INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
*Int. J. Numer. Meth. Engng* 2004; **00**:01 *Prepared using* nmeauth.cls *[Version: 2000/01/19 v2.0]*

# Constrained Boundary Recovery for Three Dimensional Delaunay Triangulations

Qiang Du[1,3] and Desheng Wang [2,3,*]

[1] *Department of Mathematics, Penn State University, University Park, PA 16802, USA*
[2] *Civil and Computational Engineering Centre, School of Engineering, University of Wales Swansea, Singleton Park, Swansea SA2 8PP UK.*
[3] *Lab for Scientific and Engineering Computing, Academy of Sciences, Beijing, China*

SUMMARY

A new constrained boundary recovery method for three dimensional Delaunay triangulations is presented. It successfully resolves the difficulties related to the minimal addition of Steiner points and their good placement. Application to full mesh generation are discussed and numerical examples are provided to illustrate the effectiveness of guaranteed recovery procedure. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: guaranteed constrained boundary recovery, three dimensional Delaunay triangulation, Steiner point insertion and placement, mesh generation and optimization

## 1. Introduction

Efficient and robust mesh generation is an essential part of many scientific and engineering computational challenges. For unstructured mesh generation, Delaunay-based algorithms have become very popular[1-17]. For such algorithms, the input data are often given in a discrete formulation of the domain boundary which, in the three dimensional space, may represent a surface triangulation of the boundary. Delaunay-based methods usually first produce an initial triangulation that forms the convex hull of the boundary points which may not match with the prescribed boundary surface, that is, the triangulation may not satisfy the constraints (edges and faces in 3D) imposed by the surface triangulation. This leads to the problem of recovering the boundary geometric constraints from the initially constructed triangulation, or simply, the problem of boundary recovery. Such problems have been successfully resolved in two dimensional spaces[3,4], while they are still under active investigation in three dimension.

*Correspondence to: Civil and Computational Engineering Centre, School of Engineering, University of Wales Swansea, Singleton Park, Swansea SA2 8PP UK.

Contract/grant sponsor: China State Major Basic Research Project G199903280 and US NSF-DMS 0196522, NSF-ITR 0205232

Roughly speaking, there exist two types of three dimensional boundary recovery procedures. The first is the *conforming boundary recovery* which is done by applying edge/face splitting to recover a constraint as the concatenation of edges/faces. It usually requires the insertion of points to the missing constraints [13-16]. The effectiveness of such recovery methods has been demonstrated in many cases. Recently, we have also developed a very robust and theoretically guaranteed scheme for conforming boundary recovery [20]. The second approach is the *constrained boundary recovery*, which does not allow extra points being added to the missing constraints during the recovery. The constrained boundary recovery respects the local sizing specification of the input boundary triangulation and offers more robustness than the conforming recovery when meshes from two connected regions are merged together. We note that both the conforming or constrained boundary recovery may be applicable with input being a surface triangular mesh, however, the resulting tetrahedral mesh after boundary recovery may not be strictly Delaunay. Due to the Schoenhardt configuration[17], it is well known that the success of constrained boundary recovery may rely on the occasional insertion of interior Steiner points [17].

To list a few well-studied methods for constrained boundary recovery, we begin with the work by George et al.. In [12], they introduced ingenious techniques to reconstruct a prescribed surface triangulation. The surface edges and faces were recovered by a series of edge/face swaps (or flips), and occasionally some heuristic insertions of interior points (Steiner points). The heuristic procedures do not work in all cases, as several examples in [11] provides evidences that the above methods may fail in some situations. In [13], Weatherill and Hassan introduced methods in which they first insert intersections points on missing edges and faces to get a conforming recovery, then they try to delete the inserted points through retriangulating the tetrahedra set connected to those points. The existence of Schoenhardt configuration again implies that the success of such deletion can not always be guaranteed. Until now, there is no published algorithm on constrained boundary recovery that claims theoretically guarantee for its success in all cases. A number of difficult issues remain to be resolved, such as the placement of the Steiner points, the minimum bound on such points, etc [17].

In this paper, a new constrained boundary recovery algorithm is presented which combines conforming boundary recovery with a procedure called added-points-splitting to give a constrained recovery of the missing boundary constraints. The convergence of the method in all cases is theoretically proved. Our approach is different from the previous works [12-15,17] where one first recovers all the missing edges and then faces. Briefly, in our approach, for each missing face, we first recover its missing edges (if there exist) together itself by the modified edge/face splitting method proposed in [20]. This results in the conforming recovery of the missing face, as some points are added to its missing edges or the interior of the face. Then the points added to each missing edge of the missing face are split, one by one, into two interior Steiner points located away from the missing face. The splitting continues until a complete constrained recovery of the missing edge. Once all the missing edges of the face are recovered, we again apply the splitting operations sequentially to the added points on the face until the recovery of the missing face. The splitting operation includes a directional perturbation and a constrained Delaunay insertion of a face-symmetric point. The cavity is appropriately chosen so that a basic flippable local tetrahedra set can be generated. For a missing edge, by flipping the local tetrahedra set, the edge or a part of it is recovered in a constrained manner; for a missing face with already recovered edges, the number of added points are gradually reduced until the complete recovery of the face.

Note that in the above constrained recovery procedure, when a missing face is recovered, each point added in the initial conforming recovery procedure (if there exists any) is split into two interior Steiner points positioned on the two sides of the recovered face. For applications to full mesh generations, either vertex suppression is performed (if allowed), or the added points are relocated to suitable positions with respect to the given sizing field. For practical 3D Delaunay constrained boundary recovery, we first perform several basic swapping operations described in [20] to recover a large portion of missing constraints. This will involve some simple local operations. Then, we apply the proposed recovery method to the remaining missing faces. This combination will decrease the added Steiner points greatly and also leads obviously to convergence.

Overall, our proposed constrained boundary recovery is proved to work for all cases theoretically in this paper. Moreover, our approach also addresses some key aspects of boundary recovery: how to recover the missing constraints, where the added Steiner points should be, and how to minimize the number of added Steiner points. We also demonstrate in this paper the effectiveness and robustness of the proposed method through various numerical meshing examples.

The remaining part of the chapter is organized as follows: some basic concepts such as Voronoi tessellation, Delaunay triangulation, conforming and constrained boundary recovery are briefly described in Section 2. The existing conforming and constrained boundary recovery methods for 2D and 3D Delaunay triangulation are discussed in section 3. The details of the proposed new recovery method are presented in Section 4 and Section 5 for 2D and 3D respectively. The application of the method to full 3D Delaunay meshing is discussed in Section 6, along with various numerical meshing examples and meshing statistics. Finally, conclusion remarks are given in Section 7.

## 2. Preliminaries: some basic concepts

Given a polyhedra domain $\Omega$ in $R^d$ (d=2,or 3) and a set of points $S = \{Z_i\}_{i=1}^k$ in $\bar{\Omega}$, the *Voronoi region* $\widehat{V_i}$ corresponding to the point $Z_i$ consists of all the points in $\Omega$ which are closest to $Z_i$ than to other points $\{Z_j\}_{j\neq i}$. With the usual Euclidean metric, the union $\{\widehat{V_i}\}_{i=1}^k$ is called a *Voronoi tessellation* or a *Voronoi diagram* of $\Omega$. The geometric dual of a Voronoi tessellation is known as the *Delaunay triangulation*, which shares the empty ball property, that is, the circumsphere of each element does not contain any other vertex. This results in the *maximizing the minimal angle* property in two dimension and the *connection of nearest points* property in general dimensions. In practical implementation, the *Incremental Insertion method* [17] may be followed to construct the Delaunay triangulation. The *Delaunay insertion kernel*, including the construction of the so-called *Base, Cavity* and *Ball*, is used for the addition (or insertion) of Steiner point [17]. For a given point set $S$, the Delaunay triangulation forms a triangular mesh (tetrahedral mesh in 3D) of its convex hull in general.

There are various ways to generate a triangular mesh for a given domain $\Omega$ that is in conformity with a given sizing function, i.e., the edge lengths obeying given specifications. Among them, the popular Delaunay-based methods starts with a discretization of the domain boundary in accordance with the given sizing function. For planar domain, this corresponds to the discretization of a piecewise linear polygon, while for three dimensional cases, the discretization is formed by a closed surface triangulation. Denote the points (vertices) of the discretized boundary by $S$, the next step is to form the Delaunay triangulation of $S$ using the Incremental insertion method. The resulting triangulation forms an initial mesh of the convex hull of $S$. Then, field points are generated according to the sizing function and inserted into the initial mesh or the most recently updated mesh. Finally, optimization is performed to improve the mesh quality.

It is important to elaborate on the fact that the boundary of the initial mesh often does not match the discretization of the domain boundary, see Fig1. In other words, some *constraints* (edges, faces or surface triangles) of the surface triangulation are missing from the initial mesh. This leads to the *boundary integrity* problem [3,4,12,13]. The retrieving of the missing boundary constraints can be done through either a priori re-definition of the constraints, if a satisfactory topological and geometrical equivalent can be found in the initial mesh, or by a posteriori recovery of the topological and geometrical requirements of the missing constraint. The latter is called the *boundary recovery* [3,4,12,13,17].

The a posteriori recovery of a missing constraint from the initial mesh consists of some local operations to recover the constraint. Based on the characteristics of the local operations, the boundary recovery approaches can be classified into two categories: *conforming* and *constrained* boundary recovery [17]. In conforming boundary recovery, usually additional points are added to the missing constraint, as shown in Fig1(3); while in constrained boundary recovery, no points are allowed to be added to the missing constraint, and only occasional interior Steiner points are inserted, as shown in Fig1(4) (the most right).

If the input is a piecewise linear complex(PLC), Schewchuk proposed the *constrained Delaunay triangulation* (CDT) for boundary recovery [18]. For a PLC, Schewchuk's algorithm first adds points to missing edges of the PLC in order to reconstruct it as a union of edges; then a CDT is constructed without introducing any interior points. Such a recovery scheme preserves some nice theoretical properties of Delaunay triangulation and works well with provably good Delaunay refinement methods for tetrahedral mesh generation. However, as extra points on missing edges are added, such an approach belongs to the conforming boundary recovery category.

For practical applications, whenever possible, the constrained boundary recovery is often a better alternative to the conforming counterpart. In the next section, we will discuss the existing methods for 2D/3D conforming and constrained boundary recovery.

## 3. Existing boundary recovery methods for conforming and constrained 2D and 3D triangulation

In this section, we briefly recall the features of some typical existing conforming and constrained boundary recovery approaches. Our focus is on the extendibility of two dimensional methods to the three dimensional case.

