustness of the above algorithm depends on an accurate intersection scheme coupled with a consistent intersection point location procedure. If the edge-face intersection routine returns the intersection point which should lie inside the boundary of the face, the task of the point location procedure is to determine where exactly the point is falling inside the face; over one of its edges or in itself. Notice that intersection scheme is designed to discard touching type intersections. It only respects thru type intersections which can be seen in Figures 6 and 7. The result of point location procedure is used to decide the type of split operation; either an edge or a face split.

4.2 Deletion of Outside

The cavity mesh has a signature of the original cavity surface triangulation now. The missing cavity mesh faces are now represented by sets of face collections where the links between these sets and cavity missing mesh entities are established by the

previous split algorithm. Therefore, the region remaining between the cavity mesh boundary and the outside fictious bounding box vertices can be deleted from the Delaunay mesh. This deletion process requires the complete classification of the cavity mesh. The cavity mesh entities are classified as outside but treated as if they were interior to enable the split operations. The deletion of outside regions has three algortihmic stages;

1. Setting the classification of boundary faces on the cavity mesh on the basis of original cavity surface mesh classification. The outline of missing faces is found on the cavity mesh by the help of inter-entity links established in the previous split stage in terms of sets of face collections. This is given in pseudo-code below.

2. Setting the classification of mesh regions of the cavity mesh by walking thru face connected regions and switching the classification when an boundary mesh face is crossed. The embedded faces are treated specially. This part of the algorithm is a default for any Delaunay type triangulation scheme [10].

3. The outside entities are deleted from the cavity mesh by checking classification information. Top-down deletion is required for a hierarchical mesh database in which there is up and down inheritance in vertex-edge-face-region entity links [11].

The deletion of outside regions from the cavity mesh is shown for two example cavities in Figure 11 and Figure 12.

The algorithm of the first stage can be given in pseudo-code form as below:

Input: Original cavity boundary mesh faces as *Faces*
For each cavity boundary face M_c^2 from faces set *Faces* **do**
Get the vertex set V_c of M_c^2.
Get the corresponding vertex set V_d in bounding box mesh of the cavity via links.
Check if there exists a face with V_d.
If face M_d^2 exists **Then**
Set the classification of M_d^2 as of M_c^2.
For each vertex M_d^0 of V_d **do**
Get the corres. vertex M_c^0 of V_c.
Set the classification of M_d^0 as of M_c^0.
EndDo
For each edge M_c^1 of M_c^2 **do**
Get the corersponding edge M_d^1 from M_c^1's vertex links.
Set the classification of M_d^1 as of M_c^1.
EndDo
EndIf

```
Else
  For each edge M_c^1 of M_c^2 do
    Get attached edges (its representation on the cavity mesh)E_d
    of M_c^1.
    For each attached edge M_d^1 of E_d do
      If there exists no classification of M_d^1 Then
       Set the classification of M_d^1 as of M_c^1.
      For each vertex M_d^0 of M_d^1 do
        Append the vertex in a vertex list V_f.
        If M_d^0 has no classification Then
          Get the cavity vertex M_c^0 linked with M_d^0.
          Set the classif. of M_d^0 as of M_c^0.
        EndIf
      EndDo
    EndDo
  EndDo
  Append attached vertex list of M_c^2 to V_f.
  For each vertex M_d^0 of V_f do
   If M_d^0 has no classif. Then
    Set the classif. of M_d^0 as of M_j^2.
  For each vertex M_d^0 of V_f do
    Get the faces list F around M_d^0.
    For each face M_d^2 of F do
      If all the vertices of M_d^2 is in V_f Then
        Set the classification of M_d^2 as of M_c^2.
        For each edge M_d^1 of M_d^2 do
          If M_d^1 has no classif. Then
            Set the classif. of M_d^1 as of M_c^2.
      EndIf
    EndDo
  EndDo
EndElse
EndDo.
```

4.3 Applying Edge Collapses on Cavity Mesh

The primary goal of face recovery algorithm is to preserve the original cavity surface triangulation with minimum additional vertex insertions. In other words, each missing face is represented on the cavity mesh by the minimum number of face concatenations. At this level of the algorithm, a missing face is represented in the cavity mesh such that there is a set of faces whose union represents its boundary by means of edge and vertex links as depicted in Figure 8. There are two sets of links for each missing face; edge lists for each of its missing edges, and a vertex list for the missing face itself. The collapse operation is a planar edge collapse confined within the boundary of each missing edge and face such that the boundary of the original surface triangulation is never violated.

First, collapses are attempted on the vertices having links with the missing face along

one of its connected edges which should also be on the missing face. To increase the chance of collapse operation, every candidate edge is checked for a possible collapse. This is depicted in Figure 13.(a). Second, the collapses are attempted on the edges having links with the missing edges along one of its connected edges which should also be on the missing edge. Again, every possible configuration is tried to increase the chance of collapse operation. This is depicted in Figure 13.(b). Edge collapse operations on two example cavities are also shown in Figure 14 and Figure 15 where the number of additional vertices are reduced significantly.

The above algorithm can be given in pseudo-code form as below:

```
Input: Missing faces list MFaces
For each missing face M_j^2 of MFaces do
  Get the cavity mesh vertices V_d1 linked with M_j^2.
  Get the cavity mesh vertices of the edges, V_d2 linked with
  the edges of M_j^2.
  Form the list FVerts by appending two lists V_d1 and V_d2.
  Set MaxCount to the number of vertices in V_d1 list.
  Set Count to 0.
 Repeat
  Increment Count by 1.
  For each vertex M_d^0 of V_d1 do
    Get the connected edges list E to M_d^0.
   For each edge M_d^1 of E do
     If the other vertex of M_d^1 is in FVerts Then
       Check geometrical and topological validity of edge collapse
       of M_d^1.
       If collapse is valid Then
         Remove the vertex M_d^0 from FVerts and V_d1.
         Apply Edge Collapse; Collapse M_d^1 by deleting M_d^0.
       EndIf
     EndIf
   EndDo
  EndDo
 Until the size of V_d1 equals 0 or Count exceeds MaxCount
 Modify the link V_d1 to M_j^2.
 Delete FVerts.
EndDo
For each missing face M_j^2 of MFaces do
  For each missing edge M_k^1 of M_j^2 do
   Get the cavity mesh edges list E_d linked with M_k^1.
   Constrain the cavity end vertices of edge M_k^1 from
   being collapsed.
   Repeat
     Set Collapse Flag to 0.
    For each cavity mesh edge M_i^1 of E_d do
     For each possible collapse configuration of M_i^1 do
       Try a constrained collapse of M_i^1 over one of
       the connected E_d edges.
       Check geometrical and topological validity of the collapse.
       If collapse is valid Then
         Remove the edge from the list E_d.
         Collapse the edge M_i^1.
```

```
        Append new edges appearing over M_k^1 to the
        list E_d.
        Set Collapse Flag to 1.
      EndIf
    EndDo
  EndDo
 Until Collapse Flag equals 0.
EndDo
EndDo
```

4.4 Splits on the Original Volume Mesh

If the algorithm 1 of Section 4 is chosen then the vertices that can not be collapsed in Section 4.3 are inserted on the original cavity surface of the volume mesh by edge and face splits. The vertices can be accessed by using the mesh entity links that are created in the split stage of the cavity mesh. For each missing edge E_i, there is a list of cavity mesh edges e_{ij} and for each missing face F_i, there is an attached cavity vertex list v_{Fi}. Also for each vertex of the original surface triangulation V_i, there is a corresponding cavity mesh vertex v_i (Fig.8). First of all, the vertices v_{Fi} are localized on the original cavity boundary of the volume mesh. Again the vertex localization routine must be robust for the correct decision of the type of split operation; either edge or face. Second, the edge vertices of the edge list e_{ij} are inserted onto the volume mesh by means of all edge splits since the vertices are always on an existing edge. The result of split operations on the cavity boundary of the volume mesh can be envisaged in Figure 16(a) and Figure 17(a). It can be seen from the figures that although the split locations are the same, the resulting splits on the boundary faces of cavity and the volume mesh would be different since the existing tetrahedra on the volume mesh and the cavity mesh sides of the cavity boundary are not the same. Also, to emphasize the difference between the splits on the cavity and the volume mesh, the collapse operations are skipped in Figure 16(a) and Figure 17(a).

