INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2003; **58**:1061–1089 (DOI: 10.1002/nme.808)

'Ultimate' robustness in meshing an arbitrary polyhedron

P. L. George[1,*,†], H. Borouchaki[1,2] and E. Saltel[1]

[1]*INRIA, Projet Gamma, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France*
[2]*UTT, GSM-LASMIS, Université de Technologie de Troyes, BP 2060, 10010 Troyes Cedex, France*

SUMMARY

Given a boundary surface mesh (a set of triangular facets) of a polyhedron, the problem of deciding whether or not a triangulation exists is reported to be NP-hard. In this paper, an algorithm to triangulate a general polyhedron is presented which makes use of a classical Delaunay triangulation algorithm, a phase for recovering the missing boundary facets by means of facet partitioning, and a final phase that makes it possible to remove the additional points defined in the previous step. Following this phase, the resulting mesh conforms to the given boundary surface mesh. The proposed method results in a discussion of theoretical interest about existence and complexity issues. In practice, however, the method should provide what we call 'ultimate' robustness in mesh generation methods. Copyright © 2003 John Wiley & Sons, Ltd.

KEY WORDS: mesh of a polyhedron; triangulation; Delaunay triangulation; facet partitioning; robust mesh generation

1. INTRODUCTION

The corresponding problem in two dimensions, e.g. the existence of a triangular mesh of a given (non-self-intersecting) polygon, has long been addressed. The existence of a solution is a well-known issue proved in a number of references, for instance References [1, 2]. Almost optimal algorithms (with a linear complexity in practice) have been proposed. Proving (exact) linear algorithm has been the topic of a number of studies. In this respect, Chazelles [3] proposed a solution. Nevertheless, this solution is far from being implementable in a computer. On the other hand, meshing a polygon is a common topic and a series of methods have been developed over the last few decades. Popular construction methods include algebraic [4], advancing-front [5], quadtree [6] and Delaunay based algorithms [7].

In three dimensions, proving the existence of a tetrahedral mesh of an arbitrary polyhedron is known to be a tedious problem. Such a problem is reported to have a non-polynomial

*Correspondence to: P. L. George, INRIA, Projet Gamma, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France.
†E-mail: paul-louis.george@inria.fr

Contract/grant sponsor: E.E.C, MAGIG-FEAT; contract/grant number: 1999-11433

Received 9 May 2002
Revised 17 October 2002
Accepted 20 November 2002

complexity [8, 9]. A key question is to decide if a triangulation exists with no added points (the so-called Steiner points). Another concern, when such additional points are necessary to ensure the existence, is to decide what is the optimal (minimal) number of Steiner points. Actually, by means of heuristics, several construction methods exist which complete a triangulation without explicitly considering this problem of optimality (i.e. the number of Steiner points is not a concern).

George *et al.* [10] proposed a method using two ingredients. At first a Delaunay triangulation of the points in the surface mesh of the given polyhedron is constructed. Then topological modifications (edge and face swaps) together with point addition (the Steiner points) allow to modify the current triangulation so as to recover the initial facets of the given surface mesh. In terms of theory, this method results in the desired mesh but, in terms of computer implementation, numerical troubles may arise in some tedious pathologies. Examples of such configurations include the case where faces with rather different sizes are close to one another and the case where these faces are strongly anisotropic.

In Reference [11], a similar approach can be found in which the initial facets can nevertheless be modified by means of swaps (in case of coplanarity). In this way, the strong constraint we have in hand is somehow relaxed and the solution is easier to obtain (which does not claim to conform to the initial boundary facets but to a more or less 'equivalent' geometry).

Weatherill *et al.* [12] proposed a method leading to splitting the constrained facets after the field points have been added (as such points are required in the envisaged C.F.D. applications) The underlying idea is that those points facilitate the construction of the sub-facets resulting from the initial facet after has been split. This method completes a mesh of the given polyhedron but, while a phase is used to remove the added points, there is no guarantee that all of them can be successfully removed. In such a case, some of the initial facets may be missing in the resulting mesh.

Chazelle *et al.* [9] proposed a method resulting in a triangulation of a non-convex polyhedron (without holes) with n vertices and r reflex edges in $O(n + r^2)$ tetrahedra. The idea is to partition the polyhedron by means of cylinders whose bases are the triangles in the surface mesh. This paper is of purely theoretical interest and, as far as we know, there has been no actual implementation.

Methods related to the notion of Delaunay admissibility (or Delaunay conformity) must be added to the above list of methods. In this respect, two approaches are popular. For instance, in Reference [13], the constraints are partitioned so as to be enveloped by spheres which guarantees no encroachments from one facet to another. Points are added resulting in this partition. This approach is of a purely theoretical interest as, in practice, the number of added points can be prohibitive. It is not sure that this method is of actual interest. The other approach, [14, 15], is more realistic. The key is to modify the initial surface mesh using a series of edge swaps and adding points in such a way that the Delaunay triangulation of the resulting set of points includes exactly the 'new' surface mesh. In this case, the number of added points is limited as noticed in Reference [15]. Note that the resulting boundary facets are, in general, different to the initial facets.

In this paper, a simple method is proposed to construct the triangulation of an arbitrary polyhedron while maintaining the given surface mesh of this domain. This method resembles, in its principle, that of Weatherill *et al.* but is different in the sense that no field points are *a priori* added and, moreover, it is guaranteed that the added points are purely removed or repositioned outside the initial facets. In other words, the resulting mesh conforms to the initial data.

The schema of the method is described and various application examples are shown to demonstrate the robustness of the method. In Section 2, before considering the three-dimensional case and for clarity, we return to the same problem in two dimensions. Existence issues are discussed, construction algorithms are described and complexity concerns are briefly indicated. In Section 3, existence issues are addressed in three dimensions. Section 4, briefly, recalls the previous (direct) method by George *et al.* (mentioned in Section 1). In Sections 5 and 6, the proposed method is fully described in terms of a construction method and the various steps in the method are discussed. In Section 7, we mention some complexity issues. Then, Section 8 provides a series of demonstrative application examples. To conclude we give some remarks and we indicate some possible (future) applications of the method.

2. EXISTENCE OF A SIMPLICIAL MESH OF A GIVEN POLYGON

We are given a domain defined by its boundary. That boundary is made up of (simple) polygons. Each of them is defined by means of a set of line segments. The existence of a (at least one) triangulation of this domain where the above line segments are element edges is well established. Nevertheless, the following briefly describes three proofs regarding this issue. The first uses a naive method, the second makes use of a Delaunay triangulation algorithm while the third is the restriction to two dimensions of the method which will be proposed in three dimensions.

Let $(S_i^j)_{i=1,n^j}$ be the list of the endpoints of the line segments members of the discretization of polygon j in the domain boundary (n^j denotes the number of points in polygon j). Let us assume that these vertices S_i^j are ordered in the right way (clockwise or counterclockwise according to the parity of the component) in each connected component of the domain boundary.

2.1. A naive construction method

For the sake of simplicity, we consider the case where the domain is convex (with one polygonal boundary), Figure 1 (left-hand side), then a non-convex domain with one polygonal boundary and finally an arbitrary domain.

2.1.1. Convex domain. Such a case is trivial. For clarity, we assume one connected component in the boundary and we cancel index j. We consider the triples made of three

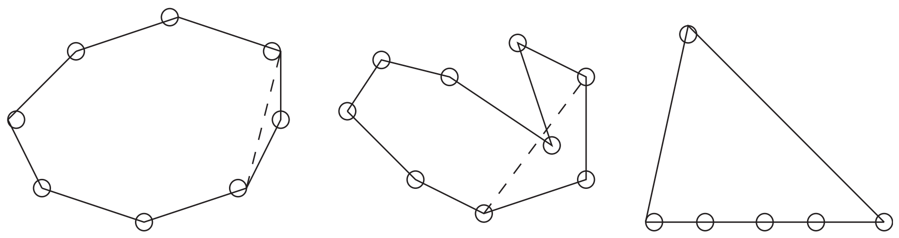
Figure 1. Left-hand side, a convex polygon. Middle and right-hand side, non-convex polygons.

non-colinear consecutive points. As the domain is convex, (at least) one such triple already exists. Let $[S_i, S_{i+1}, S_{i+2}]$ be the triple in hand. The oriented surface of the triangle corresponding to this triple, $\mathrm{Det}(S_{i+1} - S_i, S_{i+2} - S_i)$, is positive (due to a correct orientation of the boundary), therefore triangle (S_i, S_{i+1}, S_{i+2}) can be formed. Then, on the one hand, the number of line segments in the polygon decreases by 1 and, on the other hand, point S_{i+1} is no longer in the list of points. Because the remaining polygon is still convex, the same applies. Therefore, the algorithm terminates and a solution exists.