### *3.1. Conforming boundary recovery*

Many existing methods for 2D conforming boundary recovery have one common characteristic: the addition of points on missing edges through the typical Delaunay insertion procedure or others, see for example [18]. These points can be intersections points of the missing edge with the available mesh or the midpoints of the missing edge [17]. Due to the nearest connection

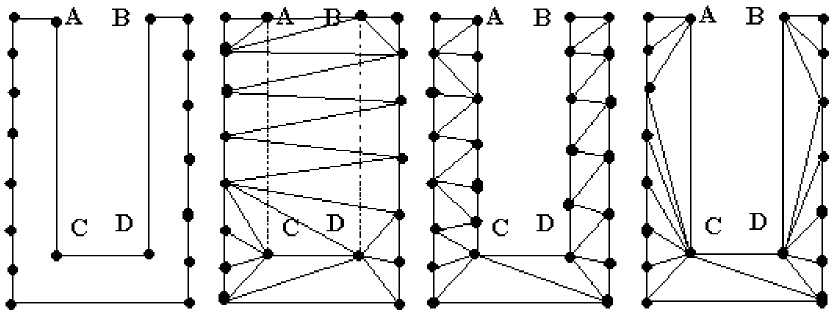

Figure 1. Boundary recovery for 2D case, from left to right: (1) boundary discretization; (2) the missing specified edges: $AC$, $BD$; (3) conforming boundary recovery; (4) constrained boundary recovery.

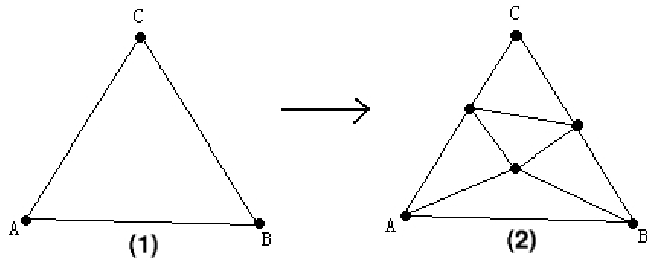

Figure 2. Conforming recovery for a missing face: (1) a missing face and its two missing edges $AC$, $BC$; (2) the face is recovered as a concatenation of sub-faces.

property of the Delaunay triangulation, a missing edge can always be recovered as a union of edges in the mesh. The difference of the existing 2D methods often lies on the number and the positions of the added points on the missing edges. Also, when recovering a missing edge, most of the algorithms do not pay any attention to whether an existing or a already recovered constrained edge could be destroyed.

Comparing with 2D cases, three dimensional conforming boundary recovery is different in several aspects. At first, a missing face may contain one or more missing edges, i.e., when recovering the face, its missing edges also need to be recovered. Secondly, one often first recovers all the missing edges then proceeds to the missing faces, or reconstructs each face (triangle) together with its missing edges independently. A similarity with the 2D procedure is the addition of intersection points or some other kinds of points (analogous to midpoints for edges) on missing constraints (edges or faces), see a detailed discussions in [17] on the three dimensional conforming boundary recovery. When recovering a missing edge or face, as in the two dimensional case, in order to protect the Delaunay property, most existing methods do not protect the existing recovered constraints, that is, no consideration is given to whether some already-recovered constraints would again be destroyed. This could result in recursive the addition of points. Special treatments must be required for the convergence. For a missing face, i.e., a surface triangle, the final reconstruction is a concatenation of sub-triangles in the mesh after the conforming recovery, see the illustration in Fig2.

In [20], the authors proposed a theoretically guaranteed three dimensional conforming boundary recovery which recovers the missing faces one by one. For each missing face, its missing edges are recovered first followed by the recovery of the face. The recovery process uses the addition of nearest-mid intersection points for missing edges and the addition of intersection points for missing faces. The Delaunay insertion kernel is modified so that it protects any existing constraint or any recovered parts of a missing constraint. This guarantees the convergence of a complete conforming boundary recovery. Moreover, when adding points to a missing constraint, some neighboring missing constraint may be recovered simultaneously.

In order to achieve constrained boundary recovery, further modification of the above method is needed. In particular, the recovery of a missing face and its edges should be independently done. In essence, such a strategy makes the method work in both two and three space dimensions.

## *3.2. Constrained boundary recovery*

Among methods for two dimensional constrained boundary recovery [3,4,17], perhaps the most popular and the easiest to implement method is the random or serial diagonal edge swapping algorithm proposed by P.L.George in [13,17]. For a missing edge, the recovery procedure begins with finding the intersection edges of the missing edge with the existing mesh (actually with the edges of the mesh), followed by randomly or sequentially swapping the intersection edges, if the operation is allowed. If the edge is recovered, the process is terminated; else, the above two steps are iterated until the final recovery, see the illustration in Fig3. The diagonal edges swapping is a local operation, hence making the recovery efficient. The recovery of each missing edge is independent from any other missing or constrained edge, and accordingly does not affect the recovery of other edges. Furthermore, due to the fact that for an arbitrary 2D piecewise linear polygon, it is always guaranteed that one may triangulate it with its boundary edges being constrained and without introducing any inner field points (Steiner points), we see from the above two properties that the two dimensional constrained boundary recovery problem is completely resolved by applying the above diagonal edges swapping [17].

For three dimensional constrained boundary recovery, however, the solution is not ideal. Due to the Schoenhardt configuration shown in Fig4, a constrained triangulation for an arbitrary polyhedron is not always guaranteed without introducing some inner field points, i.e., Steiner points [17]. In [12], P.L.George introduced a method combined with some heuristics that uses a series of complex edges and faces flipping for the recovery, and occasionally some Steiner points are added to generate the basic flippable configurations. Though the method has been demonstrated to be very successful in numerical testing, no theoretical guarantee can be claimed on the success of the method in all cases [11] as it does not adequately address issues like where to place the Steiner points and how many Steiner points are needed. Later, in [13], Weatherill and Hassan introduced another method which first generates a conforming boundary triangulation with the insertion of intersection points on the missing faces or edges, then the Ball (the elements emanating from the point) of each added point is re-triangulated to delete the added point. Since the deletion can not always be done successfully, this method also does not offer guaranteed success. Up to now, there has been very few further development on three dimensional constrained boundary recovery [17].

Since the two dimensional conforming boundary recovery approaches often can be easily extended to three dimensional cases, a natural question arises in comparison: can the 2D constrained boundary recovery algorithm be extended to 3D cases? To answer such a question

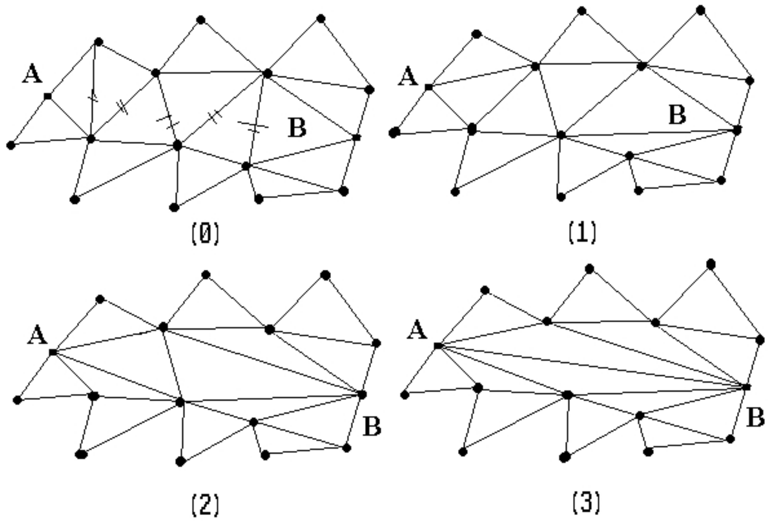

Figure 3. Diagonal edges swapping for a missing edge.

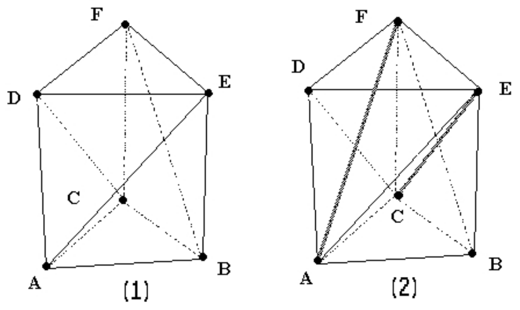

Figure 4. The Schoenhardt configuration

affirmatively, it is crucial to develop the right methodology for the two dimensional constrained boundary recovery first. In the 2D case, a series of local swapping may be applied to recover the missing edge and there is no need to introduce any Steiner point. In the 3D case, however, the issue of Steiner points starts to surface. This difference tells us that even for the 2D case, we should develop a new method which considers the generation and deletion of the Steiner points. In some sense, the diagonal edges swapping method is hiding the problem of Steiner points as their deletion can be done successfully in 2D. Thus, we see naturally where the improvement needs to be made in a new approach: for the recovery of a missing edge, the new method should include the addition of intersection points, the point splitting, the natural generation of Steiner points, the deletion of the Steiner points and the final recovery of missing edges. The two dimensional version of such an approach is described in the next section while in Section 5, the method is naturally extended to the three dimensional cases, and it solves the three dimensional constrained boundary recovery problem by addressing several important

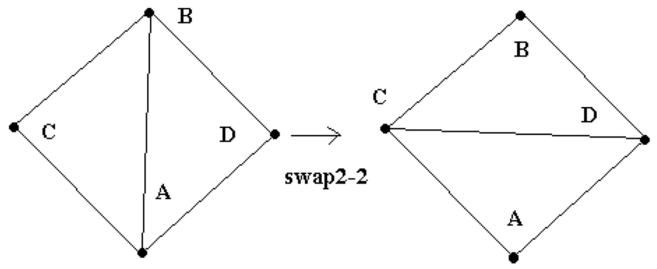

Figure 5. Diagonal edge swapping: *Swap2-2*

issues with satisfaction.

## 4. A new 2D constrained Delaunay boundary recovery

Before introducing the new algorithm for two dimensional constrained boundary recovery, we first present some basic definitions.

**DEFINITION 4.1**. Let $P$ be a vertex. All the elements (i.e., triangles) emanating from $P$ is called the *Ball* of $P$, denoted by *Ball(P)*. For an interior vertex, its Ball is a closed and connected polygon in 2D and a polyhedron in 3D.

**DEFINITION 4.2**. Let $\triangle ABC$ and$\triangle ABD$ be two triangles sharing the common edge $AB$, shown in Fig5. If the quadrilateral $ACBD$ is convex, we can swap the diagonal edge $AB$ to $CD$, and form two new triangles $\triangle ACD$ and $\triangle CBD$. This swapping is called *Swap2-2*.

The importance of *Swap2-2* is the direct recovery of edge $CD$ in the boundary recovery, which has been discussed before. It also plays a role in preserving the Delaunay property or the empty disc property and in mesh optimization [17].

**DEFINITION 4.3**. Let $P$ be an interior vertex and its Ball be denoted by Ball(P). An operation that gets a triangulation of Ball(P)'s vertices except $P$ itself, with respect to the boundary edges of Ball(P), is called a *Vertex Suppression* of $P$.

In Fig6, it is shown in 2D cases there are two methods for the suppression of $P$. We can first delete $P$ and then re-triangulate the remaining polygon; also, we can perform a series of *Swap2-2* operations to the inner edges of Ball(P) until the existence of a configuration where three edges are connected to $P$, then, we can remove $P$ and get the triangulation we want.

**DEFINITION 4.4**. Let $P$ and Ball(P) be defined as above. Then, there must be a polyline which consists of some inner edges connecting $P$ and divides Ball(P) into two connected and non-overlapping parts. Such a division is called a *Polyline Cutting*.

In Fig7, the polyline $\widehat{APB}$ and $\widehat{CPB}$ both form Polyline Cuttings of Ball(P).