4.5 Planar Swaps on the Volume Mesh

In the previous stages of face recovery algorithm, the split and oriented collapse operations are applied on the cavity mesh. The vertices that can not be collapsed are used to perform splits on the original cavity boundary of the volume mesh. As seen from Figure 16 and Figure 17, the splits on cavity mesh on the cavity boundary of the volume mesh do not have the same pattern since the regions (tetrahedra) connected to the cavity boundary are not the same on both meshes. However, the original

missing face pattern must ultimately be conserved in both meshes. Therefore, to make matching boundaries, it would be enough to swap the non-matching boundary edges of the volume mesh on the planar facets of the original missing faces. In other words, recovering the non-matching edges thru planar edge swaps on each missing face will make the face recovery process possible. If the swap operation is a 2D planar swap, it is proven that the recovery of edges is guaranteed thru successive edge swaps [6, 9]. In practice, the cavity boundary can be connected to interior regions of the volume mesh where the intended 2D swap on the missing face is actually a 3D one which may not always be possible due to geometrical constraints imposed on the 3D swap operation. Nevertheless, the algorithm shows great success in many cases, though a chance of failure always exists if a 3D swap is found to be impossible. To show extensive amount of swap operations, the collapse operation is not carried out for the example cavities depicted in Figure 16(b) and Figure 17(b). By means of planar edge swaps, the boundary of the volume mesh is exactly transformed to a mesh which has identical boundary connectivity of the cavity mesh as shown in Figure 16(a-b) and Figure 17(a-b). The pseudo-code of the algorithm is given below:

```
Input: Boundary mesh faces of the original cavity surface, Faces and
the cavity mesh T_d and the volume mesh T_c.
Find the missing faces mfaces of T_c.
Find the boundary edges list E_c which are not
represented by T_d by visiting each missing
face M_j^2 of mfaces.
Store the size of E_c in NE.
Set Count to 0.
Repeat
Increment Count by 1.
 For each edge M_k^1 of E_c do
   If M_k^1 in T_d Then
     Remove M_k^1 from E_c.
     Skip to another edge M_k^1.
   EndIf
   Find the number of regions NR around M_k^1.
   Determine the type of swap operation Stype ; 2D or 3D.
   Stype=2D if NR =0.
   Stype=3D if NR >0.
   If Stype=2D Then
     If Count > NE Then
       Check swap validity.
       If swap is valid Then
         Carry out swap ONLY if the after swap config.
         results in an edge on T_d.
       EndIf
     EndIf
     Else
       Check swap validity.
       If swap is valid Then
         Carry out swap EVEN if the after swap config. does NOT
         result in an edge on T_d.
       EndIf
```

```
      EndElse
      If swapped edge is not in  T_d Then
        Append the new edge into the list E_c.
    EndIf
    If Stype=3D Then
      If Count > NE Then
        Check swap validity.
        If swap is valid Then
          Carry out swap ONLY if the after swap config.
          results in an edge on T_d.
        EndIf
        Else
          Split the edge ONLY if the after swap config.
          results in an edge on T_d.
          Split point is at the intersection of M_k^1 and
          its after split edge which should be on T_d.
          Perform split on T_d as well.
          Put the resulting edges into the list  E_c.
          Skip to another edge M_k^1.
        EndElse
      EndIf
      Else
        Check swap validity.
        If swap is valid Then
          Carry out swap EVEN if the after swap config. does NOT
          result in an edge on T_d.
        EndIf
      EndElse
      If swapped edge is not in  T_d Then
        Append the new edge into the list E_c.
    EndIf
  EndDo
Until the size of  E_c equals 0.
```

4.6 Complex Splits

The vertices that remain after the collapse operations described in Section 4.3 must be inserted onto the original cavity boundary of the volume mesh. The insertions should be made in such a way that both mesh boundaries are identical to enable the merging of the two meshes. A strategy which allows creating an exact copy of the cavity mesh boundary on the volume mesh is to device the appropriate series of split operations, which we call "complex splits".

The process begins with the boundary edges of the volume mesh being made identical with the boundary edges of the cavity mesh. The mesh-links discussed in Section 4.1 are used to transfer the boundary information between the two meshes. Figure 18(a) shows two shaded missing faces connected to the interior regions that are represented by the surface triangulation drawn below them on the cavity mesh. These missing faces are connected by a missing edge which is actually split on the cavity mesh.

The same split must be done on the corresponding volume mesh edge. However, this split (Figure 18(b)) will create two new edges that are not connected to the touching faces. The shaded faces must also be split as shown in Figure 18(c). Again this split is not a complete split since the touching regions have yet to account for them. The splits have made the cavity boundary of the volume mesh identical with the cavity mesh boundary. However, the volume mesh is invalid since interior entities are not connected to the split vertices. The remaining steps modify the appropriate faces and regions to regain a valid mesh. The interior volume mesh faces connected to the boundary edges which were split are triangulated by inserting a vertex at the centroid of the interior face. The split edge in Figure 18(a) has one interior face. The geometrical center of the interior face is computed and a mid vertex is created as shown in Figure 18(d). The edges surrounding the face including newly created edges are hooked up with this interior vertex by creating a set of new faces all of which is inside the interior face. Regions which are connected to the missing faces and the interior face are still invalid and must be split appropriately using the complex splits. New vertices at the geometrical center of each of the invalid regions are created as depicted in Figure 18(e). The surrounding faces of the regions including the newly created ones are connected to form new interior regions in Figure 18(f). In this study, we have chosen the above convex triangulation methodology which can be proven to be robust and guaranteed to mesh the interiors. The volume mesh is now valid and can be merged with the cavity mesh. To demonstrate extensive amount of complex-split operations, the collapse operations are skipped for the example cavities depicted in Figure 19 and Figure 20. The algorithm is given in pseudo-code form as below:

```
Input: Missing faces list MFaces
For each missing face M_j^2 of MFaces do
 For each edge M_i^1 of M_j^2 do
   Get the cavity mesh edges E_d list attached to  M_i^1.
   For each edge M_k^1 of E_d do
     Apply "complex edge splits".
     Attach newly created volume mesh edges list Redges to M_k^1.
   EndDo
 EndDo
 Get the Interior vertex list V_d attached to M_j^2.
 Find the cavity mesh Face list F_d inside M_j^2 from V_d.
 Apply "complex face splits".
 Attach newly created volume mesh faces list Rfaces to M_j^2.
EndDo
For each missing edge M_i^1 do
 Get the volume mesh edges Redges attached to M_i^1.
 Get the Faces Flist around M_i^1.
 For each face F_c of Flist do
  If F_c is an interior face Then
    For each edge M_j^1 of  F_c do
      Get the volume mesh edges list Redges_j attached to M_j^1.
```

```
      Append Redges_j to Redges.
    EndDo
    Create a vertex V_m at the geometric center of F_c.
    For each edge E_k of Redges do
      Form a new interior face intFace by connecting E_k with V_m.
    EndDo
    Attach new interior faces to F_c as a list of Rfaces.
    Attach F_c to intFaces.
  EndIf
 EndDo
EndDo
For each missing face and interior face M_i^2 of Mfaces and intFaces do
  Get the faces Rfaces attached to M_i^2.
  Get the Regions Rlist around M_i^2 do
  For each region R_c of Rlist do
    For each face M_j^2 of R_c do
      Get the faces list Rfaces_j attached to M_j^2.
      Append Redges_j to Rfaces.
    EndDo
    Create a vertex V_m at the geometric center of R_c.
    For each face R_k of Redges do
      Form a new interior region by connecting R_k with V_m.
    EndDo
    Attach new interior faces to F_c as a list of Rfaces.
    Attach R_c to a list intRegs.
  EndDo
EndDo
For each face M_i^2 of MFaces do
  Get the edges of M_i^2 and attach to Elist.
For each region M_i^3 of intRegs do
  Remove M_i^3 from the Volume mesh.
For each face M_i^2 of intFaces do
  Remove M_i^2 from the Volume mesh.
For each face M_i^2 of MFaces do
  Remove M_i^2 from the Volume mesh.
For each edge M_i^1 of Elist do
  Remove M_i^1 from the Volume mesh.
```