Notice choosing a suitable triple (thus constructing a valid triangle) may be not unique thus resulting in different solutions. In this respect, an appropriate choice may be used to select the triples in some order so as to guarantee some extent of quality in the resulting triangles.

2.1.2. Non-convex (simple) domain. An arbitrary simple domain (without holes) with one connected boundary component requires a more subtle analysis. In contrast to the previous case there exists at least one reflex vertex where the angle bounded by the two edges sharing this point is greater than π (as seen from the interior of the domain). Such a vertex impedes the above construction, Figure 1 (middle). Indeed, this vertex may fall in a triangle corresponding to a triple which is *a priori* suitable for the previous scheme.

The idea is then to find the list of those reflex vertices before going back to the previous method with some appropriate adaptation. Actually, validating a triple means guaranteeing that no vertex falls inside the region thus defined.

More precisely, we prove that at least one triple (one index i) exists such that:

- $\mathrm{Det}(S_{i+1} - S_i, S_{i+2} - S_i) > 0$, the surface of triangle (S_i, S_{i+1}, S_{i+2}) is positive,
- triangle (S_i, S_{i+1}, S_{i+2}) does not contain any (reflex) vertices.

The proof makes use of the fact that a simple polygon can be triangulated with no extra points (other than its vertices). For every closed simple polygon the number of triangles is $n - 2$ where n is the number of vertices in the polygon. Because the number of edges in the polygon is n, there exist at least two triangles that have two boundary edges.

Once a triple satisfying the two required criteria has been exhibited, the corresponding triangle is formed. After updating the list of points, the process is repeated. Notice that considering a triple may result in forming a non-reflex vertex (with regard to the resulting active boundary) which was reflex before this treatment.

Another construction consists in finding a 'positive' triple and examining whether it contains one or several reflex vertices. If it does not, the related triangle is formed and the number of edges in the polygon decreases. On the contrary, among all the points (in the current polygon) included in the triple $[S_i, S_{i+1}, S_{i+2}]$, we consider that which is reflex and closer, for instance, to segment S_i, S_{i+1}. A valid triangle is formed using this line segment and this vertex as can easily be seen. Introducing this triangle divides the domain into two disjoint sub-domains each of which has a polygonal boundary where the number of edges is strictly less than the number of edges in the previous active polygon.

2.1.3. Arbitrary domain. In such a case, the domain boundary is made up of several simple polygons with arbitrary shape. Then, it is only necessary to use the above method while observing that two components may be linked together when the retained point does not belong to the component where the endpoints of the selected edges live.

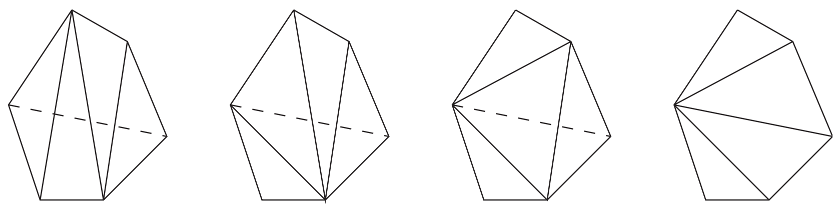

Figure 2. Successive edge swaps to regenerate a missing edge.

In contrast with the three-dimensional case, it should be noticed that meshing a polygon does not require adding a point. The triangle vertices are the points of the given boundary.

2.2. A method using a Delaunay triangulation algorithm

For simplicity,[‡] we define a convex box, say a square, enclosing the S_i^j's. At first, a Delaunay triangulation of the S_i^j's and the four box corners allows us to complete a triangular mesh of this box. In general, this mesh does not include all the line segments (whose endpoints are in S_i^j) in the given boundary. Nevertheless, simply using a number of edge swaps (flips) results in the formation of all these segments which are then element edges, Figure 2. As a consequence, it is now easy to identify those triangles inside the polygon and thus to remove the external triangles. In this way, a mesh of the domain is obtained thus proving the existence of a solution (with no extra points).

The underlying idea is to decrease (by means of edge swaps) the number of triangles intersected by a missing edge. As this is not always true (an edge swap leading this number unchanged), the solution is obtained by randomly swapping the candidate edges in the case where more than one edge may be swapped.

2.3. A partitioning method

This method follows the first step of the previous method. The Delaunay triangulation thus constructed does not generally include the initial line segments. Such an edge is missing (i.e. is not an element edge) because there exist a number of edges in the current mesh which intersect it. The intersection points will be inserted to enforce a partition of the missing edge. Then, these points will be removed so as to recover the initial line segment.

Partitioning a given segment into edges: Let us consider a missing segment. Its endpoints are vertices in the current mesh. Starting from one of these endpoints, A, we look for the first current edge intersected. The intersection point, P_1, is computed and inserted in the mesh, Figure 3. To this end, the two triangles sharing the intersected edge are processed. The first triangle in this pair is formed by vertex A and the intersected edge, the second is formed by this edge and the point opposite this edge. The construction simply subdivides these two triangles. The first is split into two by using AP_1, the second is also split by joining P_1 with the point opposite the common edge in the initial situation. Therefore edge AP_1 is an element edge in the resulting mesh.

[‡]While not being strictly required.

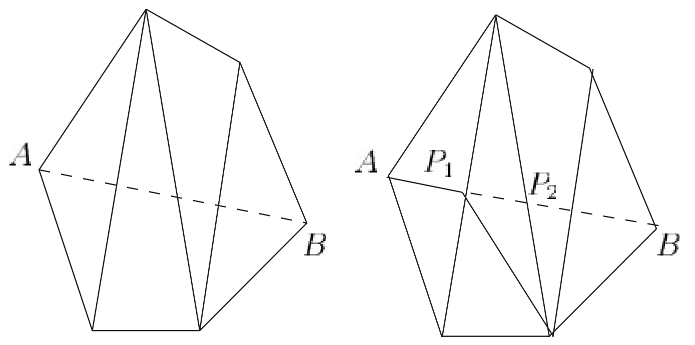

Figure 3. The edge to be enforced (dotted line, left-hand side), introduction of the first intersection point and formation of the first segment of the partition of the missing edge under consideration.

We consider now the two triangles resulting from the partition of the above second triangle. Among their edges we find the one intersected by the line segment in hand. In this way either a new intersection point P_2 is found, to be inserted or we meet one edge joining P_1 to B, the final endpoint of the constraint. In this case, the sought partition is completed, if not, the same construction applies.

As a consequence, introducing the intersection points of the current edges and the missing constraint results in forming a partition of this constraint. Note that the method uses a rather trivial mesh modification tool. Applied to all the constraints, this construction insures the existence, by means of partitions, of all these constraints.

Removing the added points: Now, all the missing constraints being processed, the points added in this partitioning process are removed thus resulting in a mesh with no additional points. Removing the points of a given partition is done in several steps:

- divide the polygon formed by the triangles intersected by the constrained segment into two polygons with disjoint interior;
- for each polygon;
  - ○ sort in decreasing order the vertices boundary of the polygon other than the intersection points introduced in the segment and its endpoints according to their distance from this segment;
  - ○ disconnect by means of edge flips these boundary vertices following this sort until only one point remains;
  - ○ replace the resulting pencil by a triangle with the constrained segment as an element edge.

Disconnecting a boundary vertex involves suppressing all the edges joining this point to the intersection points. To suppress a vertex S in this polygon, the method consists in defining the polygonal region bounded by:

- the edge boundary of the polygon which joins vertex S and its predecessor S^- (for instance, on the left),
- the edge joining S and its successor S^+ (then on the right),
- the edge joining S and the intersection point at the extreme left in the constrained segment,

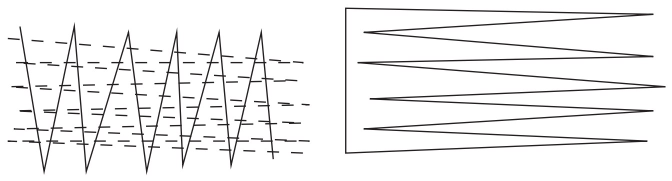

Figure 4. A reasonable example of the worst case. Schematic principle (left-hand side), the Delaunay edges are shown in solid lines, the sought edges are in dotted lines. A peculiar case (right-hand side) with this property when there is a large number of points in the upper and lower boundary sides of this polygon.