**DEFINITION 4.5**. Let $P$ and Ball(P) be defined as above and shown in Fig8(4), where $P$ is on the line $AB$. We call the following operations *Point Splitting*, i.e., the splitting of $P$, by mirror symmetry with respect to $AB$.

(a). First, $P$ is perturbed to $P^1$ along the direction normal to $\overrightarrow{AB}$. Denote the symmetric point (mirror image) of $P^1$ with respect to the line $AB$ by $P^2$. Let the perturbation distance of $P$ be $\epsilon$ (here, $\epsilon$ is required to be small enough such that both $P^2$ and $P^1$ stay inside Ball(P), such

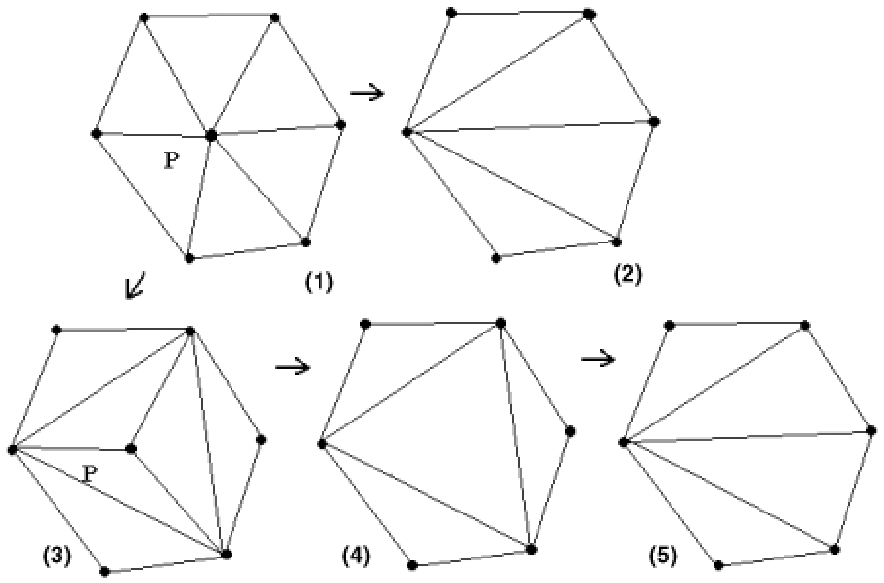

Figure 6. Vertex Suppression: (1) ball(P); (2) method 1: retriangulation; and method2: from (3) edges swapping to (4) final vertex deletion until (5) final swapping

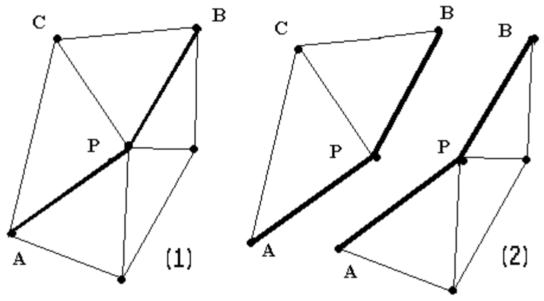

Figure 7. Polyline Cutting: (1) choosing the polyline; (2) cutting into two parts

an $\epsilon$ always exists and it it related to the point $P$ and its ball configuration). An example of the perturbation is shown in Fig8:(2)-(3).
(b). The second step (shown in Fig8) is to add the symmetric point $P^2$ in the Delaunay mesh. In the addition, we let $\widehat{AP^1B}$ be a Polyline Cutting of Ball(P) and choose the part which contains $P^2$ as the Cavity, shown in Fig8.
(c). Insert $P^2$ into the mesh using the classical Delaunay insertion kernel[17], with the Cavity chosen as in the above. This operation is shown as in Fig8(5).
(d). Perform *Swap2-2* to the edge $P^1P^2$, then $AB$ becomes an edge of the final mesh. The last step in Fig8(6) gives an illustration of this operation.

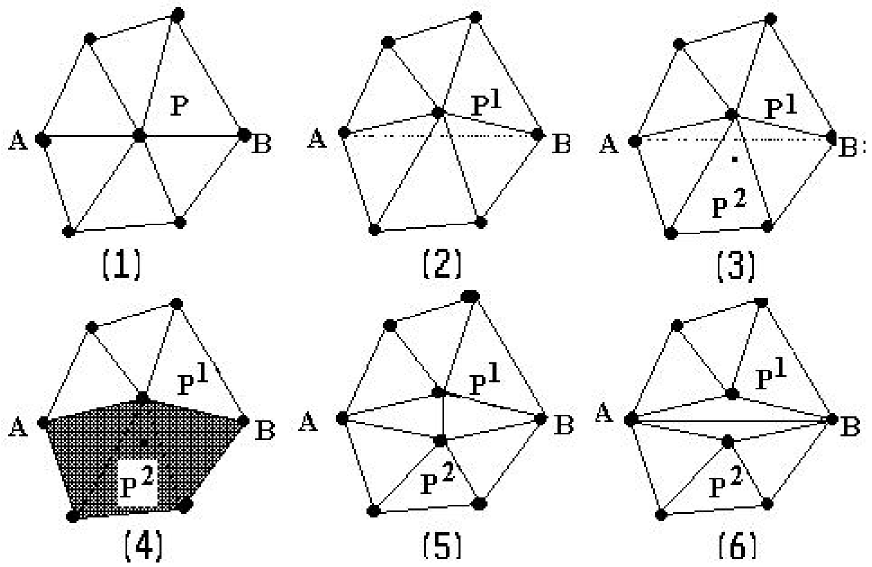

Figure 8. Point Splitting: (1) ball(P); (2) perturbing P to $P^1$; (3) adding $P^2$; (4) choosing cavity; (5) mesh updating; (6) final swapping

By the above splitting, $P$ is replaced by two points $P^1$, $P^2$, and $AB$ becomes an edge of the final triangulation. If $AB$ is an constrained edge or a part of some constrained edge, we have recovered a missing edge or a part of it through the splitting. Furthermore, two inner field points, i.e., two Steiner points are generated naturally and they can be subject to vertex suppression. Obviously, for two dimensional cases, we can delete them through the vertex suppressions. For 3D, the vertex suppression cannot not always be guaranteed to work, even though it can be performed in 2D.

With the above preparation, we now come to the description of our new algorithm for constrained 2D boundary recovery. For each missing edge, first, we perform the proposed nearest-mid intersection points insertion to recover the edge in a conforming manner; then, for each added point on the edge, we sequentially perform point splitting and vertex suppressions to recover a part of or the whole missing edge in a constrained manner. This recovery process is shown in Fig9. The pseudo-code form of the algorithm is given as follows:

**Algorithm: Point-Splitting Boundary Recovery.**

```
For each missing edge (say $AB$), perform the operations:
(a). Obtain a conforming recovery of $AB$ by adding the nearest-mid intersection points
     on $AB$ (see Section2). $AB$ is then reconstructed as a union of edges in the mesh.
(b). For each added point (say P) on $AB$, perform the operations:
         (c). Perform Point Splitting to $P$ to obtain two Steiner points $P^1, P^2$ in
              the mesh. A part of $AB$ is then recovered in a constrained form.
         (d). Perform Vertex Suppressions to $P^1$ and $P^2$.
Enddo
```

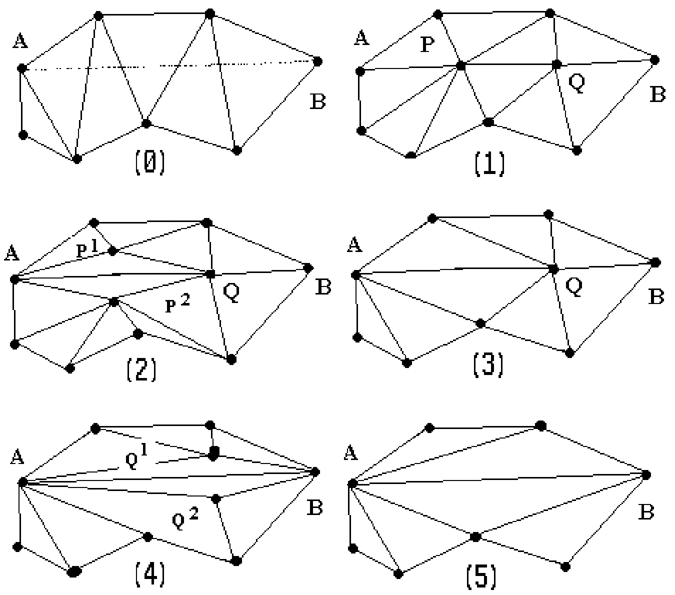

Figure 9. Point-Splitting Constrained recovery for a missing face: (0) edge $AB$ is missing; (1) conforming recovery for $AB$; (2) splitting of the added point $P$; (3) suppression of Steiner points; (4) splitting of added point $Q$; (5) final recovery

Enddo

In the above boundary recovery procedure, the convergence of the initial conforming boundary recovery for each missing edge has been theoretically proven in [20]. The splitting of each added point recovers a part of the missing edge (or the whole edge) in the constrained form. With such splittings, the recovered part of the missing edge becomes longer and longer and after a finite step, the complete constrained recovery is achieved. Furthermore, the vertex suppressions can always be done successfully. Thus, we have the following theorem for the convergence of the proposed algorithm:

**Theorem 4.1.** The above *Point-Splitting Boundary Recovery algorithm* is convergent, and there is no Steiner points to be included in the final mesh.

Generally speaking, the above new algorithm is more complex than the diagonal edges swapping introduced by P.L.George[17] but it captures the essence of constrained boundary recovery in having a suitable treatment of Steiner points. Through the insertion of nearest-mid intersection points, a conforming recovery of the missing edge is obtained, which provides the initial positions for the Steiner points introduced later; through Point-Splitting, it naturally generates the Steiner points and recovers the constrained edge or a part of it; through Vertex-Suppression, it deletes the generated Steiner points naturally, while preserving the recovered edges. More importantly, it leads naturally to a three dimensional constrained boundary recovery method that includes, similar to 2D cases, conforming 3D boundary recovery, vertex splitting for the natural generation of Steiner points and the recovery of constraints, vertex

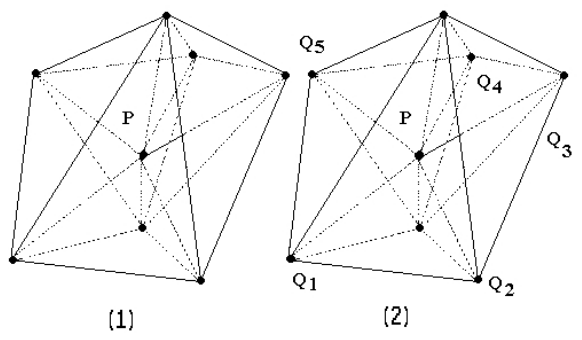

Figure 10. Poly-triangles Cutting.

suppression or optimal placement for the treatment of the Steiner points. Details on these extensions are given in the next section.

## 5. 3D constrained Delaunay boundary recovery

In three dimension, the recovery procedure contains edges recovery and faces recovery which will be treated separately, even though there are many common features between them.

### *5.1. Basic definitions and Properties*

We again begin with some definitions.