4.7 Merge Cavity

The final boundary of the cavity is guaranteed to be conforming on both sides of the cavity after "complex splits". A mesh merging algorithm which ensures topological compatibility is carried out after "complex splits". The connected regions of boundary faces of both meshes are merged by taking into account that the directions of the faces could be different. The original cavity boundary face directions of the volume mesh are favored and the cavity mesh boundary face directions are modified accordingly. Then the edge, face and region mesh entities of the cavity mesh are copied into the volume mesh. Interior refinement of the cavity mesh can be done before the merging process. Otherwise, there exist no interior vertices inside the cavity. The cavity mesh

is disposed after it is copied and merged inside the master volume mesh. The cavity meshing and merging algorithm works for one model region. This is particularly desired since if the cavity mesher is to be used as a mesh generator to mesh the whole domain of a non-manifold model, it divides the problem of meshing into each model region and generates a triangulation with no interior vertices. On the other hand, it is always possible to refine the interior of the mesh at each model region meshing stage or after the whole domain is meshed.

5 Results and Discussions

The face recovery and cavity merging algorithm has been introduced into a complete mesh generation procedure to deal with the complex cavities. The addition of this procedure has allowed the mesh generator to complete meshes of models which did not previously complete due to the presence of a small number of complex cavities.

The cavity boundary can be extended to the model boundary in which case the cavity mesher is the main mesh generator. Examples of models which are meshed by the cavity mesher are shown in Figure 21, Figure 22 and Figure 23. The examples are all given for the cavities whose boundaries are extended to the actual model boundary. The number of missing original faces in each case and the extra vertex insertions needed to geometrically recover those faces are also depicted in the figures. Note that the meshes are all volume meshes with no interior vertices. The complex split operations are always valid and therefore robust whereas the edge swap method is not a guaranteed approach for general 3D configurations where cavity is a part of a volume mesh.

We have developed a guaranteed cavity meshing algorithm that allows the successful completion of the volume meshing processes in all cases.

References

[1] N. P. Weatherill and O. Hassan. Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints. *Int. J. Numer. Meth. Engng.*, 37:2005–2039, 1994.

[2] S.W. Sloan. A fast algorithm for generating constrained delaunay triangulations. *Int. J. Numer. Meth. Engng.*, 47(3):441–450, 1993.

[3] Houman Borouchaki and Paul Louis George. Aspects of 2-d delaunay mesh generation. *Int. J. Numer. Meth. Engng.*, 40:1957–1975, 1997.

[4] W. J. Schroeder and M. S. Shephard. Geometry-based fully automatic mesh generation and the Delaunay triangulation. *Int. J. Numer. Meth. Engng.*, 26:2503–2515, 1988.

[5] N. A. Golias and T. D. Tsiboukis. An approach to refining three dimensional tetrahedral meshes based on delaunay transformations. *Int. J. Numer. Meth. Engng.*, 37:793–812, 1994.

[6] F.Hecht P.L. George and E. Saltel. Automatic mesh generator with specified boundary. *Comp. Meth. Appl. Mech. Engng.*, 92:269–288, 1991.

[7] H. L. de Cougny and M. S. Shephard. Parallel refinement and coarsening of tetrahedral meshes. *submitted to International Journal of Numerical Methods in Engineering.* Technical Report 21-1995, Scientific Computation Research Center, Rensselaer Polytechnic Institute, Troy, NY 12190-3590.

[8] P.-L. George and H. Borouchaki. *Delaunay Triangulation and Meshing - Application to Finite Elements.* Hermes, 8, quai du Marché-Neuf, 7500, Paris, 1998.

[9] B. K. Karamete, R. Garimella, and M. S. Shephard. Recovery of an arbitrary edge on an existing surface mesh using local mesh modifications. *submitted to International Journal for Numerical Methods in Engineering*, 1998. SCOREC Technical Report #19-1998.

[10] Hugues de Cougny. *Parallel Unstructured Distributed Three Dimensional Mesh Generation.* PhD thesis, Mechanical Eng., Aeronautical Eng., & Mechanics, Rensselaer Polytechnic Institute, Scientific Computation Research Center, RPI, Troy, NY 12180-3590, May 1998. Scientific Computation Research Center, Technical Report # 9-10-1997.

[11] M. W. Beall and M. S. Shephard. A general topology-based mesh data structure. *Int. J. Numer. Meth. Engng.*, 40(19):1573–1596, 1997.

[12] T. J. Hughes C. A. Taylor and C. K. Zarins. Finite element modelling of blood flow in arteries. *Comp. Meth. Appl. Mech. Engng.*, 58(1):155–196, 1998.