- the edge joining S and the intersection point at the extreme right in the constrained segment.

This region is necessarily convex, so it is possible to swap all its internal edges (those joining S with the intersections in the segment) onto S^- or S^+. In this way, S is only connected with S^- and S^+. For example, to swap these edges towards S^-, it is simply necessary to apply some edge swapping from the left to the right.

2.4. Complexity

In general, the naive method is in $\mathcal{O}(n(1+r))$ where n denotes the number of vertices in the boundary polygon and r is the number of reflex vertices. Nevertheless the convex case is linear, i.e. in n.

The second method has the same complexity as the Delaunay triangulation algorithm plus that of the edge swap phase therefore that of this latter. It is then quadratic in the worst case (same number of swaps and current edges for each constraint). The cost is that of a method which constructs a Delaunay triangulation from an arbitrary triangulation.

The last method has the complexity of its partition–elimination phase. To analyse the complexity of the method by partition–elimination we examine the worst case. The difficulty is to define such a case. Indeed we are interested in a case of this nature which is actually reasonable. A naive feeling dictates that such a case would be like 'all the edges in the current mesh intersect all the constraints'. Nevertheless, such a case is not realistic. Indeed, if a constraint (say the first one) intersects all the current edges, it is clear that this cannot be true for all of them.

A potential worst case: One may define a relatively realistic example, Figure 4, where there are n current edges and m missing constraints, each of which intersects the n current edges. To have an idea of such a case, we consider a (relatively flat) quadrilateral and we distribute a number of points (uniformly positioned) along its sides with the same number on each side. Then, adjusting the side lengths, the resulting Delaunay edges clearly join the 'upper' points to the 'lower' points. Assuming the sought edges (the m missing constraints) to be in the other direction then the n current edges intersect all the missing constraints. In this way we have an intuitive idea of what the worst case could be.

Analysis of the partition process: Here, the edges are split so as to recover the sought partitions, i.e. the added points are not removed. We consider as the first missing edge the one that is on the 'top'. Along this edge we add n points. This edge has $n_0 = n$ added points.

From these new points, we add, in the worst case (pencil, if not only the half is needed) n new edges which intersect the other missing edges (those 'below' the edge in hand). Therefore, we have $n_1 = n_0 + n$ added points on the second missing entity. Going on to the next edge, we add $n_2 = n_1 + n/2$ points and so on. To compute the cost, we assume $n_2 = n_1 + n$, that is in the same range, and so on. The number of added points is then:

$$n_0 + n_1 + n_2 + \cdots + n_{m-1}$$

Or again

$$n + n + n + n + 2n + \cdots + n + (m-1)n$$

thus in the range of

$$mn + \frac{m(m-1)n}{2} = \mathcal{O}(m^2 n) \tag{1}$$

For n and m in the same range, we find a complexity in n^3.

Analysis of the partition–elimination process: In this case, the edges are split to form the sought partition but, given a partition, the added points are removed. We consider as the first missing edge the one which is at the 'top'. On this edge, we add n points. Thus this edge has $n_0 = n$ added points and gives the desired partition. By means of edge collapsing, these n added points are removed. We consider the second constraint. It is easy to see that the same process applies, we add again n points (or less). This second constraint having been processed, the n added points are removed in the same way. The m initial line segments are processed in the same way. In terms of (temporarily) added points, the complexity is in mn, thus n^2 for $m = n$. The cost is then in n^2, in terms of the number of points, to which we must add the cost of the elimination phase. Assuming the same cost (i.e. n operations to remove n points) the algorithm globally has a complexity in n^2.

Analysis of a 'divide and conquer' method: A subtle choice of the first missing edge leads to dividing the problem into two sub-problems of smaller size. For the first missing edge we add n points, thus n new edges but, as above, half of these new edges are located in the half-plane above the edge in hand and half in the other half-plane. Therefore, repeating this process leads to the same complexity as above but with a lower constant. However, in practice, choosing the appropriate edge is not so trivial (see below a way to have a chance of meeting this case on average while avoiding returning to the case of the previous methods).

Computer implementation conclusions: While not being of actual interest in two dimensions (as a number of various well-known meshing methods are currently available), the above discussion allows us to find some ideas to facilitate a possible computer implementation (in the view of the three-dimensional case).

The aim is then to see how the partition method could be implemented. To this end, a number of ideas could be of interest:

- clearly, minimizing the value of m, even in the worst case, minimizes the global cost. To do so, the partition method is only applied after other classical methods have been used to enforce the missing entities (these methods being supposed to have a lower cost).

- avoiding the worst case, e.g. the creation of cascade of points on all missing edges (or half of them). Two approaches can be advocated in this respect:
    - randomly processing the missing entities,
    - removing a maximum of added points on one edge, after its treatment, to minimize the cascade effect. Removing a point impedes the creation of new edges which could intersect a not yet processed edge.

To conclude at this point, these few remarks about a computer implementation certainly lead to a reasonable cost in most cases. Therefore, the above ideas will be retained when the problem in three dimensions will be considered.

3. ABOUT THE EXISTENCE OF A SIMPLICIAL MESH FOR AN ARBITRARY POLYHEDRON

In three dimensions, the problem reads as follows:

- we are given a triangular mesh of the surface of the arbitrary domain,
- the question turns out to know whether there exists a tetrahedral mesh of this polyhedron such that the triangles in this surface mesh are element faces.

This problem is known to be NP-hard. This does not mean that there is no solution (actually, a number of methods exist which allow to complete a mesh even in severe geometries. In this respect, methods like those for two dimensions are currently available [7, 16, 17]) but the construction cost is not guaranteed to be polynomial although, in practice, it is in most cases. Moreover, in contrast with the two-dimensional case, it is known that a mesh without internal points (the Steiner points) does not exist in some instances. Before entering further into the discussion, let us give two examples, each highlighting a particular difficulty. The first example shows that even a simple geometry may require adding a Steiner point to be triangulated. It also shows the difficulty of finding such a point (i.e. properly locating this point). The other example illustrates the difficulty of bounding the number of elements required to triangulate an arbitrary domain (Figure 5).

3.1. The Schönhardt prism

This polyhedron is nothing more than a cylinder with two triangular bases, it has 6 vertices and 8 faces, [18]. The facet at the bottom is a triangle as is the facet at the top. The three lateral facets are quadrilaterals each of them being split into two triangles. Let us assume a straight prism, Figure 6, then, based on the way the lateral (planar) quads are split, a mesh exists (made of three tetrahedra) which is easy to obtain or no such mesh exists (one of its tets being null (in terms of volume)). Nevertheless, adding a single point inside the domain allows a solution. In this way we obtain a mesh with 8 tetrahedra (one for each of the boundary facets). The key is that all these boundary facets are visible by this point. Moreover, choosing such a point is easy as any point inside the domain is a suitable candidate.

Let us twist the prism, the bottom base being clamped we rotate the upper facet around the axis of the cylinder while maintaining a valid prism (with no self-intersection). The problem involves finding a point inside the domain which is visible by its 8 boundary faces. In theory,

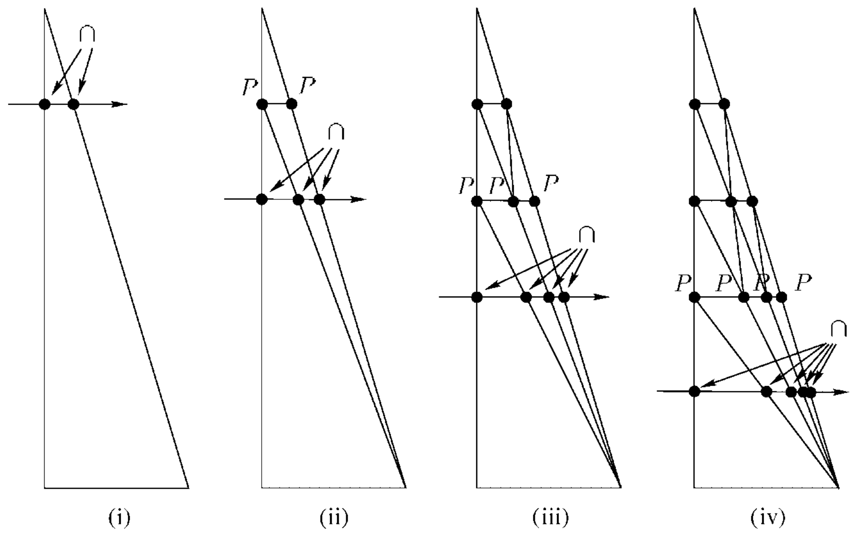

Figure 5. From left to right: (i) a Delaunay triangle and its intersections (∩) with the first missing edge; (ii) introduction by means of a local remeshing of the two intersection points P and creation of some new intersections (∩) with the following missing edge; (iii) and (iv), propagation of the phenomenum (creation of new intersections after each local remeshing).