**DEFINITION 5.1**. In a three dimensional tetrahedral mesh, all the tetrahedra emanating from a vertex is called its *Ball* and all the elements connecting an edge is called its *shell.*

**DEFINITION 5.2**. Let $P$ be an inner vertex of a tetrahedral mesh with ball Ball(P). An operation is called a *Vertex Suppression* of $P$ if a constrained tetrahedralization is obtained from the surface points of Ball(P) with the input being the surface triangles of Ball(P) but with $P$ not being a vertex of the tetrahedralization.

The suppression of an inner vertex is always guaranteed in the 2d case, but not in the 3D case due to the existence of the Schoenhardt configuration.

**DEFINITION 5.3**. Let $P$ and Ball(P) be defined as above. A subset of the inner triangles of Ball(P) is called a *Poly-Triangle Cutting* of Ball(P) if it is connected and divides Ball(P) into two connected and non-overlapping subsets.

As an illustration, the set of inner triangles $\{PQ_1Q_2, PQ_2Q_3, PQ_3Q_4, PQ_4Q_5, PQ_5Q_1\}$ forms a Poly-Triangle Cutting of Ball(P) in Fig10.

**DEFINITION 5.4** Let $\triangle ABC$ be a triangle with $O$ as the centroid, and $P$ be a point shown in Fig11. The vertices of $\triangle ABC$ are called *directionally oriented* (or ordered), if the outer normal direction $\overrightarrow{ON}$ is defined as $\overrightarrow{AB} \times \overrightarrow{BC}$. The triangle $\triangle ABC$ is said to be *visible* to $P$, if $\overrightarrow{ON} \cdot \overrightarrow{OP} > 0$.

The visibility of $\triangle ABC$ to $P$ is equivalent to the signed volume of the tetrahedron $ABCP$ being positive, or, $P$ being on the side of $\triangle ABC$ that $\overrightarrow{ON}$ points to. In Fig11, $\triangle ABC$ is

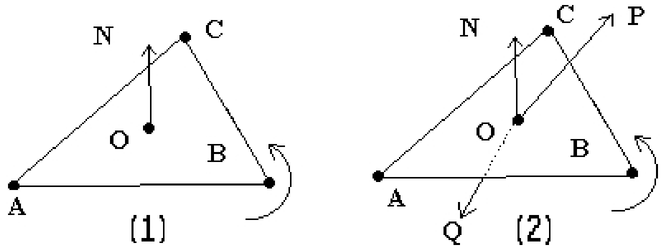

Figure 11. Visibility of a face to a point: (1) directionally oriented; (2) visible to $P$, not to $Q$

visible to $P$, but not to $Q$. If $\overrightarrow{ON} \cdot \overrightarrow{OP} = 0$, $P$ is on the plane defined by the three points $A, B, C$.

The incremental Delaunay insertion kernel for adding a new point (say $P$) consists of three parts: finding the Base, constructing the Cavity, forming the Ball and updating the mesh[17]. In practice, due to the round-off errors, the initially formed Cavity is usually processed by a correction procedure to ascertain its *validity* [17] which is defined as: first, it should be connected and have no inner vertex; then, each boundary face (i.e. triangle) of the cavity should be visible to the added point $P$. Let $T_c$ be the initially defined cavity for $P$, we now recall a simplified version of the correction procedure proposed in [20] in the following pseudo-code form.

**Algorithm: Cavity Correction.**

```
1 Find $\{F_i\}_{i=1}^m$, the external faces $F$ of $T_c$; define the $m$ simplices $K_j$: $K_j = \{F_j, P_{i+1}\}$
  with $Det(K_j)$ being the signed volume of $K_j$.
2 Set $T = T_c$, loop over $j = 1$ to $m$
      If $Det(K_j) > 0$, continue;
      If $Det(K_j) < 0$, $T = T - K_j$;
      If $Det(K_j) = 0$, $T = T \cup K$,
  where $K$ is the simplex of $T_{old}$, with face $F_j$, not in $T_c$.
3 If $T$ is not affected by step 2, end; else update the set of $F$ and go back to step 2.
```

**DEFINITION 5.6.** With reference to Fig12, the three basic single-step swappings are

1 *Swap23*: exchange *abcd* and *abce* with *deab*, *debc*, *deca*, where *de* intersects the interior of *abc*;
2 *Swap32*: the inverse of *Swap23*;
3 *Swap44*: exchange *abde*, *bcde*, *abfe*, *bcdf* with *acde*, *abce*, *acdf*, *abcf*, where *ac* intersects the interior of *bde* or *bdf*.

*Swap23* results in the generation of an edge *de* and will be used in the recovery of missing edges; *Swap32* results in the generation of an face *abc* and will be used for recovering missing faces; and *Swap44* can be used in both edges and faces recovery. The first two swaps will be the most basic swapping operations in our 3D constrained boundary recovery.

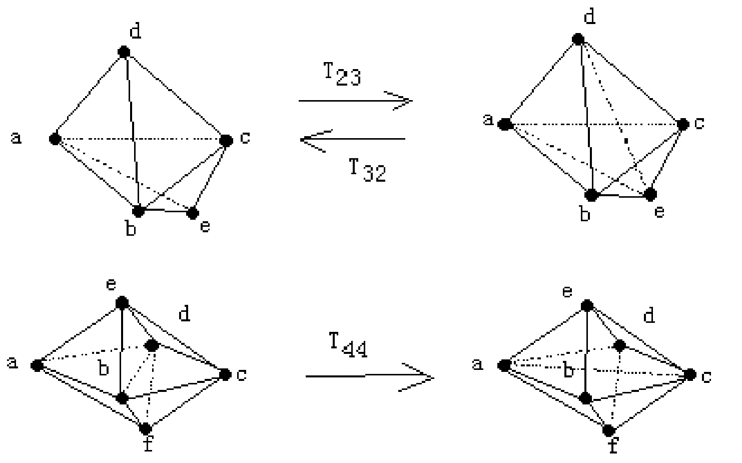

Figure 12. Basic swappings

These three basic operations are not enough to recover all missing constraints in practice. In [17], P.L.George proposed a series of complex swapping operations. We describe two more basic tetrahedra configurations and transformations (or say flippings) here which forms the key parts of our 3D constrained boundary recovery.

**DEFINITION 5.7 Cap-Configuration and Cap-Transformation**: With reference to Fig13, $P, Q$ are two plane-symmetric points with respect to the plane defined by the co-planar points $P_1(=A), P_2, ..., P_{n-1}, P_n(=B)$. And $AB$ intersects with the interior of all the faces $\{PQP_j\}_{j=2}^{n-1}$. The set of tetrahedra $\{PP_iP_{i+1}Q\}_{i=1}^{n-1}$, whose union is called a *Cap-Configuration.*

In the above tetrahedra union, $AB$ does not exist as an edge. In Fig13, it is shown on the right that if we triangulate the 3D-polygon $\{P_j\}_{j=1}^{n}$ into a set of triangles $\{\triangle_i\}_{i=1}^{n-2}$, we can exchange the above tetrahedra set with the union of three parts: $ABPQ$, $\{P\triangle_i\}_{i=1}^{n-2}$, $\{\triangle_iQ\}_{i=1}^{n-2}$. Here, we generate a new edge $AB$ in the exchanged tetrahedra union. Such a exchange is called *Cap-Transformation.*

Obviously, if $AB$ is a missing constrained edge or a part of a constrained edge, through the Cap-Transformation, it can be recovered in a constrained manner. In the simplest case, $n = 3$, the Cap-Transformation is reduced to *Swap23*, thus it gives a natural extension of *Swap23*.

The above co-planarity condition of $\{P_j\}_{j=1}^{n}\}$ can be relaxed. Actually, we only need that $AB$ intersects the interior of the faces $PQP_2, ..., PQP_{n-1}$.

In the configuration of *Swap32*, edge *de* intersects only one triangle *abc*. If there are more than one triangle, we have the following extension of *Swap32*.

**DEFINITION 5.8 Shell-Configuration and Shell-Transformation**: With reference to Fig14, $A$ and $B$ are two points on the two sides of the plane defined by the co-planar points $\{P_j\}_{j=1}^{n}\}$. Let $AB$ intersect the interior of the 3D-polygon $P_1P_2...P_n$. With these $n+2$ points, we form a tetrahedra union: $U^* = \cup_{i=1}^{n-1}\{P_iP_{i+1}BA\} \cup \{P_nP_1BA\}$, which is called a *Shell-Configuration.* As in the Cap-configuration, if the 3D-polygon $P_1P_2...P_n$ is triangulated to be a union of triangles $\{\triangle_i\}_{i=1}^{n-2}$, we can exchange $U^*$ with the union of $\{\triangle_iA\}_{i=1}^{n-2}$ and $\{\triangle_iB\}_{i=1}^{n-2}$. Such an exchange is called a *Shell-Transformation.*

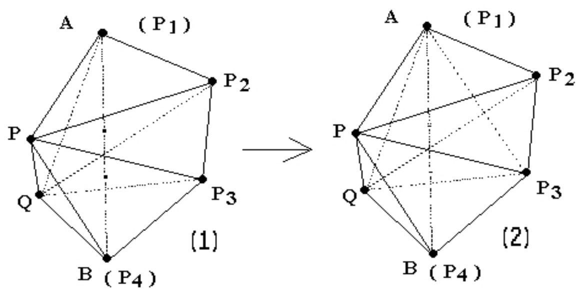

Figure 13. (1) Cap-configuration; (2) Cap-transformation.

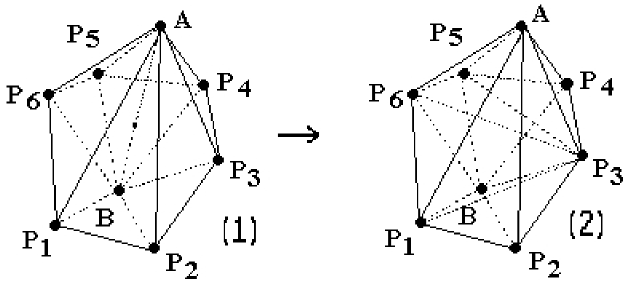

Figure 14. (1) Shell-configuration ; (2) Shell-transformation.

The simplest Shell-Conformation with $n = 3$ is just the basic operation *Swap32*.

We now present two important points splitting operations for removing edge-added-point (point added to an edge) and face-added-point (point added to a face) which will results in the constrained recovery of edges/faces.