Figures

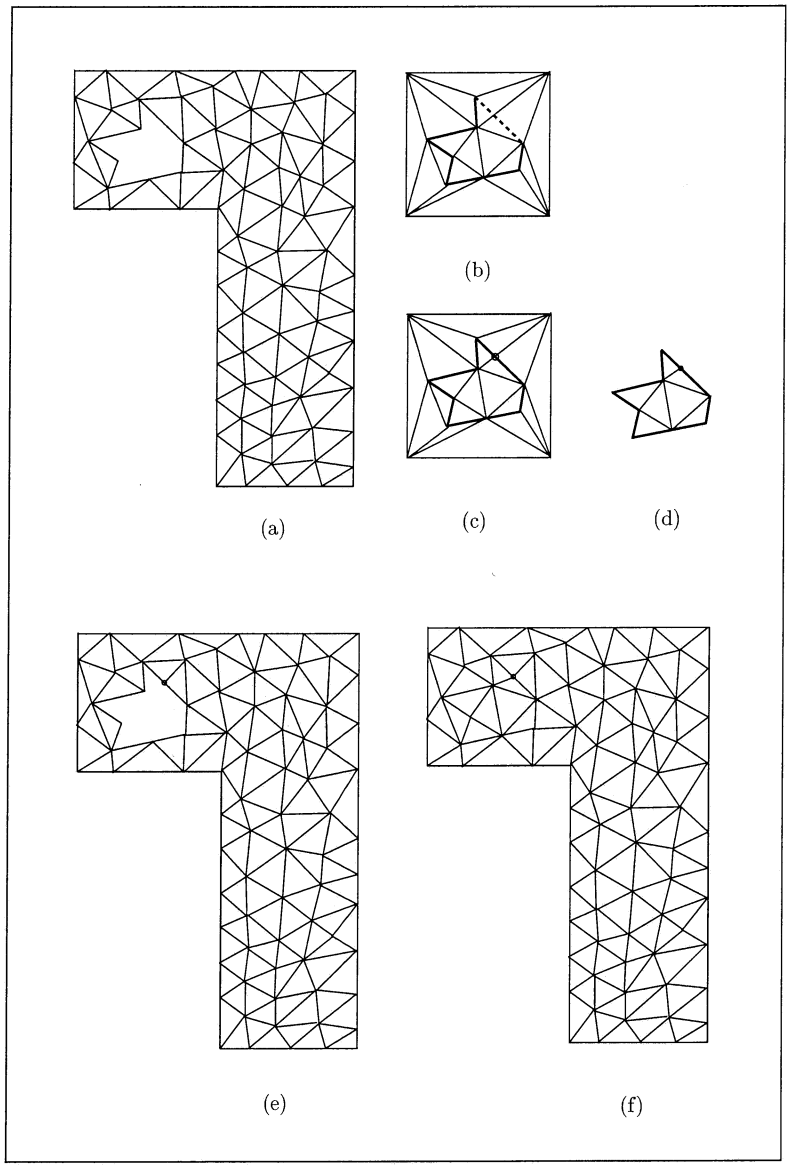

Figure 1: (a)-(f) Basic steps of cavity meshing method. (a) Original triangulation with unmeshed cavity. (b) Bounding box Delaunay mesh of the original cavity vertices. (c) Recovering the missing edges by edge splits. (c) Deletion of the outside and formation of a classified cavity mesh. (d) The required operations (split + planar edge swaps or "complex splits") on the cavity boundary of the original volume mesh is performed. (e) Cavity mesh is merged into the volume mesh.

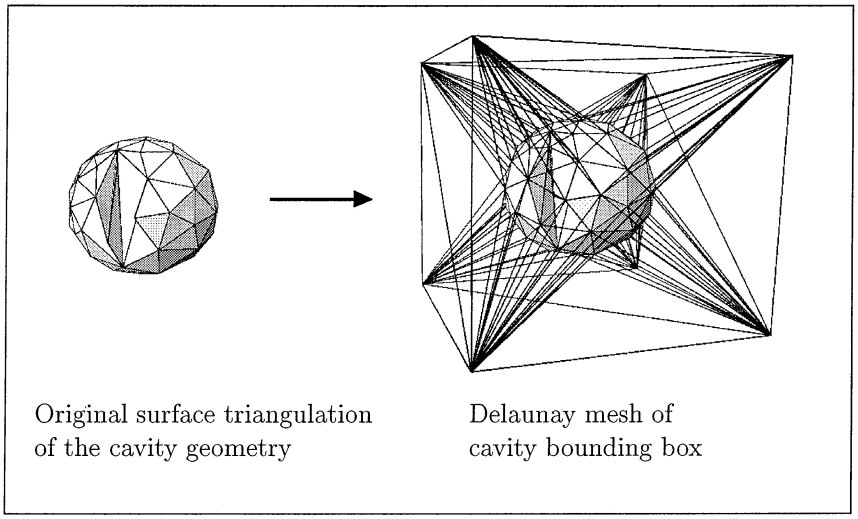

Figure 2: Meshing the bounding box of the cavity by Delaunay vertex insertions.

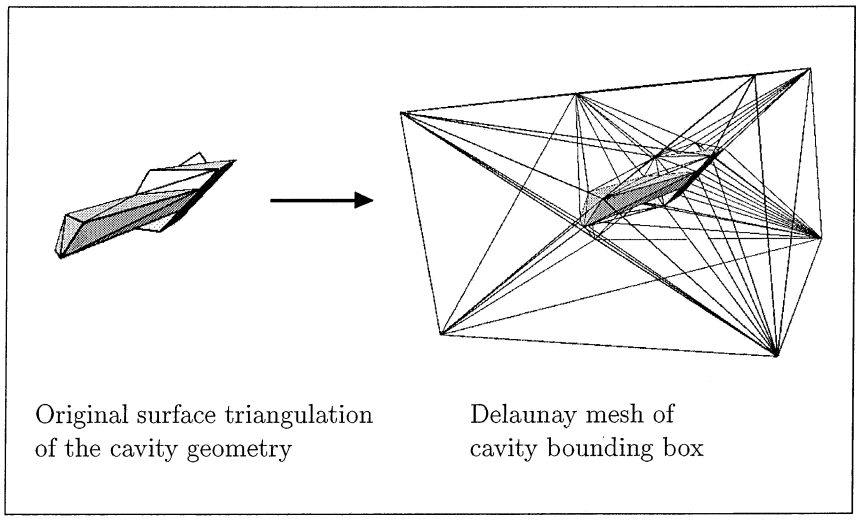

Figure 3: Meshing the bounding box of the cavity by Delaunay vertex insertions.

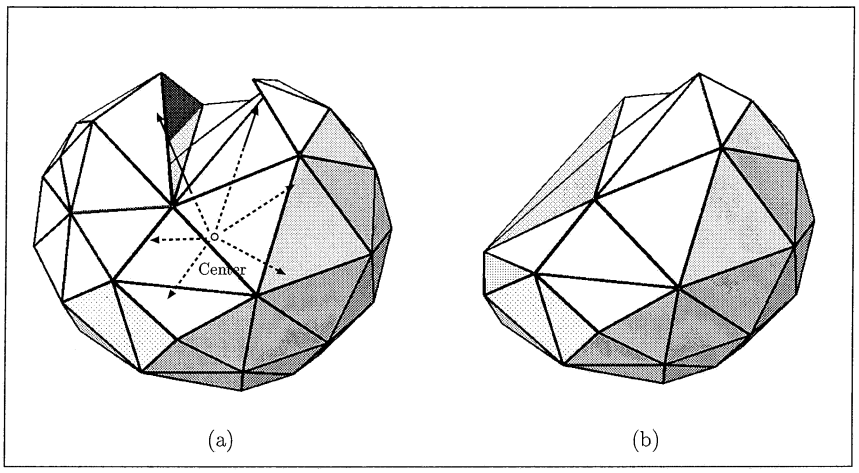

Figure 4: (a) Checking the visibility of Delaunay cavity faces with respect to the vertex at the center. (b) Correcting the visibility by shrinking the cavity to form valid regions in the reconnection process.