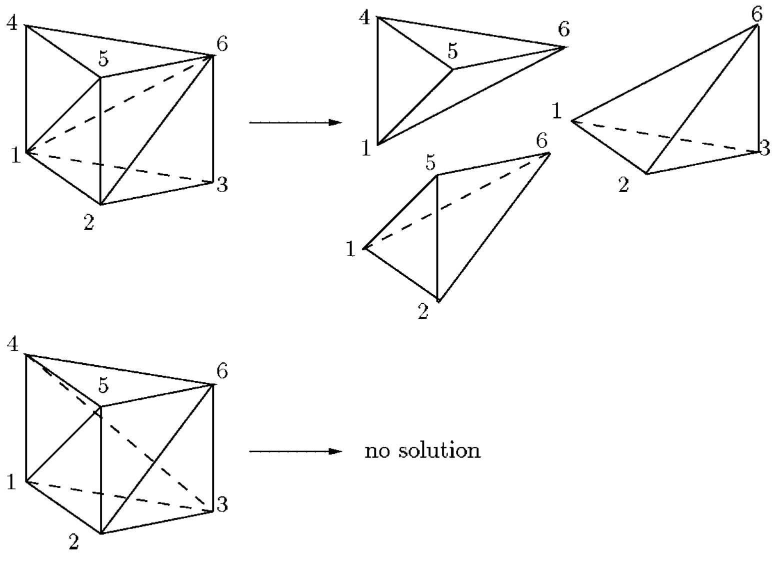

Figure 6. The Schönhardt prism: constrained triangulation of a regular prism leading to a valid partition (with no added points) or leading to an impossible configuration where no triangulation exists if no point is added.

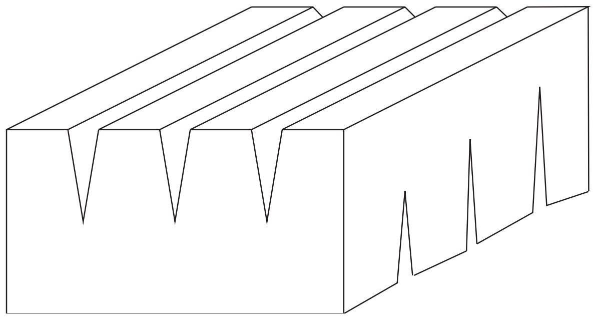

Figure 7. The Chazelle polyhedron: the geometry illustrating the notches' distribution. To simplify the view, we display only three notches and the scale is not accurate.

as long as the prism is valid, a non-empty visibility kernel exists (it is the region intersection of the half-spaces bounded by the domain facets). However, in practice, finding a suitable point is numerically tedious.

3.2. The Chazelle polyhedron

One of these polyhedra (there is a series of such polyhedra), Figure 7, is defined as a straight parallelepipedon such that x and y range from -1 to $n+1$ (thus a length of $n+2$, n being an integer value) and such that $-2n < z < 2n^2$ thus bounding the upper and lower facets. Each of these facets includes $n+1$ notches evenly spaced. At the top, the notches are formed by an entering edge parallel to the y-axis and incident to two facets that are almost vertical. These notches have their lower edges in the hyperbola $z = xy + \varepsilon$, $\varepsilon > 0$. At the bottom, the notches have their entering edges parallel to the x-axis and are defined by $z = xy$, see Reference [19].

It is reported (and proved in various references, [2, 19], among others) that triangulating this polyhedron requires a quadratic number of elements if $\varepsilon < 1/n^2$. This is due to the fact that the proximity between the lower and upper facets (in the notches) leads to visibility kernels whose volume is in the range of ε.

3.3. Arbitrary polyhedron

An arbitrary polyhedron is a domain where, locally, patterns with some extent of regularity (aligned points, co-planar regions, cocircular items, cospherical items, etc.) and with configurations similar to the two pathological cases above. Therefore, meshing such a domain may be tedious or costly.

4. NAIVE METHOD AND DELAUNAY BASED METHOD

The naive method as seen in two dimensions does not work in the case of an arbitrary (i.e. non-convex) polyhedron as proved in the example in Figure 6.

The second method, Delaunay triangulation and generalization of the edge swap has been studied in a number of references, for instance Reference [10]. The idea is to triangulate a (convex) box enclosing the domain by means of inserting the 8 corners of this box and the points in set S_i^j using a Delaunay triangulation algorithm. At completion we have a triangulation of the above box where some of the facets defining the boundary of the domain may not be formed (while their endpoints are element vertices).

Facet swapping (between two adjacent tetrahedra) and generalized edge swapping (in the polyhedron made of the tets sharing such an edge) combined with the addition of Steiner points (if necessary, see the above prism) allow us to regenerate the missing facets. As these facets are in the triangulation of the box, it is easy to classify those tetrahedra inside the domain and those outside the domain. Therefore, removing the tets outside the domain results in a mesh of the given polyhedron.

In Reference [10], it is proved that the above method completes a valid solution. This is true in theory apart from the numerical problem. In practice, a solution is obtained in most cases. It is nevertheless easy to construct examples where the computer understanding of the theoretical algorithm fails. Such a case occurs when at least one initial facet fails to be (re)formed and therefore it is impossible to define the domain in the triangulation of the box. Notice that such cases correspond to geometries where the boundary of the polyhedron is triangulated in a coarse manner (shocks exist in the size of facets geometrically close, and we meet the 'edge-hog' effect reported to be dramatic in Delaunay type methods) or where strongly anisotropic facets are in the surface mesh of the domain. As a consequence, the method, while intensively used with success in various engineering problems, is not able to address all meshing situations. This is why we propose a new method which could be seen as an alternative method or could be coupled with the previous in order to attain what we called 'ultimate' robustness in the title of this paper.

5. A CONSTRUCTIVE METHOD TO PROVE THE EXISTENCE OF A SOLUTION

In this section, we propose a method which, while appearing simple, is able to successfully deal with all the cases including those causes of failure mentioned above. This method is the mechanical extension of the partition–elimination method described in two dimensions (thus justifying our description of this simple case). The existence of an appropriate solution (a mesh that conforms to the given surface mesh) is obtained because our method is a constructive method. In other words, the method completes a mesh and thus a solution exists. The construction method is made up of three steps:

(1) constructing a Delaunay triangulation based on the points of the boundary of the polyhedron;
(2) seeking the faces (in the initial surface) that are missing in this triangulation, constructing the intersection points of these facets with the current facets and inserting these points using a local insertion process;
(3) suppressing the above added points in order to retrieve the initial surface mesh of the polyhedron.

These three steps are detailed in the following sections.

5.1. Delaunay triangulation

The boundary points (endpoints of the surface triangles) are inserted, one at a time, using a Delaunay triangulation algorithm. For simplicity, a convex box is created which encloses all these points and the triangulation algorithm completes a mesh of this box. The resulting triangulation does not generally contain all the surface triangles as element facets. Therefore, a domain mesh is not possible to exhibit from the box mesh.

5.2. *Partitioning the boundary facets*

A given facet is missing because there exist one or a number of current edge(s) (face(s)) with an intersection with this entity. We therefore split these edges (faces) by introducing the intersection points, [20]. At first, the missing edges are processed then the missing faces will be considered. A given edge is missing because:

- it is intersected by some faces in the current mesh,
- it is intersected by some edges in the current mesh.