**Algorithm: Edge-Added-Point Splitting.**

Suppose $\triangle ABC$ is a missing face with edges either missing or not missing and a conforming recovery has been performed through the addition of intersection points so that $\triangle ABC$ has been reconstructed as a union of triangles in the existing mesh. Without loss of generality, we make reference to Fig15:(1), where two points are added on $AC$, one point is added on $BC$ and one interior point is added on the face $ABC$. Relabel the points set $\{c, g, f, e_2\}$ by $\{P_1, P_2, P_3, P_4\}$ for convenience, see Fig15:(2). The "Splitting" of the added point $e_1$ means that $P_1P_4$ is generated as an edge in the updated mesh after the "Splitting", i.e., a part of the edge $AC$ is recovered in a constrained manner. Such an operation for $e_1$ is called an *Edge-Added-Point Splitting*. In more details, the operation is as follows:

(a) First, we perturb $e_1$ in the direction $\overrightarrow{N_1}$ ($\overrightarrow{N_1} = \overrightarrow{AB} \times \overrightarrow{BP_1}$) by a distance $\epsilon_1$, i.e., $e_1 = e_1 + \epsilon_1 \cdot \overrightarrow{N_1}$ (see Fig15:(3) for an illustration of its new position).
Then, we perturb $e_1$ in the direction $\overrightarrow{N_2}$ by a distance of $\epsilon_2$. Here,$\overrightarrow{N_2}$ is parallel to the plane defined by $ABP_1$ and is vertical to $\overrightarrow{AP_1}$. Fig15:(4) shows the newly perturbed

position of $e_1$. Denote the symmetric point of $e_1$ (with respect to the plane $ABP_1$) by $e_1^*$. The two distances $\epsilon_1, \epsilon_2$ are chosen to be suitably small such that $e_1, e_1^*$ are both contained in the ball of $e_1$ and all the faces $\{e_1P_jP_{j+1}\}_{j=1}^{n-1}$ are visible to $e_1^*$ (such distances $\epsilon_1, \epsilon_2$ can obviously be defined in all cases and they are related to the added point and its ball configuration).

(b) Insert $e_1^*$ into the existing mesh using the *constrained Delaunay insertion kernel*:
First, the cavity of $e_1^*$ (Denoted by $T_c$) is initialized to be the ball of $e_1$, i.e., $T_c = \text{Ball}(e_1)$. Then, $T_c = T_c - \{t_i\}$, where $\{t_i\}$ include the tetrahedra having $\{e_1P_iP_{i+1}, i = 1, 2, 3\}$ as faces and lying "above" the triangle $\triangle ABP_1$ (i.e. in the same direction as $\overrightarrow{N_1}$). With reference to Fig15:(5), $\{t_i\}$ are the tetrahedra $e_1P_3P_4f_1, e_1P_2P_3f_2, e_1P_1P_2f_2$. which do not intersect the triangle $\triangle ABP_1$. Obviously, $T_c$ may not be a valid cavity.
Then, the third step is to perform the *Cavity Correction* procedure described above to $T_c$. Denote still the updated and valid cavity by $T_c$. Since the faces $\{e_1P_iP_{i+1}\}_{i=1}^{n-1}\}$ are visible to $e_1^*$, they will be in $T_c$ as boundary faces.
Finally, we insert $e_1^*$ into the mesh with $T_c$ being the cavity. According to the constrained Delaunay insertion kernel, new tetrahedra will be formed by $e_1^*$ and the boundary faces of $T_c$. Hence, a Cap-Configuration $C_c = \{e_i^*f^i, f^i = e_iP_iP_{i+1}\}_{i=1}^{n-1}$ will be generated (see Fig15:(6)).

(c) Perform Cap-Transformation to the Cap-Configuration $C_c$. Then, $P_1P_4$ becomes an edge in the the updated mesh (the final configuration is shown in Fig15:(7)).

The above point splitting procedure deletes an added point $e_1$ on $AC$ and generates two Steiner points $e_1, e_1^*$, which are on the two side of $\triangle ABC$. And a part of the edge $AC$ is recovered in a constrained manner. Having a suitable definition for the cavity $T_c$ is the key to reach the above result. In conclusion, we have the following lemma for the Edge-Added-Point Splitting.

**Lemma 1**. The above Edge-Added-Point Splitting operation deletes one added point on a conformly recovered edge once a time, and recovers a part of the edge in constrained form (if there is only one added point, this operation results in the complete constrained recovery of the edge). Moreover, each splitting generates two Steiner points, and this splitting will never results in new added points in the interior of the missing face. Also, it will not affect the existence of any recovered or constrained face or edge.

We now deal with the splitting of points added to faces.

**Algorithm: Face-Added-Point Splitting.**

Let $\triangle ABC$ be a missing face after a conforming recovery with its three edges all recovered through the above Edge-Added-Point Splitting operations. Assume that $\triangle ABC$ still does not exist in the mesh as a face, that is, there are still points added in the interior of $\triangle ABC$. Without loss of generality, suppose three points $\{P_1, P_2, P_3\}$ are added on $\triangle ABC$ (see Fig16:(1) for reference). We call the following operation a *Face-Added-Point Splitting* which will "Split" an added point (say $P_1$) into two Steiner points. Denote the triangles in $\triangle ABC$ that connect $P_1$ by $\{f_i\}$ (see in Fig16:(2)), which forms a subpart of $\triangle ABC$. Now, we describe the Face-Added-Point Splitting operation for $P_1$.

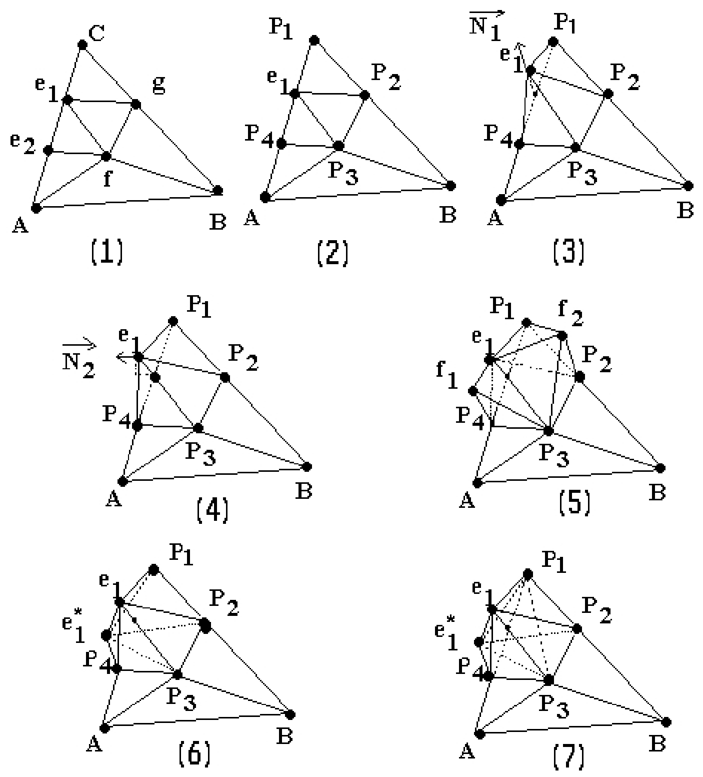

Figure 15. Operation for edge-added-point-splitting: (1) points added for conforming recovery; (2) re-labeling the added points around $e1$; (3) first perturbation of $e_1$ in the normal direction; (4) second perturbation; (5) cavity modification; (6) inserting the symmetric point $e_1^*$ and obtain a Cap-configuration; (7) Cap-transformation.

(a) Perturb $P_1$ in the direction $\overrightarrow{N}$ by a distance $\epsilon$. Here, $\overrightarrow{N} = \overrightarrow{AB} \times \overrightarrow{BC}$, i.e., the normal direction of $\triangle ABC$. Denote the symmetric point of $P_1$ (with respect to the plane $ABC$) by $P_1^*$. The distance $\epsilon$ is chosen appropriately or small enough such that $P_1, P_1^*$ are both contained in the ball of $P_1$ and the faces $\{f_i\}$ after the perturbation becomes visible to $P_1^*$. (The perturbed configuration is shown in Fig16:(3).)

(b) Insert $P_1^*$ into the existing mesh via the constrained Delaunay insertion kernel:
The cavity for $P_1^*$ is chosen as follows:
Since $\{f_i\}$ forms a Poly-triangle Cutting of the Ball of $P_1$, Ball($P_1$)is divided into two parts. We choose the part which contains $P_1^*$ as the cavity (denoted by $T_c$) for our use. Since $\{f_i\}$ are $T_c$'s boundary faces and are visible to $P_1^*$, $T_c$ is a valid cavity for $P_1^*$. Inserting $P_1^*$ into the existing mesh, a Ball-Configuration is generated (see the illustration in Fig16:(4)).

(c) Perform Ball-Transformation to the generated Ball-Configuration.
Two Steiner points $P_1, P_1^*$ are generated and the subpart shown in Fig16:(2) is reduced to that shown in Fig16:(6), which is equivalent to say that the configuration of the conforming recovery for $\triangle ABC$ is reduced. The union of new triangles is shown in

Fig16:(6), where the added points are reduced by one. In the simplest case where there is only one added point in the initial union, the result of the transformation gives a complete recovery of $\triangle ABC$ (consequence of a simplified Ball-Transformation - *Swap32*).

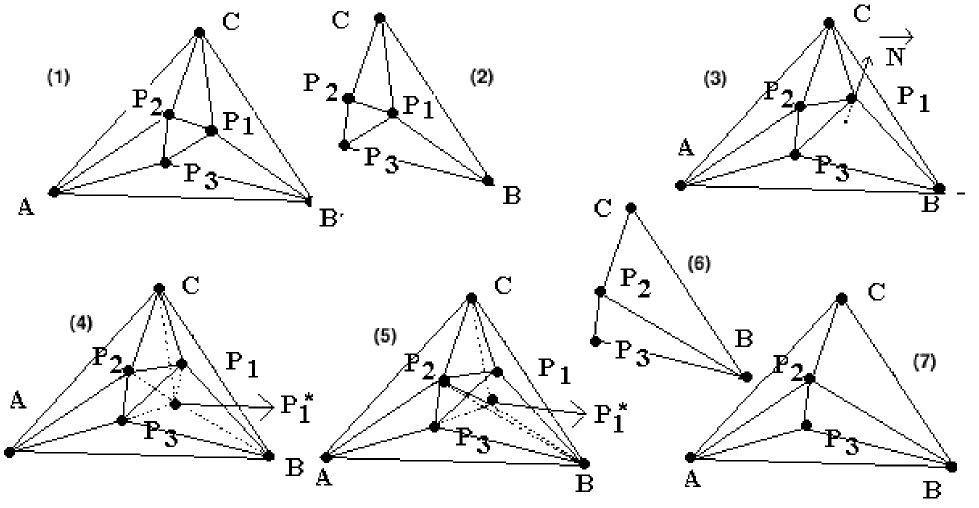

Figure 16. Operation for Face-added-point-splitting: (1) Conforming recovery; (2) A subpart:triangles connecting $P_1$; (3) Perturbing $P_1$; (4) Adding the symmetric point $P_1^*$ and generate a Ball-configuration; (5) Ball-Transformation; (6) New subpart; (7). Reduced configuration

We have the following lemma for the function and property of the Face-Added-Point Splitting .

**Lemma 2**. For a conformly recovered missing face with its three edges being recovered in constrained manner, if we perform the above Face-Added-Point Splitting, an added point on the interior of the face will be deleted; two Steiner points will be generated, and the face is still formed as a union of triangles but the configuration for the conforming recovery of the face (or more precisely, the number of triangles in the union) will be reduced in a strictly monotonic manner. In the simplest case that only one added point exists, the face is recovered completely in constrained form. Furthermore, this splitting operation will never influence the existence of any other existing/recovered constraint.

The 3D constrained boundary recovery procedure is ready to be presented which recovers the missing faces one by one, with the missing edges of each face being recovered simultaneously. Such a procedure is different from previous methods that first recover missing edges, then missing faces.