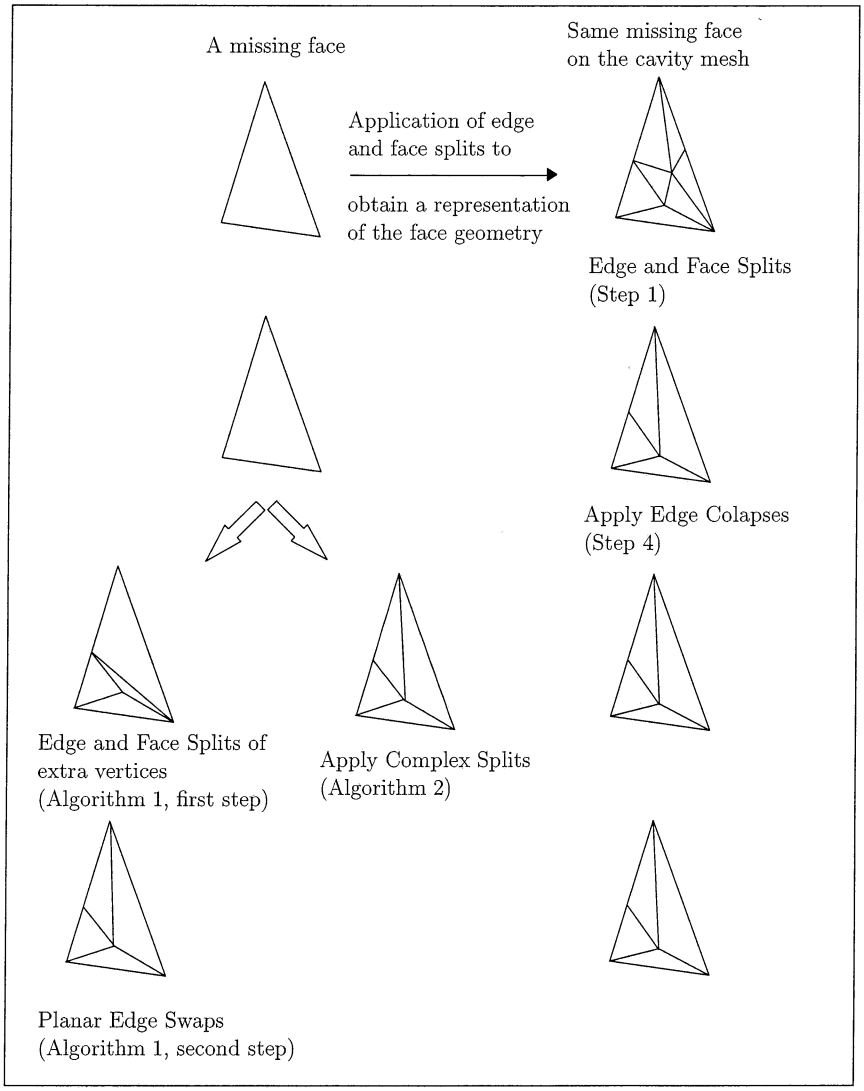

Figure 5: Mesh modification operations needed to recover a missing face on both cavity and original volume meshes.

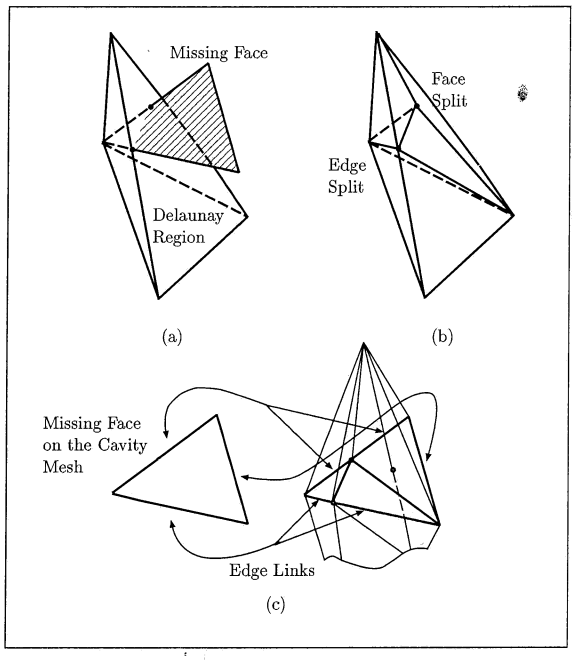

Figure 6: (a), (b) Recovering the missing edges by sets of edges; edge and face splits on the cavity mesh depending on the intersection location between missing face edges and the cavity mesh. (c) Edge links between a missing face and its representation on the cavity mesh.

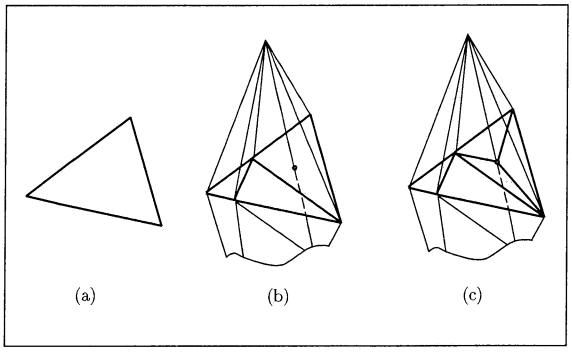

Figure 7: (a) A missing face. (b) Same missing face after the recovery of its missing edges; still the face is intersected. (c) Edge split on cavity mesh to recover the missing face by sets of faces on the cavity mesh.

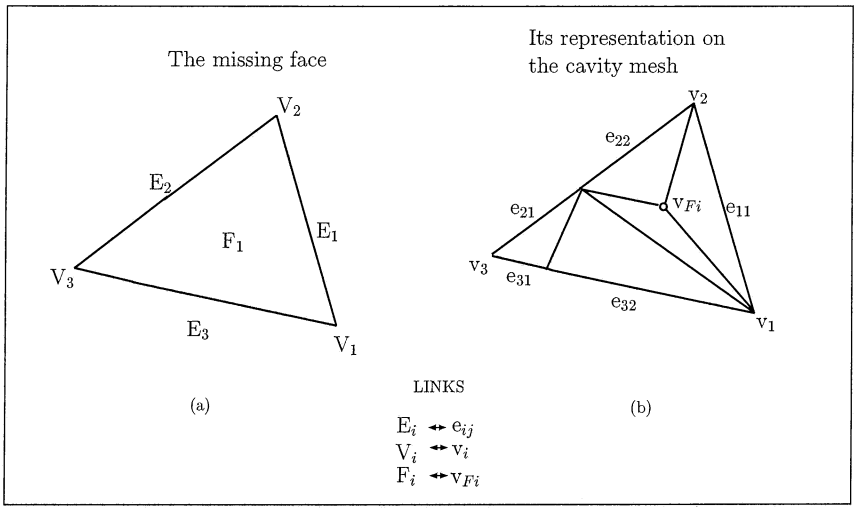

Figure 8: Links between the missing entities and the cavity mesh. (a) A Missing face. (b) Same missing face represented by a set of faces on the cavity mesh.

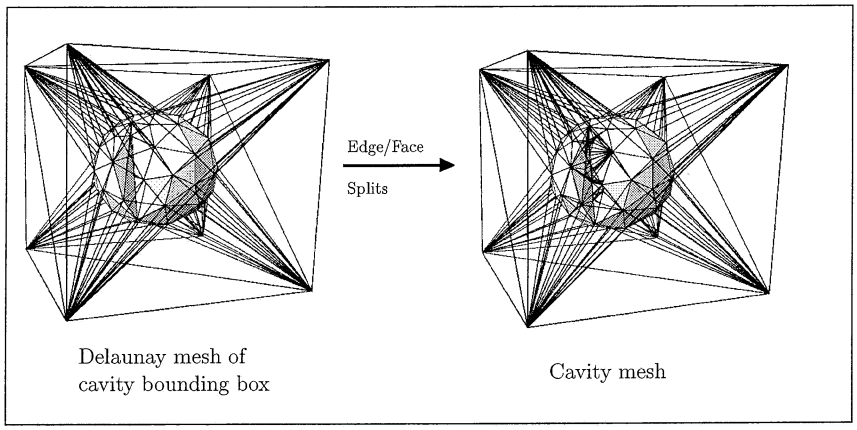

Figure 9: Edge and face splits are performed at the intersections between the bounding box mesh of the cavity and the original surface triangulation of the cavity geometry.