Introducing the points intersection with the sought edge with the current faces, we locally modify the mesh by splitting the two tetrahedra which share the intersecting face into six tetrahedra. Similarly, introducing the points intersection of the sought edge with the current edges, we locally modify the mesh by splitting the tets sharing the intersecting edge into tets. This involves replacing the shell (the set of tets sharing the edge) by two sub-shells. This process is repeated for all the missing edges and, at completion, the initial edges exist in the current mesh in the form of a partition.

We examine whether the facets induced by these partitions locally form a partition of the initial facets. In general, cases occur where a part of a given initial facet is missing (while looking these facets one at a time). This results from the fact that there exists at least one current edge which intersects the region interior to the constrained facet. As above, we compute the corresponding intersection points and we insert them. This simply involves splitting the related shells into two sub-shells. Once all the given facets have been processed, we have a mesh in which all the initial facets exist either unchanged or as a partition. Therefore the domain is fully meshed, and a solution exists but this is not the desired solution (due to the partitions). The new step is then to remove these partitions so as to recover the initial surface mesh.

5.3. *Enforcing the boundary facets*

The points added in the previous step must be removed in some way. We will show that it is possible to remove these extra points either by *suppressing* them or by *relocating* them outside the constraints. This will result in the sought solution: a mesh that conforms to the surface mesh. In this section, for the sake of simplicity, we assume the specified facets to separate the domain from its exterior (thus the domain has only one connected component). The general case will be addressed in Section 6.

Suppressing a point in a face: Let P be a point introduced in given face F and let $\mathscr{B}_P$ be the ball of P (those tets with P as a vertex). While face F exists (being partitioned), there exists in ball $\mathscr{B}_P$ a set of tets, $\mathscr{B}_P^{\text{in}}$, fully inside the domain (this 'half-ball' is the intersection of the half-space including the domain with the full ball of P. This 'half-ball'[§] has as

[§]We use this notation for simplicity.

external facets:

(i) the facets in the partition of F which define an arbitrary polygon in the plane of F,
(ii) the facets other than the previous ones which are inside the domain.

Because this half-ball exists, point P is visible from all the facets of type (ii). Therefore a positive volume polyhedron exists enclosing P, and is the visibility kernel of those facets of type (ii). As a consequence, point P can be relocated anywhere in this kernel. The point is then such that:

- the facets of type (ii) are visible from P,
- point P also sees the plane of F. As the above polygon is triangulable by means of triangles (Section 6), point P is visible from all these triangles.

As a result, the half-ball under consideration has been replaced by a valid ball. To conclude, a point in the partition of F has been removed. Repeating the same, all the vertices introduced in this partition can be removed, one at a time.

Suppressing a point in an edge: Now, the boundary points of the partition must be removed. These points are those added to the boundary of any partition, e.g. they are the points added on the boundary edges of the initial facets. The principle of the method is the same. The only difference is that the initial edge is common to two initial facets which are not necessarily in the same plane. We restrict ourselves to the case of manifold edges, where one edge is common to two facets (the other cases, an edge is common to 0, 1 or more than two facets will be discussed in Section 6), F_1 and F_2. After this restriction, we can find a half-ball, $\mathscr{B}_P^{\text{in}}$, fully contained in the half-space where the domain is. This ball has as external facets:

(i) the facets in the partition of F_1 which define an arbitrary polygon in the plane of F_1,
(ii) the facets in the partition of F_2 which define an arbitrary polygon in the plane of F_2,
(iii) the facets other than the previous ones which are inside the domain.

Because this half-ball exists, point P is visible from all the facets of type (iii). Therefore a positive volume polyhedron exists enclosing P, and is the visibility kernel of those facets of type (iii). As a consequence, point P can be relocated anywhere in this kernel. The point is then such that:

- the facets of type (iii) are visible from P,
- point P also sees the plane of F_1 and that of F_2. As the two above polygons are triangulable in triangles (Section 6), point P is visible from all these triangles.

In this way we have constructed a valid ball for point P thus relocated. To conclude, one point boundary of the partition of F_1 and F_2 has been removed. By the same argument, all the vertices added in a partition of an initial edge can be removed, one at a time (Figure 8).

Conclusion: By applying the two above methods we have demonstrated that all the introduced points can be removed from the constrained entities. The final mesh includes some extra points but the initial facets are now present in this mesh. However, in practice, some added points may disappear after some appropriate edge collapsing.

Remark: Note that the uncancelled points which have been relocated outside the constraints are nothing more than the Steiner points needed to allow the construction of a suitable mesh.

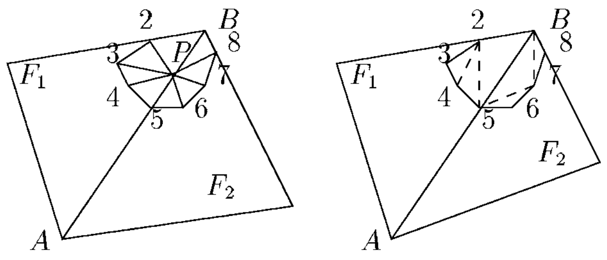

Figure 8. Configurations before and after removing point P previously added in edge AB. In the planes of F_1 and F_2, the two facets sharing edge AB, polygons $B2345$ and $B5678$ are found and remeshed without P while resulting tets are created.

Moreover, among these points, those outside the domain will be automatically cancelled when defining the domain (i.e. when removing the external tets).

6. TECHNICAL DESCRIPTION OF THE PARTITION–ELIMINATION METHOD

In this section, we return to some previously mentioned points in terms of computer implementation. In this respect, we discuss how to compute the intersection points involved in the construction, and we describe some powerful point insertion processes. We also take a look at cases where the edges in the constraint are common to 0, 1 or more than 2 initial facets. We discuss some procedures that make it possible to complete a valid mesh (with positive volume elements) by processing a mesh where null or even negative volume tets exist (as occurs in practice when relocating or duplicating a point). Then we conclude with some remarks about the robustness of the proposed method.

6.1. Intersection detection and computation

One ingredient used in the partition method is to detect and subsequently compute the intersection points between the current entities and the sought constraints. These intersections concern two cases, the intersection of a current edge or facets with a sought edge and those with a sought facet.

Let AB be a constrained edge, we pick a tet with A as its vertex and we examine the tets in the ball of A. Let $\alpha\beta\gamma$ be the face opposite A in one such tet, then we can define three (virtual) tets around AB with $\alpha\beta$, $\beta\gamma$ and $\alpha\gamma$ as other vertices. The sign of the three corresponding volumes makes it possible to decide whether AB cuts $\alpha\beta\gamma$ or not. In case, those volumes are the barycentric coordinates of the intersection point. This point $P_1 = P$ being computed, it is immediately inserted into the mesh (see next section about this point insertion) and the same is repeated (while P replaces A) to find the possible other intersections. At completion, line segment AB exists and reads:

$$AB = AP_1 \cup P_1P_2 \cup \cdots \cup P_nB$$

where n is the number of tets intersected by AB.

We turn now to computing the intersections with a given facet. This is more subtle, Figure 9. After the edge partition, we face cases where some of the initial facets have been recovered

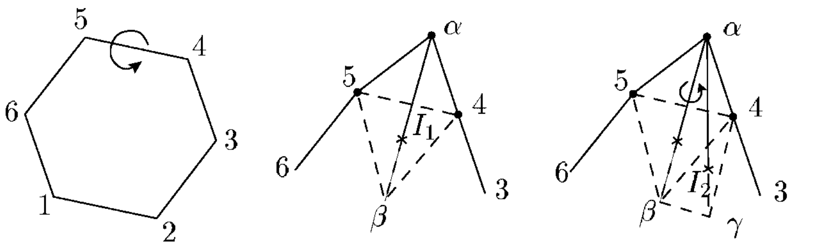

Figure 9. Schema to find those current edges intersecting a polygon. To this end a stack is considered. Edges 12, 23, 34, 56 and 61 are put in the stack when considering edge 45 which is cancelled from the stack. Processing the current stack we meet tet $45\alpha\beta$ where edge $\alpha\beta$ cuts the plane of the initial facet in point I_1. This new edge is put in the stack. When processing this edge, we find another element where edge $\alpha\gamma$ cuts this plane in I_2, this edge is put in the stack, etc.

by a partition (in sub-facets) or where still exists a number of current edges intersecting such facets. Therefore a polygon exists (part of the initial facets) traversed by edges. The schema in Figure 9 illustrates how to find these edges. Once these edges have been found, intersection points are inserted (see below). It can be observed that detecting and computing the intersection points requires making sure that the current mesh is strictly positive (in terms of element volumes). Therefore, inserting such intersections must maintain this property (while furthermore accepting to (temporarily) have null or negative elements).