## *5.2. Conforming Face/Edges Recovery*

For each missing face, we apply the addition of intersection points procedure discussed in Section 2, see [20] for details. Here, we simply recall the basic steps.

When adding intersection points, the Delaunay insertion kernel is modified such that the insertion of each point will not delete any other existing/recovered constrained edge/face, nor a recovered part of a missing constraint. This is realized through the modification to the initial chosen cavity for the insertion point[20]. The non-interference to other constraints results in a monotonic decrease in the number of missing constraints. In [20], when we recover one missing face, its neighboring faces may also be recovered as a by-product. Here, such a by-product is given up since the Edge-added-point Splitting obviously will destroy the conforming recovery of some neighboring faces. However, there are two properties shared by the above conforming recovery: one is that when a missing edge of a missing face is recovered as a union of sub-edges, then, in the recovery of the connecting faces of this edge, the recovery of this edge is preserved, i.e., the common edges are already recovered. This will keep the conformity of the recovery of the missing faces. Another important property is that when all three edges of a face have been recovered but the face still missing, the subsequent addition of the interior intersection points is independent from the addition operation of intersection points for other missing faces. This will guarantee the convergence of the recovery.

## *5.3. Constrained Edge Recovery*

Let $f$ be a missing face and has been recovered in a conforming manner via the addition of intersection points described earlier and in [20]. If it contains edges on which intersection points have been inserted, then the first step for the constrained recovery of face is to perform the Edge-Added-Points Splittings for the added points on these edges. The pseudo-code form for the algorithm is given as follows:

**Algorithm: Constrained Edge Recovery.**

```
For the three edges (say $e_i, i = 1, 2, 3$) of $f$, do the following operations:
    if there are points $\{P_{ij}, j = 1, ..., n\}$ added on $e_i$, then
        for each $P_{ij}$, perform Edge-Added-Point Splitting.
    Endif
Enddo
```

For the above constrained edge recovery procedure, we have the following theorem:

**Theorem 5.1.** Suppose $f$ is a missing face which has been recovered in a conforming manner. If it is processed in the above constrained edge recovery procedure, then its three edges can be recovered in constrained manner through a finite number of Edge-Added-Point Splittings.

**Proof.** For an edge with intersection points being added on, by Lemma 1, we have that each Edge-Added-Point Splitting recovers a part of the edge in constrained manner. And the number of added points is monotonically decreasing. Since there are only finitely many added points, a configuration with only one added point will be reached. From Lemma1, the final Edge-Added-Point Splitting for this last added point results in the complete recovery of the edge. Furthermore, as each Edge-Added Point Splitting is performed in the Ball of the added point, it is independent of the recovery of other edges. Moreover, each Splitting will not result in new points being added. Hence, the number of required splittings is just equal to the number of initially added points which is finite.

## *5.4. Constrained Face Recovery*

Suppose $f$ is defined as in Section5.3 and its three edges have been recovered by the above constrained edge recovery procedure. If $f$ is still formed as a union of triangles, i.e., there are still added points in the interior of the face, then, we perform the Face-Added-Point Splitting for its added points for the final constrained recovery of $f$. The pseudo-code form for such recovery procedure is simply given as:

**Algorithm: Constrained Face Recovery**

```
For the interior added points $\{P_i, i = 1, ..., n\}$ of $f$, do:
    Perform Face-Added Point Splitting for $P_i$.
Enddo
```

We now give the following theorem:

**Theorem 5.2**. Let $f$ be defined as in Theorem 5.1 but with its three edges all recovered by Edge-Added-Point Splitting. If there are still interior points added on $f$, one can perform the above constrained Face Recovery operation for $f$ to recover $f$ in constrained manner through a finite number of Face-Added-Point Splittings. Moreover, its recovery will not affect other existing or recovered constrained faces.

**Proof.** The proof is straightforward. We know from Lemma2 that each Face-Added-Point Splitting reduces an added point and reduces the configuration for the conforming recovery of the face in a strictly monotonic manner. As the number of added points is finite, eventually, we get to the situation that there is only one added point on $f$. Then, the final Face-Added-Point Splitting for this point is reduced to the simple operation *Swap32* and it results in the constrained recovery of $f$. Furthermore, by Lemma 2, each Face-Added-Point splitting will not generate new added point, hence, the complete recovery needs only finite splittings, or more precisely, the number of the splittings is equal to the number of points added in the interior of the face $f$.

Combining Theorem 5.1 with Theorem 5.2, we have the following conclusion:

**Theorem 5.3**. Let $f$ be a missing face in a 3D Delaunay tetrahedralization. By performing the above described three operations: conforming recovery through adding intersection points, constrained edge recovery through Edge-added-Point Splitting, and constrained face recovery through Face-Added-Point Splitting, we can recover $f$ in constrained manner through finite steps of operations. And the constrained recovery of each missing face in the initial tetrahedral mesh will not delete any existing/ recovered face.

In conclusion, for an initial Delaunay tetrahedral mesh, if we perform the recovery procedure for its missing faces, we can get a constrained 3D Delaunay tetrahedralization in all cases. Of course, the above recovered tetrahedral mesh may not be strictly Delaunay [17].

In practice, a large portion of missing faces/edges can be recovered through the three basic swappings: *Swap23*, *Swap32*, and *Swap44*. These operations are cost-effective. Hence, like in [20], we can first perform these operations for some missing constraints; then, for the remaining missing ones, we perform the above constrained recovery procedure. The pseudo-code form for this combined constrained 3D boundary recovery scheme is as follows:

**Algorithm: Combined 3D Constrained Boundary Recovery.**

```
(a) For each missing edge or face, compare its configuration with those of Swap23, Swap32 or
    Swap44. If a match is found, perform the corresponding swapping to recover the missing
    constraint.
(b) For the remaining missing faces$\{f_i, i = 1, ..., n\}$ do:
        Perform conforming boundary recovery for $f_i$ by adding intersection points.
        Perform Constrained Edge Recovery for $f_i$'s edges if there are added points on them.
        Perform Constrained Face Recovery for $f_i$ if it is still not recovered in constrained
    form.
    Enddo.
```

Using the above constrained boundary recovery, we can derive successfully a constrained 3D Delaunay (not strictly) tetrahedral mesh from an initially given Delaunay tetrahedral mesh with a theoretical guarantee in all cases.

This above procedure addresses several important aspects of the open problems related to the 3D constrained boundary recovery, such as where to place Steiner points, when to generate Steiner points, how many Steiner points are needed. There are naturally other issues when one considers the full mesh generation process. Those issues are discussed further in the next section.

## 6. Application to 3D full Delaunay mesh generation

When applying the above proposed 3D constrained boundary recovery to full Delaunay mesh generation, the naturally generated Steiner points may result in low-quality elements near the missing constraint. To overcome this problem, we propose an approach that combines the minimization of the number of Steiner points and an advancing front repositioning of the remaining Steiner points. Let $F$ be a missing face, and $\{P_i\}$ be some Steiner points generated after the constrained recovery.

If $F$ is a boundary face, one half of these Steiner points are located outside of the domain which require no special treatment as they will not participate in the ensuing interior field points insertion for full mesh generation.

If $F$ is an inner constrained face, all of these Steiner points need to be processed. The first operation to be applied on these points is to try to delete them while keeping the recovered face $F$ protected. Such a deletion procedure goes as follows: when each Edge/Face-added-points Splitting is done, we find the ball Ball($P_i$) (or balls) of the generated Steiner point $P_i$. Delete the point $P_i$ from the ball, and try to tetrahedralize, for instance, via the Advancing front method, the remaining polyhedron formed by the boundary triangles of the ball without allowing the insertion of any additional points. This tetrahedralization can also be called a Vertex Suppression to $P_i$. Obviously, this suppression can not be done for all cases [17]. If the suppression fails, we simply keep the Steiner point unchanged. The first step described here is referred as the Steiner points minimization.

Since the objective of processing the Steiner points is to lessen the deterioration in mesh quality, naturally, the second operations on the Steiner points are to reposition them to better locations. Once $F$ is recovered, we move the remaining Steiner points to new positions via the Advancing front points placement method so that the placement is in line with the giving sizing field. For the case of a single Steiner point with respect to $F$, the operation can be easily performed. The Steiner point is repositioned to an optimal location so as to generate an optimal element. For the case of multiple points, more steps need to be taken. One approach is to perform local constrained Laplacian smoothing to the inserted Steiner points. During the smoothing, the points are always constrained in their Balls [17] in order to keep the validity of the triangulation. If the points can be moved considerably from the initial positions, the inner field refinement (see the later meshing scheme) will inevitably improve the quality of the surrounding elements connecting them. On the other hand, if the freedom for moving them is limited, such as in a small cavity, it is obviously more difficult to improve the situation. Moreover, the repositioning of the Steiner points can also be performed in the final mesh optimization by mesh smoothing or by the recently developed CVT methodology [19].

To apply the above constrained boundary recovery together with the post-processing of Steiner points to full 3D Delaunay mesh generation, we again assume that the input is a surface triangulation of the domain boundary, and a sizing field is given or can be derived from the surface triangulation by interpolation. Naturally, the surface mesh should conform with the sizing field. A complete meshing scheme goes as follows.

**Full 3D Delaunay Mesh Generation Scheme:**

(a) Input the surface triangulation.
(b) Perform the initial Delaunay tetrahedralization in the classical Delaunay insertion kernel using points in the surface triangulations and some other specified points if they exist.
(c) Perform the proposed 3D constrained Boundary recovery to the initial unconstrained tetrahedral mesh, together with the Steiner points post-processing.
(d) Generate inner field points by coupling AFT method with the constrained Delaunay insertion procedure, insert them into the existing mesh iteratively until the sizing field is satisfied[10].
(e) Optimize the mesh with local or global geometrical and topological operations, or using CVT optimization[19].
(f) Output the mesh.

The above scheme is applied to mesh various complex geometric models. In the following, we present four meshing examples. The four geometric models are: a box containing nine stiffness (spheres) which is used for composite material multi-scale simulation; a complex geometry with many intersecting spheres, which is for simple molecular surface modeling [22]; a complex geometry with several small cavities and large sizing variation; and lastly the Gulf2 airplane model for external flow simulation. The examples includes pictures of surface triangulation, cutting views of constrained boundary and the full 3D triangulation. They are presented in Fig17-21. Also, two tables are given to present the recovery data and statistics on the element quality.