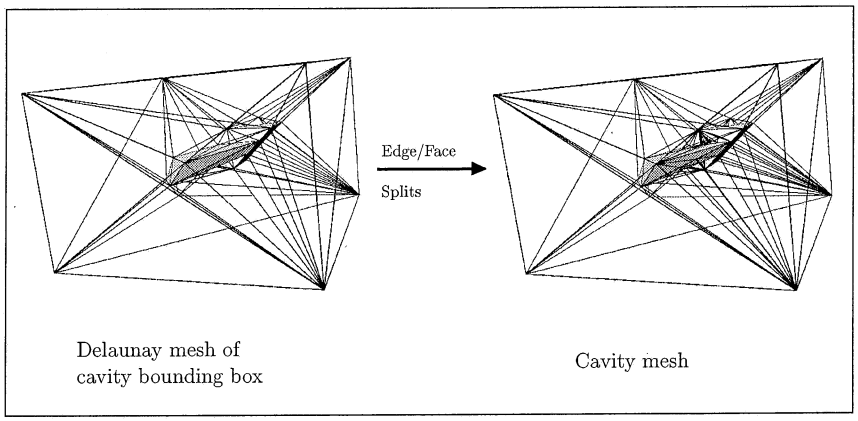

Figure 10: Edge and face splits are performed at the intersections between the bounding box mesh of the cavity and the original surface triangulation of the cavity geometry.

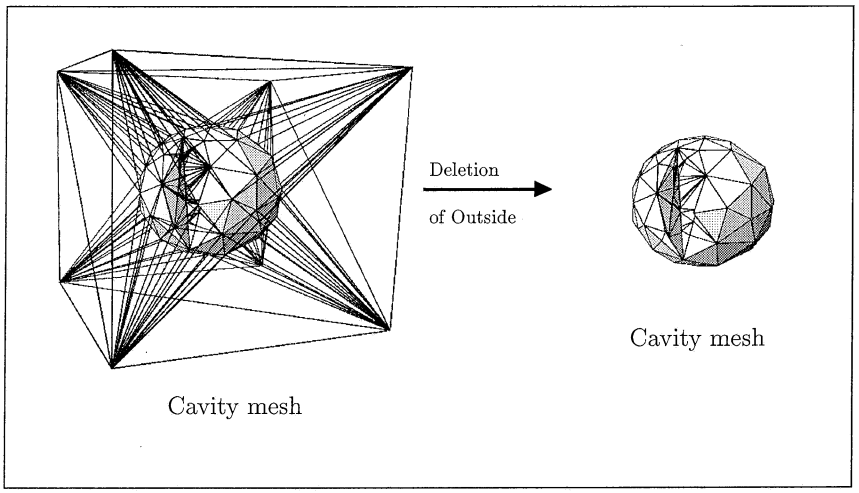

Figure 11: The regions outside the cavity interior are deleted from the cavity mesh.

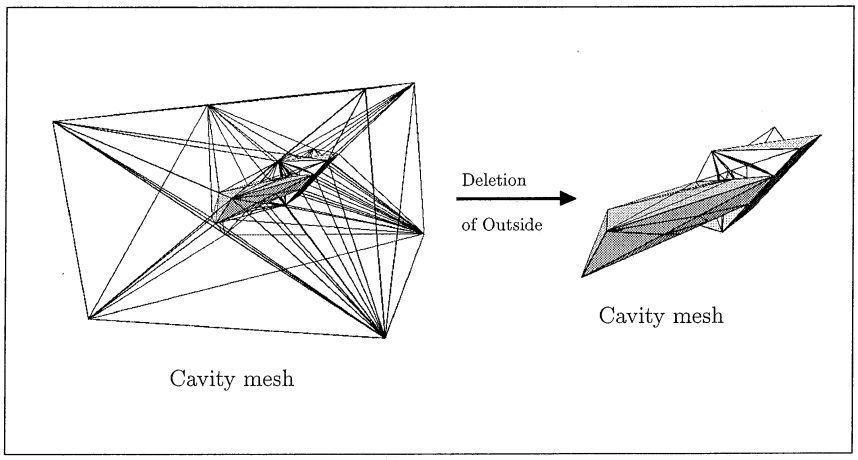

Figure 12: The regions outside the cavity interior are deleted from the cavity mesh.

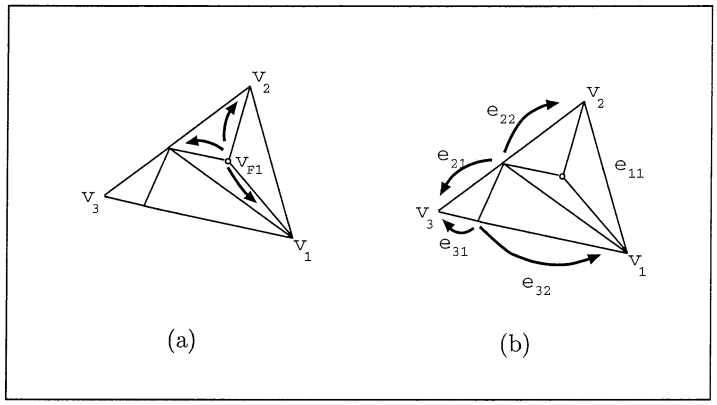

Figure 13: Possible collapse situations (a). for interior split vertices (b). for split edge vertices on the Delaunay mesh.

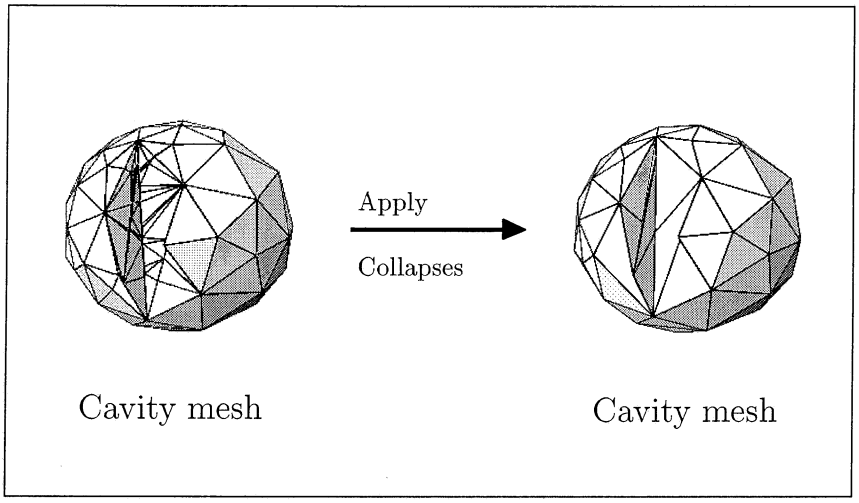

Figure 14: The number of extra vertices are reduced by planar edge collapses on the cavity mesh.

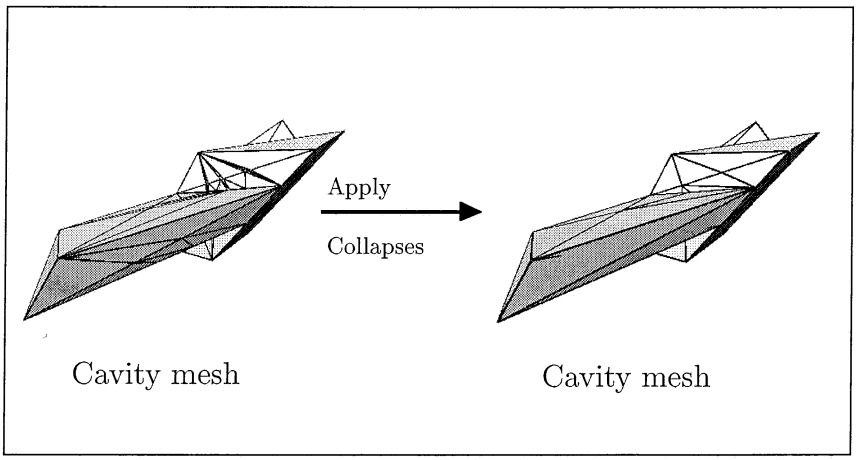

Figure 15: The number of extra vertices are reduced by planar edge collapses on the cavity mesh.