6.2. Point insertion procedures

Intersection points have been found on current edges and facets. Therefore, a local insertion method allows, in principle, for a valid mesh. When a point is to be inserted in a tet face, it is *a priori* sufficient to consider the two tets sharing this face. Let $(\alpha\beta\gamma)$ be this face, and let $A\alpha\beta\gamma$ and $\alpha\beta\gamma B$ be those two tets, then inserting point P on $(\alpha\beta\gamma)$ results in constructing the following six tets:

$$\alpha AP\beta, \beta AP\gamma, \gamma AP\alpha, \alpha PB\beta, \beta PB\gamma \quad \text{and} \quad \gamma PB\alpha$$

When a point is to be inserted in a mesh edge, we consider the shell of this edge. Let $\alpha\beta$ be the edge in hand, then the above shell reads:

$$M_i\alpha\beta M_{i+1} \text{ for } i=1,n \quad \text{with} \quad M_{n+1}=M_1$$

Inserting point P in $\alpha\beta$ leads to constructing two shells:

$$M_i\alpha PM_{i+1} \quad \text{together with} \quad M_iP\beta M_{i+1} \quad \text{for} \quad i=1,n \quad \text{with} \quad M_{n+1}=M_1$$

Shell $\alpha\beta$ has been replaced by shells (αP) and $(P\beta)$.

In theory, the two above point insertion processes result in a valid mesh (provided the initial configuration is valid). In practice, we meet examples where negative tets are formed. This is mainly due to a numerical reason. To explain such a case, let us return to a rather obvious case. Let AB be an edge and let P be a point computed as the intersection of AB with another edge. Then, in theory, we have:

$$P=\omega A+(1-\omega)B \tag{2}$$

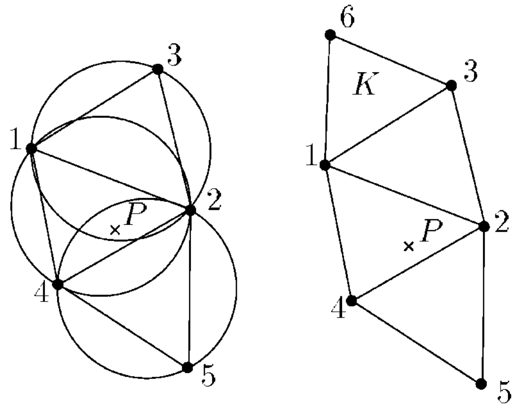

Figure 10. Schema (in two dimensions) of the generalized point insertion method. Left-hand side, point P, to be inserted, falls in triangle 142, and that triangle is put in the cavity of P. The circumdiscs of triangles 123 and 452 also contain P, these two elements are put in the cavity and the latter is now completed since no other elements violate the Delaunay criterion. Right-hand side, we add triangle K which is such that, while not violating the Delaunay criterion, its edges 16 and 36 are visible from P. Inserting P with this cavity results in a non Delaunay mesh which is valid.

where ω is in $]0,1[$. Floating point operations lead to approximate values, then P is not strictly in AB. Now, inserting P using a local process (as above) leads to negative tets if there exists a tet facet separating P and AB. This is why local insertion processes must be carefully checked and, if necessary, other more powerful processes (i.e. less sensitive to round-off errors) must be used. Two of these are advocated. The first is the well-known Delaunay insertion procedure which reads:

$$\mathscr{T}_{+} = \mathscr{T} - \mathscr{C} + \mathscr{B} \tag{3}$$

where $\mathscr{T}$ is the current triangulation, $\mathscr{C}$ is the *cavity* of P and $\mathscr{B}$ is the ball of P, e.g. the set of tets formed by joining P to the external facets of the cavity. The cavity is obtained according to the Delaunay criterion. This point insertion process has proved to be more robust than a simple local process. To obtain a more efficient procedure, we developed a generalized Delaunay-type point insertion method. The idea is to stack in the cavity the elements found by the Delaunay criterion and, evenly, some others as depicted in Figure 10. This generalized point insertion process is definitively more robust than any others. Moreover, such a method works even if the current triangulation is not Delaunay [21]. Actually, we make use of this powerful method and the limits of the global method are those of this point insertion process.

6.3. Non-manifold edges

Constrained edges were supposed to be common to two constrained facets in the above discussion. To handle concrete situations we need to consider the cases where an edge is a member of only one such facet (referred to as a onefold edge) and those where a edge is common to more than two facets (threefold, fourfold, etc.). Such situations make it more delicate to apply the method to suppress a point added on a edge by processing not only two polygons (but only one or more than two). Actually, there are only two cases, the first where

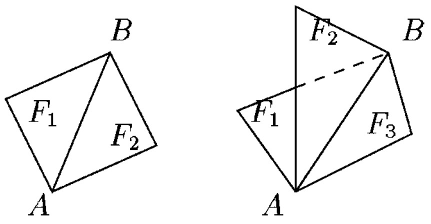

Figure 11. A 'twofold' edge shared by two boundary facets and a 'threefold' edge common to three boundary facets.

there are 2 or more prescribed facets sharing the edge, and the case where the edge is only one side of one specified facet.

6.3.1. Edge member of two or more facets. Let us consider a threefold edge which is illustrative of the general situation. Let F_1, F_2 and F_3 be the three initial facets sharing this edge and let P be a point added on this edge.

At first, we exhibit three polygons, one in each plane (of a facet). Second, we cancel P by triplicating this point in P_1, P_2 and P_3, each of them in the space delimited by the three facets. Then we mesh the three polygon into triangles where P is no longer a element vertex. To conclude, we join these triangles with P_1, P_2 and P_3 (resp.). In this way we have defined three half-balls and the resulting mesh is such that point P is no longer in the edge in hand. On the other hand, we have cancelled a point while triplicating it.

One issue is to find a proper location for P_1, P_2 and P_3. As this is tedious, we simply triplicate P and therefore P_i is coincident with P. This results in null (or negative) tets. Due to this, the mesh is no longer valid and methods must be developed to correct the mesh (Figure 11).

6.3.2. Edge member of only one facet. Now, the case of a onefold edge is more tedious. Indeed, it is no longer possible to find two half-balls since the ball of point P in hand is not separated into two disjoint parts by two facets. To carry out this pathology, we first consider the triangles in the mesh of the polygon related to the only facet we have. Then among the element facets, we guess those which can emulate a polygon on the other side of the edge under treatment. This allows us to return to the case of a twofold edge and then two half-balls are easily defined and the normal method applies (P is duplicated, etc.).

6.4. Processing the non-positive elements

After the method to deal with the edges, we have certainly created null or negative tets. Therefore these elements must be processed to correct the mesh. Available tools include topological tools (face or edge swap), metric tools (point repositioning) and some others (adding a point).

Nevertheless these tools (popular for optimizing a mesh) must be considered with great care since the current mesh includes null or negative tets. Figure 12 illustrates a case where the mechanical use of an edge swap results in a mesh which is topologically invalid. As a conclusion, face or edge swap must be directed not only by an objective function (see below) but a topological check must also be considered.

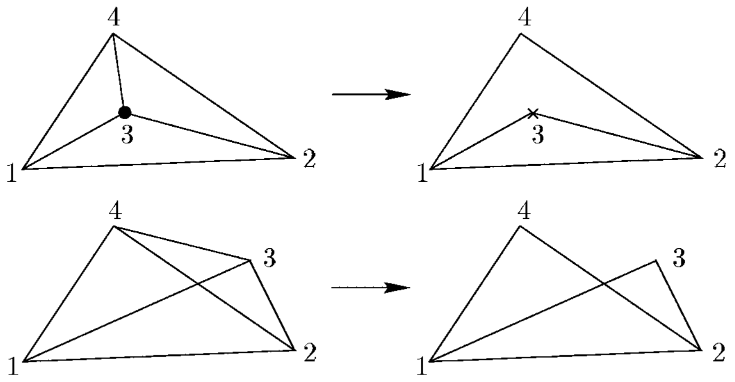

Figure 12. A two-dimensional example to demonstrate that an edge swap may result in a non-valid mesh. Top, flipping edge 43, common to triangles 134 and 324, forms triangles 124 and 123. The latter (just forget it is negative) already exists in the initial pattern. Therefore the present swap is not valid in terms of topology. Bottom, the same case but point 3 is such that triangle 324 is negative. Flipping edge 34 results in two new positive triangles but again forms triangle 123, this flip is therefore not a valid operation.