In Table1, the geometric data for each example are provided which include the total number of vertices in the triangulation, the vertices of the surface triangulation (i.e., boundary vertices), the number of missing boundary faces in the initial tetrahedralization, and the number of recovered missing faces through the simple swappings (abbr.SW), i.e., through the first step. We also report the number of intersection points added in the conforming boundary

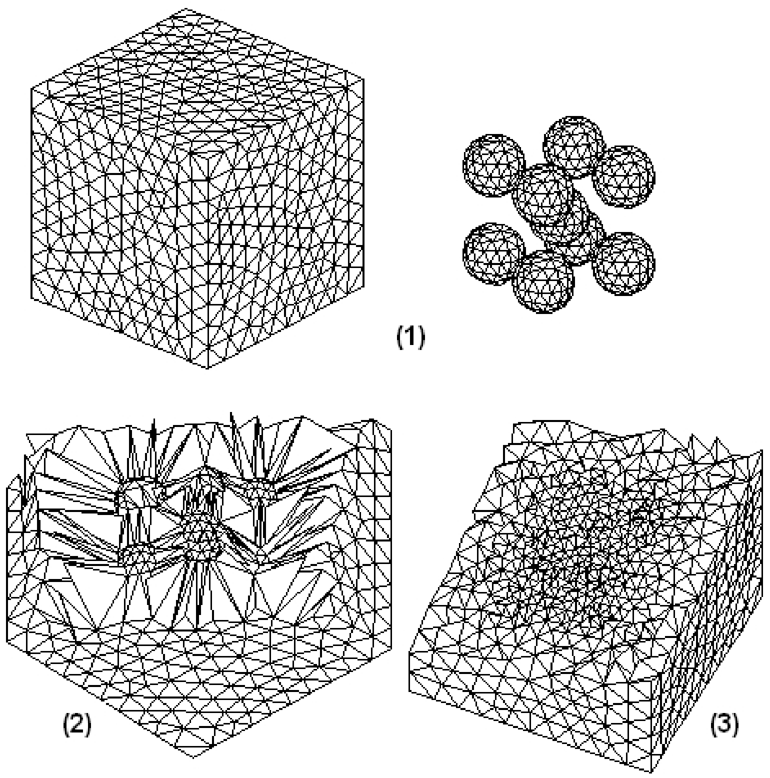

Figure 17. A box included with nine stiff inclusions (spheres): (1) the surface triangulations of the outer box and the inner spheres; (2) cutting views of the constrained boundary tetrahedral mesh; and (3) the final full 3D tetrahedral mesh.

recovery (CBR), the number of Steiner points generated inside the domain and remained after suppression. Moreover, the table also contains the ratios (in percentage) of the CPU timing of the boundary recovery step to the whole meshing process of various methods.

For the four examples, a large portion (almost 70 percent) of missing faces are recovered by the three basic swapping operations, which demonstrates that the first step in our combined constrained boundary recovery is very effective and worthwhile. For the remaining missing faces, in example1, 2 points are added for the conforming boundary recovery of 4 missing faces, and then the points splitting generates 4 Steiner points. And we find these missing faces all appear on the inner sphere surface, hence, these 4 Steiner points are all inside the domain, as we want to mesh both sides of the inner sphere surface. Clearly, we find that vertices in general cannot be suppressed through re-tetrahedralization. Nevertheless, with pure luck, we find they are all moved to optimal positions with the AFT technique, which will be desirable for the full high-quality mesh generation. In example2, there are 9 points being inserted for the conforming recovery of missing faces. After the recovery, the same number of Steiner points are generated inside the domain, and another 9 Steiner points are outside the domain, which will not contribute to the subsequent interior refinement. Also, 4 Steiner points are deleted by

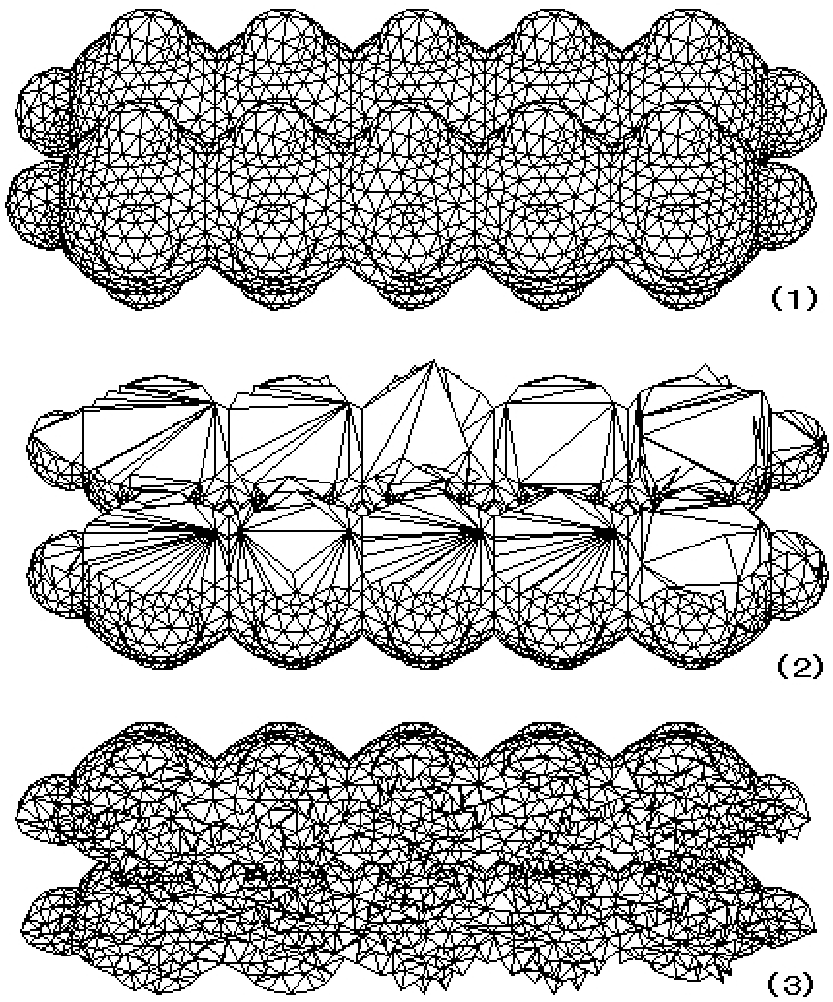

Figure 18. An intersection of spheres for simple molecular surface modeling: (1) the surface triangulation; (2) cutting views of the constrained boundary tetrahedral mesh; and (3) the final full 3D tetrahedral mesh.

suppression and the remaining 5 points are all repositioned to good locations through AFT placement. In these two examples, after careful examination, we find that for each missing face, one only needs to generate one or two (on either side of the face) Steiner points. The case that more than two Steiner points are needed has never occurred in our numerical tests, which is largely due to the smooth sizing field and a high-quality surface triangulation. This is in sharp contrast with the example 3 where 63 Steiner points are generated in the domain and in the end, 38 of these points still remain inside. We find that there are two cases for which two Steiner points are generated for each missing face, and only one point has been processed with the AFT technique while a small perturbation is performed to reposition the other point.

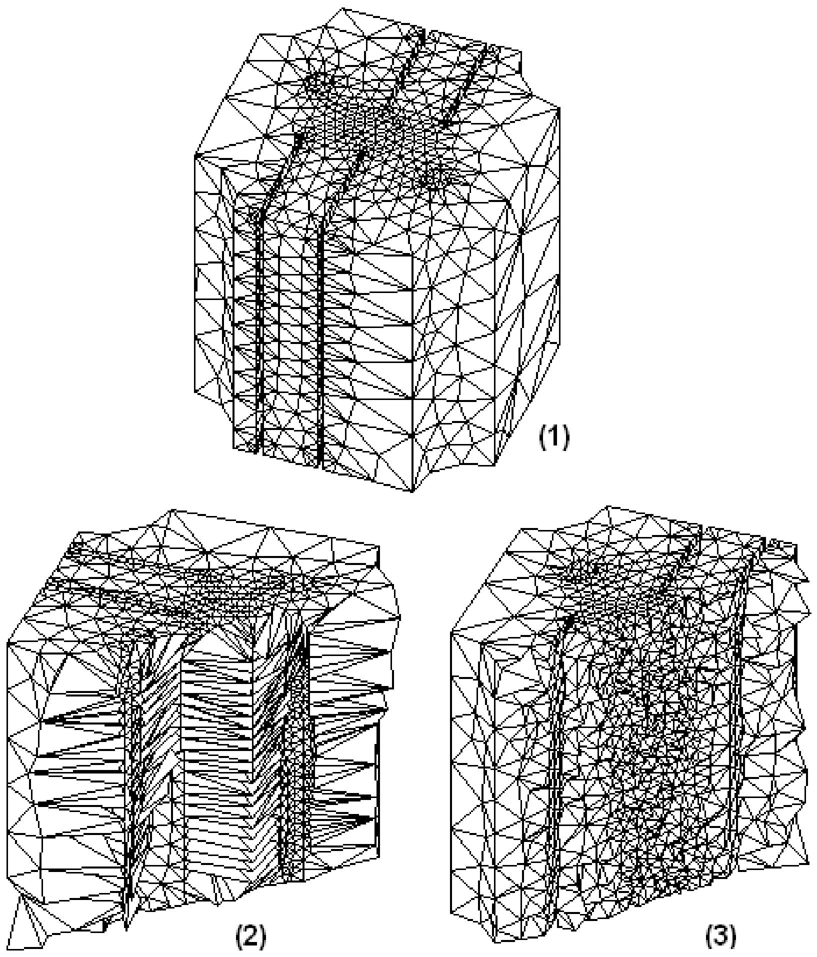

Figure 19. A complex geometry with several small cavities: (1) the surface triangulation; (2) cutting views of the constrained boundary tetrahedral mesh; and (3) the final full 3D tetrahedral mesh.

These cases happen near the part of the surface where there is a large sizing variation. But, after the final CVT optimization [19], these points were repositioned to much better locations. The final example, the Gulf2 model has a larger surface triangulation which has 31633 facets, and there are 620 faces missing after the initial Delaunay tetrahedralization. This is largely due to the existence of a lot of thin triangles in the input surface triangulation. With our recovery technique, 184 Steiner points are added inside the domain and 72 of them remain after the minimization. However, as the model is used for the external flow simulation, there is substantial freedom in moving these points around. With the AFT technique, they are all repositioned to optimal places.

We see from the last line of the Table1 that the CPU time ratio of our boundary recovery takes up only a very small fraction of the total meshing time, and its effectiveness is evident.

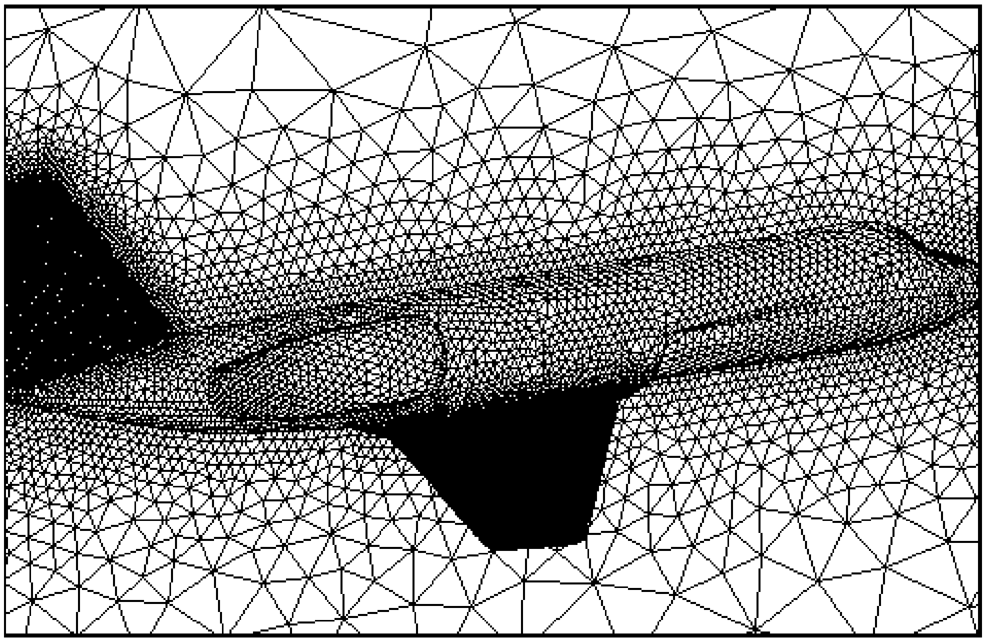

Figure 20. The Gulf2 airplane modeling: the surface triangulation

Even though we make no comparison with other techniques [10, 17] here, the efficiency is clearly satisfactory for real applications. All of the above analysis and simulation results support our theoretical prediction that the proposed boundary recovery provides a good and efficient 3D constrained boundary recovery scheme and a full 3D high-quality meshing.