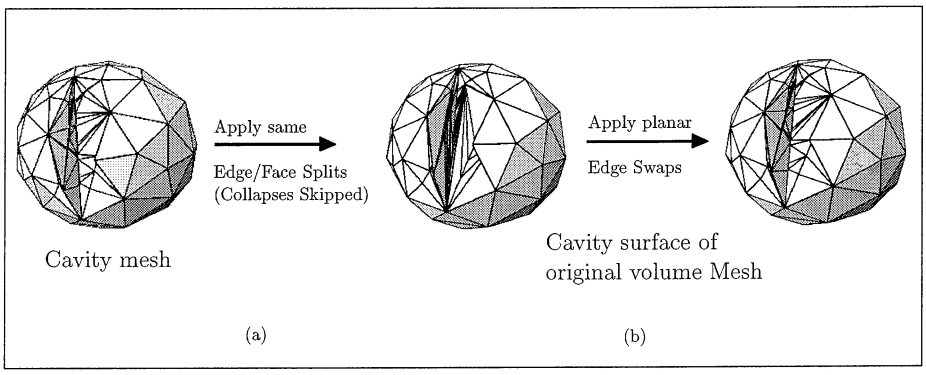

Figure 16: Algorithm 1. (a) Edge and face splits are applied on the original cavity boundary of the volume mesh by using the same split locations on the cavity mesh. (b) The edges are swapped over the planar missing faces to make an identical boundary on both cavity and volume meshes.

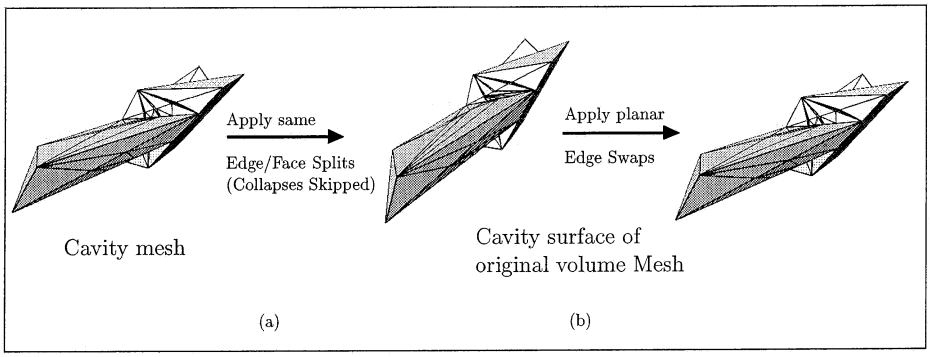

Figure 17: Algorithm 1. (a) Edge and face splits are applied on the original cavity boundary of the volume mesh by using the same split locations on the cavity mesh. (b) The edges are swapped over the planar missing faces to make an identical boundary on both cavity and volume meshes.

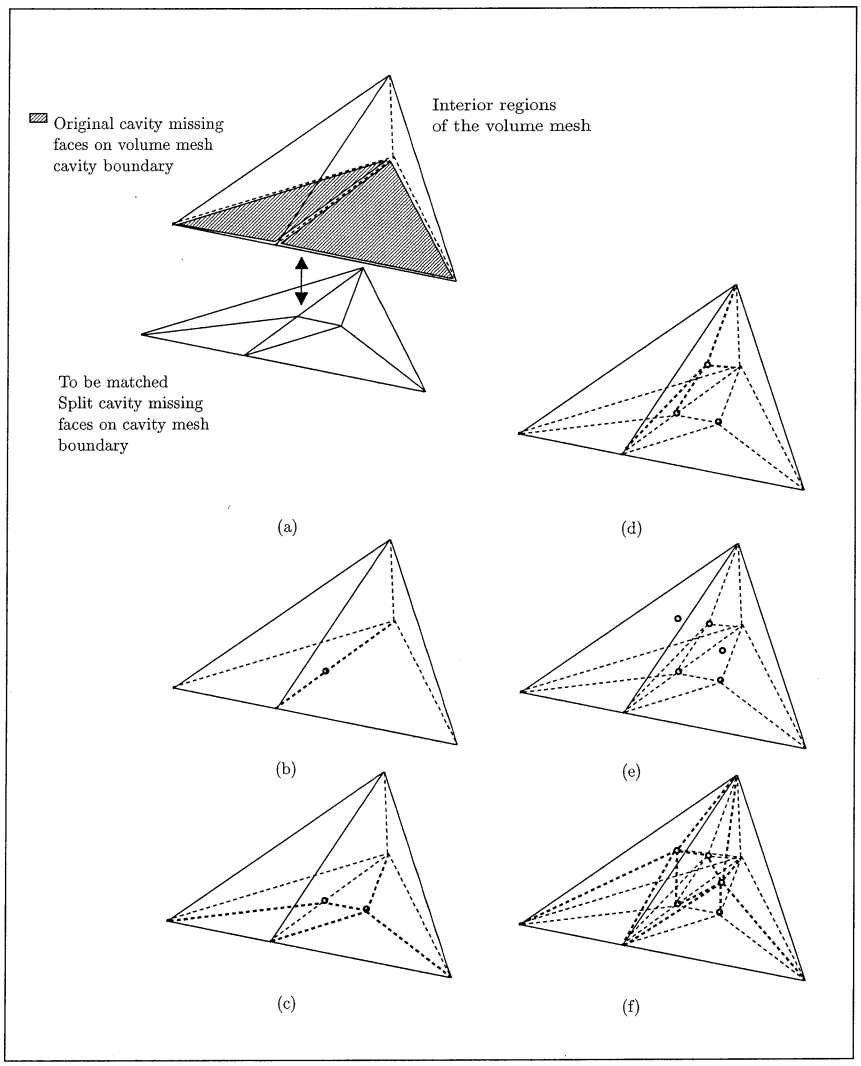

Figure 18: Complex Splits:(a) Two missing cavity faces (shaded), (b) Complex edge split, (c) Complex face split, (d) interior face triangulation, (e-f) interior region triangulation.

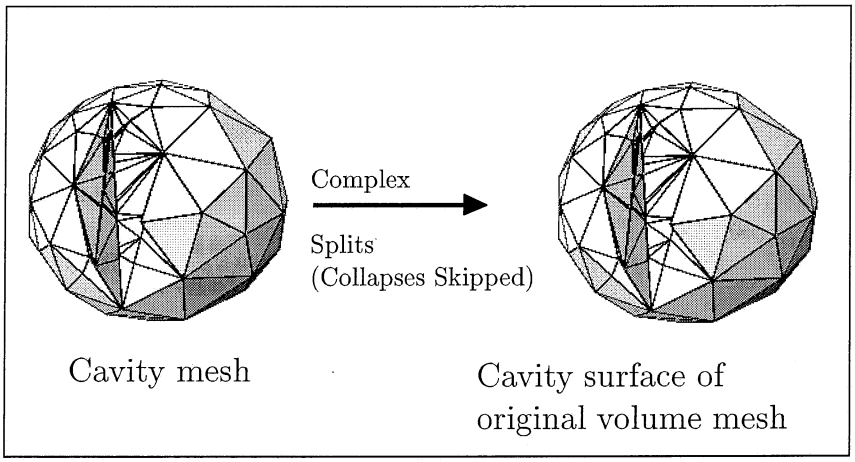

Figure 19: Algorithm 2. Complex-Splits are applied to obtain the identical connectivity between the cavity mesh and the cavity boundary of the original volume mesh.