Now we turn to how to reposition the points (bearing in mind we have duplicated some points and must find an appropriate location of the so-multiple points). The method we advocate is a classical relaxation method which reads:

$$P = \omega P + (1-\omega)P^* \tag{4}$$

as can be found in Reference [7]. The issue is to define P^* in such a way that the new location of P results in a valid mesh. To this end we consider the three cases which are encountered in our context. Let P be the point to be repositioned, let $\mathscr{B}$ be its ball made up of a number of tets, K_i (whose volume is denoted by V_i). Then:

- $V_i > 0$ for all the K_i in $\mathscr{B}$. The objective function is a classical quality function such as $Q = csteh/\rho$ where h is the element diametre, ρ is the inradius and $cste$ a scaling constant. Using this function, point P_i is computed as the point which optimizes tet K_i while moving P to P_i. Using the P_i's, point P^* in the formula is set to their centroid.
- at least one V_i is negative in $\mathscr{B}$. Following [22], the objective is to minimize $\sum_{K_i \in \mathscr{B}} |V_i|$ but we find it more effective to replace this objective by considering $\sum_{K_i \in \mathscr{B}, V_i < 0} |V_i|$. With each K_i negative is associated a point P_i and as above P^* is the centroid of those P_i. Point P_i is defined by the symmetry of P with respect to the face opposite P, which reads as

$$P_i = P - 2(\overrightarrow{AP} \cdot \overrightarrow{n})\overrightarrow{n} \tag{5}$$

  where $\overrightarrow{n}$ is the unit normal to the face opposite P and A is one corner of this face.
- at least one V_i is null in $\mathscr{B}$. Considering the K_i with positive volume in the ball, we compute P_i to be the centroid of the face in K_i opposite P. Then P^* is the centroid of those P_i.

In practice, negative tets are processed first, then null tets and then a classical optimization method (with a quality function) is used to improve the resulting valid mesh.

Again after [22] and in our experience, combining face and edge swaps, point repositioning and point addition when a problem allows for a valid mesh. Regarding negative elements, the underlying idea is that a valid mesh is one such that $\sum_{K_i} |V_i| = \sum_{K_i} V_i$, and thus a valid mesh minimizes the volume (in absolute value) of the corresponding domain.

6.5. Robustness issues

Robustness concerns have been seen on the fly in the previous sections. One concern was that of being able to insert a point in the current mesh. The other was to successfully make a mesh positive when there are a number of negative tets. The first case was carried out by means of generalizing a known point insertion method. The other was considered by means of optimization tools reviewed for the envisaged situation. To conclude, the robustness of the method is clearly an inherently important feature of the way in which the constraints have been considered. Examples of extremely severe pathologies demonstrate how robust the method is. It could be observed that robustness is not attained using sophisticated arithmetic systems (infinite or variable precision, etc.) and thus is not dependent on the computer used to construct the mesh.

7. BRIEF COMPLEXITY ISSUES

As in two dimensions, defining the worst case is not so easy. Therefore, for analysis purposes, one looks for a case whose complexity is (*a priori*) rather bad but which, on the other hand, remains realistic.

A potential worst case: Similar to the two-dimensional case, it is possible to define a case where there are n current facets and $m = n$ missing facets (e.g. the same number). This example represents a relatively realistic worst case. To have an idea of such a case. we consider a straight parallelepipedon (of adequate size) and we distribute (evenly spaced, for instance) a (equal) number of points on its lower and upper faces, Figure 13. We also prescribe a number of points on two of the lateral faces, which are opposite, thus forming in two horizontal planes only two triangular facets. Then adjusting the height of the domain, the Delaunay

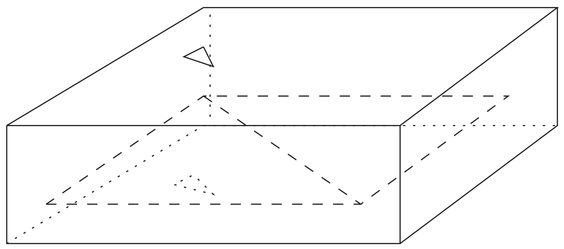
Figure 13. A specific example of the worst case in three dimensions. We assign a large number of points on the upper and lower faces of this parallelepipedon (only two of the resulting triangles are shown) and we specify a number of faces on a number of planes parallel to the two faces above (only one of these planes and the two related triangles are shown).

facets (resulting from the triangulation of the set of points in hand) clearly joins some points in the upper face with some points in the lower faces. We then assume the constrained facets to be in the other direction (say, on a horizontal plane). Therefore, the current faces intersect all the sought faces. In this way we have an idea of what the worst case could be.

Coarse analysis: For the sake of simplicity, we consider the case where the mesh of the upper face of the domain reads as a uniform grid with $n+1$ points in one direction and $n+1$ points in the other. Thus, we have $2n^2$ small triangles on the upper (resp. lower) face. Since the connections are made between these triangles, the number of faces traversing the domain is in the range of n^2. There are about n^2 intersections with the first two horizontal faces we meet (those in the upper plane for instance) and there is the same number of intersections with the other horizontal faces in the successive horizontal planes (whose number is supposed to be $m=n$). If we introduce these intersections on the first plane, we create a sub-mesh of the two faces in hand which is of the same range (in terms of the number of triangles) as the mesh at the top of the domain. This being completed. we add a number of new intersections with the faces in the remaining successive planes. Therefore, similar to the two dimensions, we face, for the edge intersections, a process which add intersections on an increasing number of faces. With no more subtle analysis, the complexity is at least as bad as in two dimensions. Even for the edges, (i.e. only the edge partitions are considered), we are in n^4, n^3 per plane, for instance $y=constant$, times n, the number of planes. The edges being partitioned, if the initial faces are adequately split, we are still in n^4. If some sub-facets are still missing (i.e. there still exist some current edges intersected by some recent new faces), then the partitioning process continues and the complexity becomes worse and worse.

Actual comment: After this coarse analysis, the worst case definitively has a bad complexity. Nevertheless, we have implemented the method assuming the worst case (or a similar one) not to be of actual interest and assuming that concrete cases, even complex, do not lead to such cases. The meshes we obtained in a dozen of complex examples allow us to conclude that the cost (thus the complexity) remains reasonable.

Remark: Eliminating the points added when partitioning the constraints is, on the one hand, more tedious than in two dimensions (edge collapsing) and, on the other hand, not always possible (the Steiner points). Nevertheless, suppressing these points in the fly, when appropriate, avoids the cascade effect previously mentioned and the complexity is not as bad as in the worst case.

8. APPLICATION EXAMPLES

To demonstrate the method, we have selected a series of application examples that are representative of the geometries we wish to process. Before going further, one could observe that 'classical' methods successfully run for reasonably defined geometries (as is required in finite element style meshes). Therefore the proposed method is only of interest for rather badly discretized domains. It is for this reason that the following examples are of this type.