Table I. Recovery statistics of the examples.

| Examples | Example1 | Example2 | Example3 | Example4 |
|---|---|---|---|---|
| Total vertices | 4856 | 12545 | 9466 | 276653 |
| Boundary vertices | 1466 | 3442 | 3107 | 31633 |
| Original missing faces | 14 | 64 | 225 | 620 |
| Recovered faces through SW | 10 | 50 | 142 | 408 |
| Points added in CBR | 2 | 9 | 63 | 184 |
| Steiner pts generated inside domain | 4 | 9 | 63 | 184 |
| Steiner pts remained inside domain | 4 | 5 | 38 | 72 |
| CPU ratio of BR/meshing(%) | 2.4 | 3.8 | 6.9 | 1.6 |

In Table2, some statistics of elements quality (after optimization through the construction of CVDT [19]) measured by the method of [19] are provided. They include the average quality, minimum quality, ratio of *good elements*, ratio of *bad elements*, and the number of *bad elements* connecting the added Steiner points. Here, the quality of each element [19] ranges from 0 to 1.0, with 1.0 being optimal. And we say an tetrahedron is a *good element* if its quality is above 0.4,

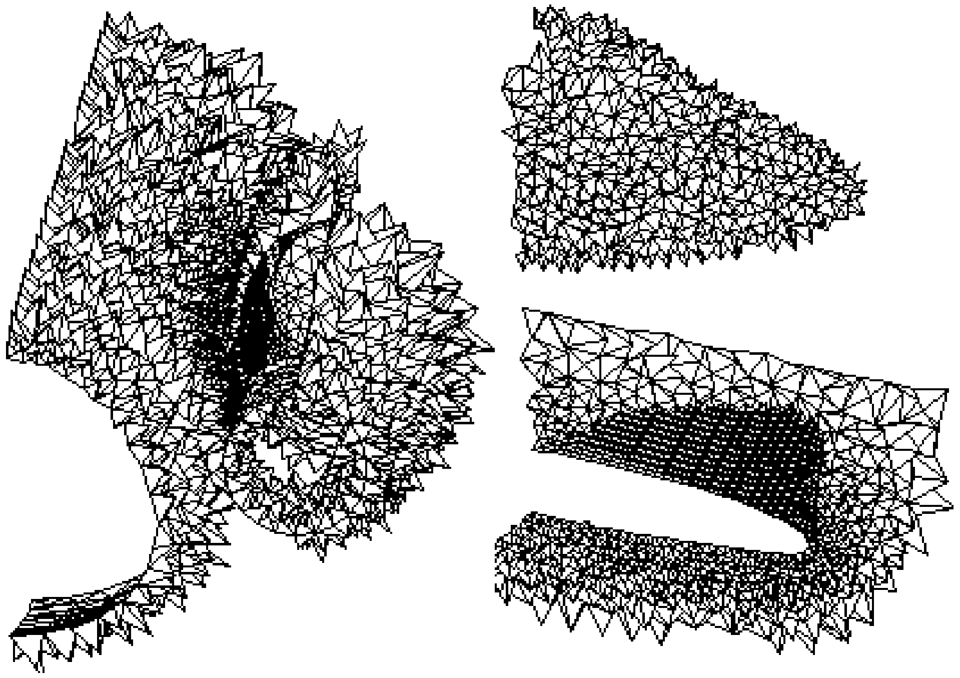

Figure 21. The cutting view of several parts of the final full 3D tetrahedral mesh of the Gulf2 model.

and a *bad element* if less than 0.1. The data of the first two examples are all very satisfactory. In the second example, there are several bad elements close to the surface boundary. For the third example, due to the large sizing variation, the average quality is only 0.702 even after CVT optimization. And a portion of the bad elements appear near the boundary. The Gulf2 model also has a fraction of bad elements near the input sliver triangles. However, in all of the four examples, none of the bad element is connected with the generated Steiner points. This is because we have either repositioned them to optimal locations through AFT techniques or through the final CVT optimization. These data demonstrate that our treatment of Steiner points is adequate and practical for the full three dimensional mesh generation.

Table II. Elements quality statistics of the three examples

| Examples | Example1 | Example2 | Example3 | Example4 |
|---|---|---|---|---|
| average quality | 0.773 | 0.771 | 0.702 | 0.766 |
| minimum quality | 0.124 | 0.073 | 0.041 | 0.032 |
| percentage of *good elements* | 98.2 | 93.4 | 90.5 | 96.8 |
| percentage of *bad elements* | 0.0 | 0.01 | 0.12 | 0.005 |
| *bad elements* connecting SP | 0 | 0 | 0 | 0 |

Based on our numerical examples, we also see that in order to obtain a high-quality 3D mesh, the sizing field of the surface triangulation should avoid large variation. This is achievable through surface remeshing which is an active current research field.

From the recovery data obtained for the examples, another observation can be made: better geometric and curvature control in the surface triangulation results in smaller number of missing faces, and also better positions for the added Steiner points. Hence, for efficient boundary recovery, not only the sizing variation of the surface triangulation matters, the curvature variation also influences the resulting constrained boundary recovery. Take the second example, the variation of the sizing field is not large, however, the ratio of missing faces is not small. With some detailed checking, we find that these missing faces are almost all near the parts of the surface where spheres intersect, corresponding to where the curvature undergoes large changes. Had the surface mesh been curvature-adapted, that is, larger density when curvature is large, the number of missing faces can be reduced, and complexity in treating the Steiner points also goes down, e.g., suffice in using the AFT techniques. The same observation is also confirmed by the third example. Future investigations along these lines will be pursued.

## 7. Conclusion and future work

A theoretically guaranteed algorithm for 3D constrained boundary recovery is presented, which combines conforming boundary recovery with Edge/Face-Added-Points Splitting. In the Splitting of added points, the Cavity is appropriately chosen such that some basic flippable configuration are generated, and the corresponding transformation of these configuration results in a partial or complete recovery of the constraint. The proposed recovery scheme solves the open problem of the 3D constrained boundary recovery by providing answers to several key issues: successful recovery for all cases without any heuristics, natural generation and placement of Steiner points through points splitting, minimizing the number of Steiner points through vertex suppression, and optimizing Steiner points through AFT points placement or other smoothing techniques. Various meshing examples and their geometrical and elements quality data demonstrate the effectiveness and robustness of the presented boundary recovery algorithm and its application to full mesh generation. Future directions of research include the further optimization of Steiner points, the extension to higher dimensional cases, and the application to 3D anisotropic meshing. As suggested in [21], we will also consider the adding AFT-related recovery to the initial simple swapping operations.

## Acknowledgment

The authors are grateful to the anonymous referees and Prof.Oubay Hassan of University of Wales Swansea for the valuable suggestions and comments that helped us to improve the presentation of this manuscript.

## REFERENCES

1. D. Watson, Computing the n-dimensional Delaunay tessellation with applications to Voronoi polytopes, Comput. J. 24, 167-172(1981).
2. A. Bowyer, Computing Dirichlet Tessellations. Computer Journal 24(2):162-166, 1981.

3. H. Borouchaki and P. George, Aspects of 2-D Delaunay Mesh Generation, Int. J. Numer. Meth. Engng.,40, 1957-1975 (1997).
4. N. Weatherill, The integrity of geometrical boundaries in the two-dimensional Delaunay triangulation, Comput. Appl. Numer. Methods 6, 101-109 (1990).
5. H. Borouchaki, P. George and S. Lo, Optimal Delaunay Point Insertion, Int. J. Numer. Methods, Engrg., 39, 3407-3437 (1996)
6. H. Borouchaki and S. Lo, Fast Delaunay Triangulation in Three Dimensions, Comput. J. Numer. Methods. Eng., 128, 153-167 (1995).
7. A. Rassineux, Generation and Optimization of Tetrahedral Meshes by Advancing Front Technique, Int. J. Numer. Methods, Engrg.,41, 651-674 (1998).
8. M. Shephard and M. Georges, Automatic Three-dimensional Mesh Generation by the Finite Octree Technique, Int. J. Numer. Methods Engrg., 32, 709-749 (1991)
9. R. Lohner and P. Parikh, Generation of Three-dimensional Unstructured Grids by the Advancing-front Method, Int. J. Numer. Methods Engrg.,8,1135-1149 (1988)
10. P. J. Frey, H. Borouchaki and P. L. George, 3D Delaunay Mesh Generation Coupled with an Advancing-front Approach, Comput. Methods Appl. Mech. Engrg., 157, 115-131 (1998)
11. A. Liu and M. Baida, How far flipping can go towards 3D conforming/constrained triangulation, 9th International Meshing Roundtable 2000.
12. P. George, F. Hecht and E. Saltel, Automatic Mesh Generation with Specified Boundary , Comp. Meth. Appl. .Mech. Engrg.,92,169-188 (1991).
13. N. Weatherill and O. Hassan, Efficient Three Dimensional Delaunay Triangulation with Automatic Point Creation and Imposed Boundary Constraints, Int. J. Numer. Methods Engrg., 37, 2005-2039 (1994).
14. J. Shewchuk, Delaunay Refinement Mesh Generation, Ph.D. Thesis. Computer Science Dept. Carnegie Mellon, Univ. 1997.
15. B. Karamete, M. Beall and M. Shephard, Triangulation of arbitrary polyhedra to support automatic mesh generators, Int. J. Numer. Methods Eng., 49, 167-191 (2000)
16. J. Wright and A. Jack, Aspects of three-dimensional constrained Delaunay meshing, Int. J. Numer. Methods Eng., 37,1841-1846 (1994).
17. P. George and H.Borouchaki, Delaunay Triangulation and Meshing: Application to Finite Elements. Hermes, Paris, 1998.
18. J. Schewchuk, Constrained Delaunay Tetrahedralization and Provably Good Boundary Recovery, Proceedings, 11th International Meshing Roundtable, pp.193-204, 2002.
19. Q. Du and D. Wang, Tetrahedral Mesh Generation and Optimization Based on Centroidal Voronoi Tessellation, Int. J. Num. Method. in Eng. Vol56, No.9, pp.1355-1373 (2003).
20. Q. Du and D. Wang, Boundary Recovery for 3D Conforming Delaunay Triangulation, accepted by Computer. Methods in Mech. Eng., 2003.
21. O. Hassan, private communication, 2003.
22. P. Laug and H. Borouchaki, Molecular Surface Modelling and Meshing, Engineering with Computers 2002, 18: 199-210.