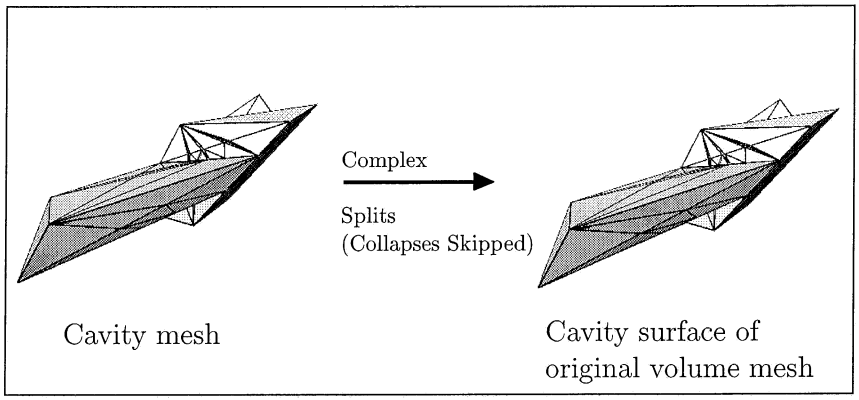

Figure 20: Algorithm 2. Complex-Splits are applied to obtain the identical connectivity between the cavity mesh and the cavity boundary of the original volume mesh.

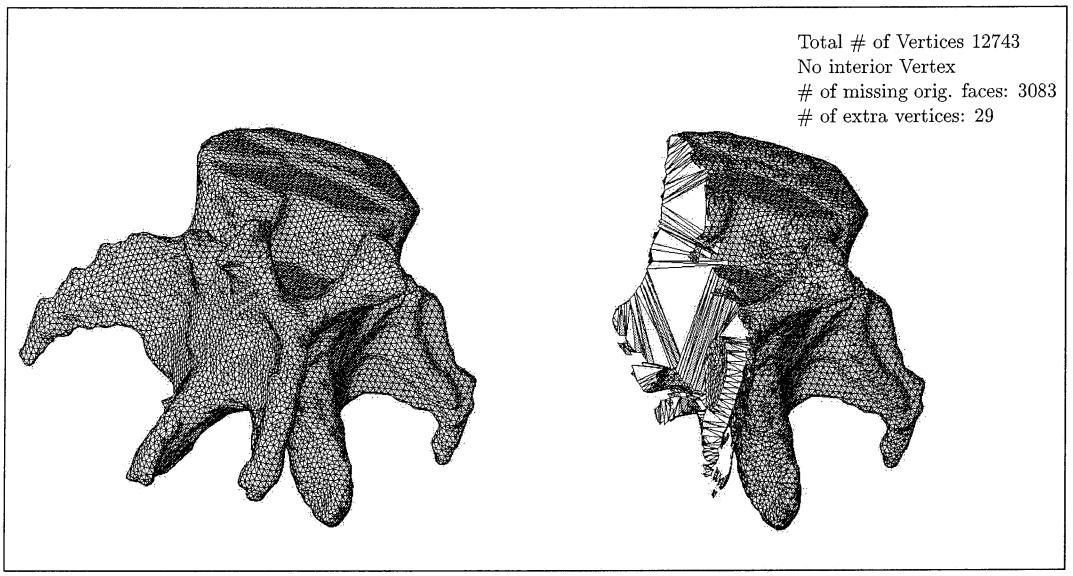

Figure 21: The face recovery algorithm coupled with Delaunay meshing over a vertebra model and its cross sectional view.

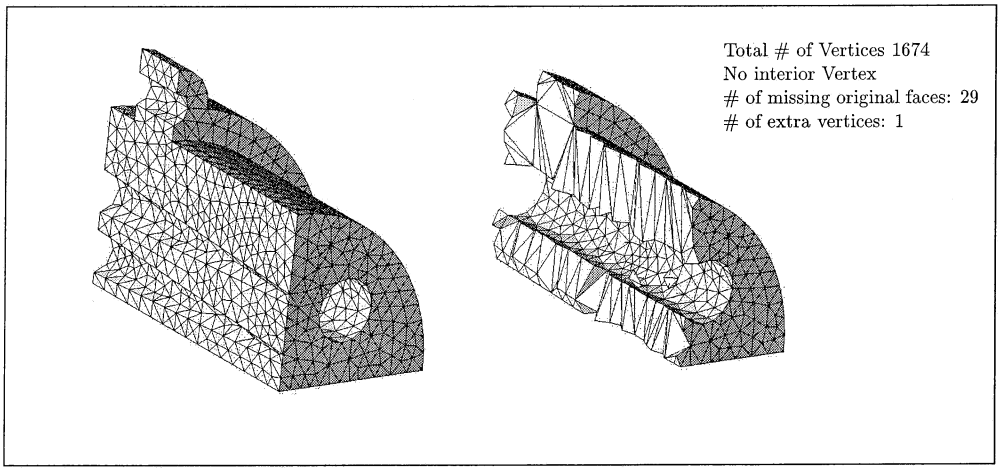

Figure 22: The face recovery algorithm coupled with Delaunay meshing over a CAD model and its cross sectional view.

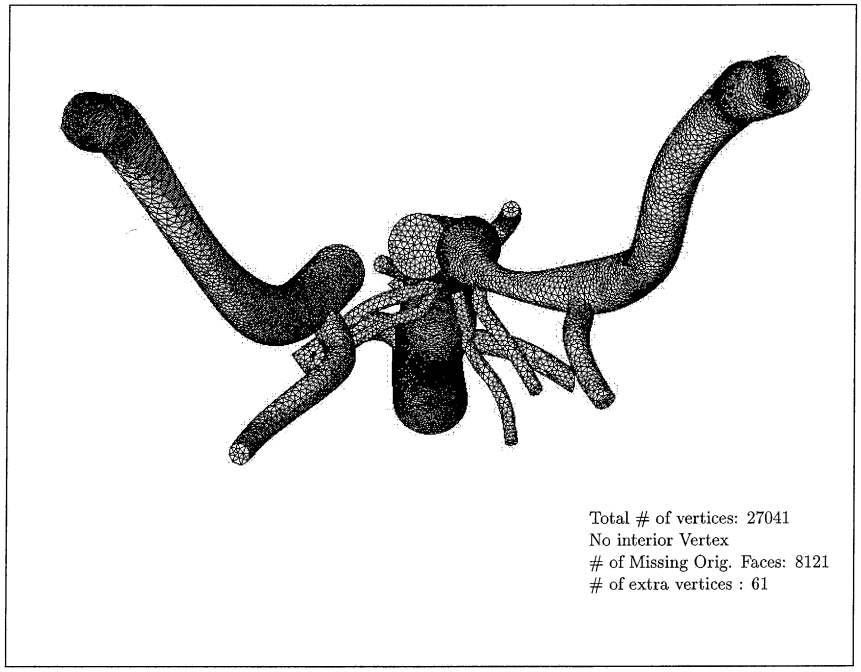

Figure 23: The face recovery algorithm coupled with Delaunay meshing over a non-manifold artery model [12]