8.1. Two academic cases

To demonstrate the capabilities (in terms of robustness) of the discussed method we have built two 'academic' examples to easily show some tedious situations. These examples naturally

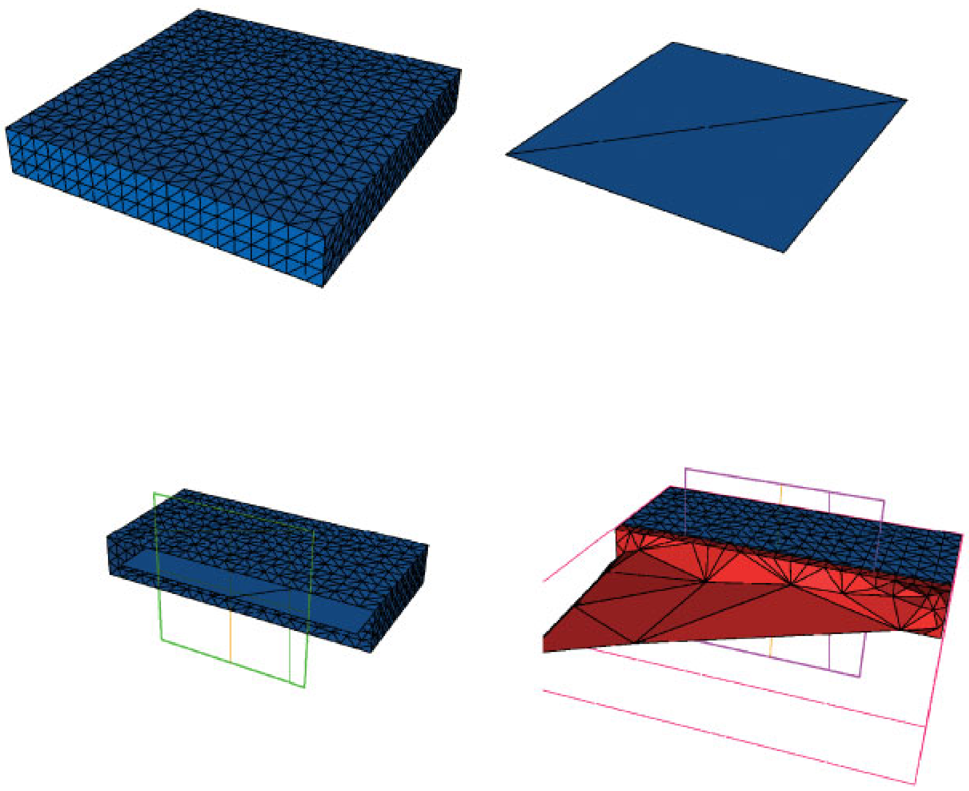

Figure 14. Top left, the boundary surface of the polyhedron where the triangulation is made up of small triangles. The bottom face of the domain is similarly triangulated. Top right, two large triangles are prescribed by the mid-plane of the domain. These two triangles do not touch the domain boundary as can be seen on the top left (by means of a cut). Bottom right, a cut through the resulting tet mesh where the tets are shown. Note the 'flat' tets due to the two prescribed triangles and the 'reasonable' tets close to the upper small boundary triangles.

include the so-called 'edge-hog' effect (where large sized faces are close to small sized faces) and therefore are judged extreme for a classical mesh generation method. The first example includes two dangling triangles thus leading to a number of 'onefold' edges (such an edge being a member of only one prescribed face). The second example shows a case where the prescribed faces are dramatically ill-shaped. In this respect pencils of edges are included in the model where the angles between two consecutive edges are rather small.

Before going in details with these two cases, let us comment about the Schönhardt prism. Following the notations in Figure 6, the proposed method results in creating point P intersection of edges 15 and 24. This allows for the completion of a valid mesh with P as a vertex. Now P is moved towards the interior of the prism creating a cavity (a pyramid with P as apex and 1254 as basis). This region is meshed by means of two tets following the proposed procedure.

The first example, Figure 14, corresponds to a surface mesh made up of 2082 triangles and 1046 vertices. Only five edges (two faces) are missing before entering the partition–elimination phase.

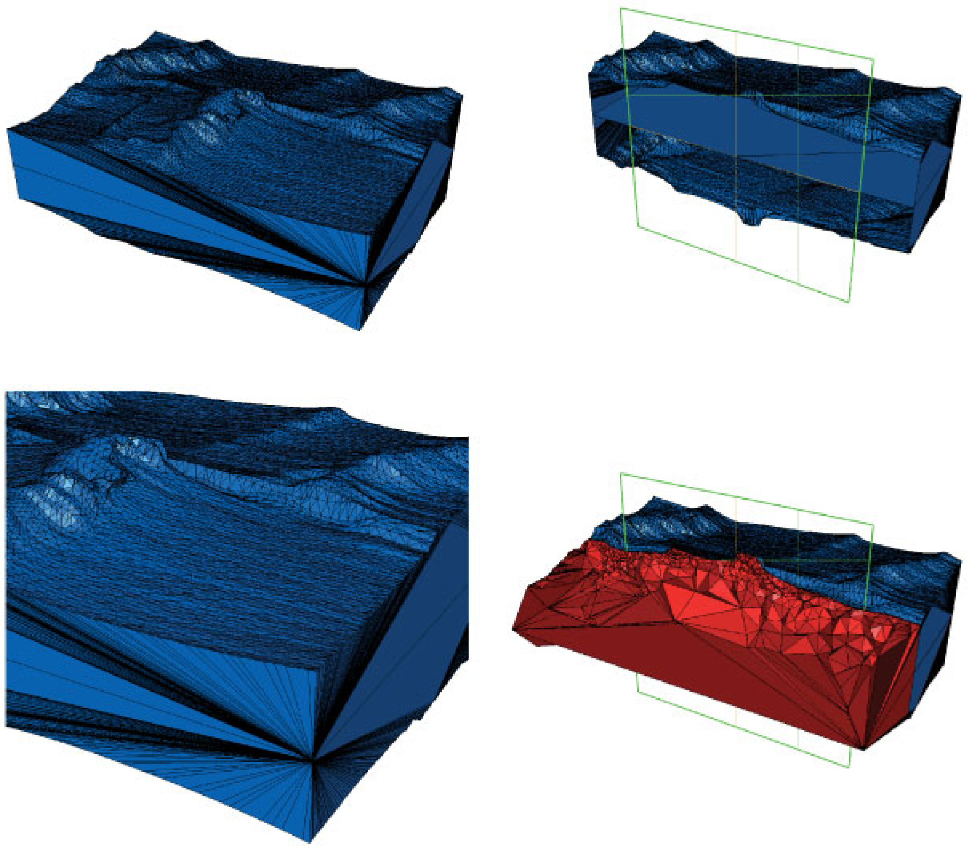

Figure 15. Top left, the boundary surface of the polyhedron where the triangulation is made up of small triangles. The bottom face of the domain is similarly triangulated. Top right, two large triangles are prescribed by the mid-plane of the domain. These two triangles touch the domain boundary as can be seen on the top left (by means of a cut), therefore we meet a threefold case. Bottom left, a view of the pencil of edges close to a corner of the domain (to emphasize the difficulties in this discretized geometry). Bottom right, a cut through the resulting tet mesh where the tets are shown. Note the 'flat' tets due to the two large prescribed triangles and the 'reasonable' tets close to the upper small boundary triangles.

44 intersection points are inserted when partitioning those five edges. 531 and 532 points are necessary to partition the two missing faces. After duplicating the points, 122 tets are null or negative. After moving the points and optimizing this mesh, a valid mesh is completed. It is made up of 3937 tets and 1273 vertices including 37 Steiner points. The mesh quality is 808 while the targeted value is 1.2. The construction time is 25.33 s. After one optimization step (in 2.17 s) the quality becomes 40.

The surface mesh in the second example, Figure 15, includes 26 326 triangles and 39 487 edges. The number of missing edges is 12, the number of points for the edge partitioning is 57. More than 4000 intersection points are necessary to partition each of the two large triangles inside the domain. Duplicating the intersection points results in only 2 null or

Table I. Statistics for the concrete examples.

	faini	*arini*	*faper*	*arper*	*npface*	*nparet*
Ex 1 (mohne)	5560	8340	12	9	80	30
Ex 2 (thru-mazewheel)	5622	8433	1	0	3	0
Ex 3 (cami1)	932	1398	7	7	27	13
Ex 4 (try5.2)	13 482	20 119	6	3	120	11

Note: Number of missing faces and edges, number of intersections temporarily added.

Table II. Statistics for the concrete examples.

	npste	*ne*	*np*	*Target*	*Q*	*Cpu*
Ex 1 (mohne)	42	16 839	4208	627	737	27.90
Ex 2 (thru-mazewheel)	16	8377	3160	73	253	7.46
Ex 3 (cami1)	14	1720	546	734	755	6.43
Ex 4 (try5.2)	10	56 460	10 767	530	1230	29.06

Note: Number of Steiner points, mesh description, quality and cpu.

negative tets. The final mesh includes 110 329 tets and 25 407 vertices, 57 of them being Steiner points. The final mesh quality is 41 430. The construction time is 311.16 s (to be improved!).

8.2. Concrete examples

The three following examples were found in the repository *edge.mcs.drexel.edu*. This repository collects a series of examples (surface meshes) which are not generally suitable for finite element analysis and, therefore, is a source of instructive tedious examples.

The last example was collected from the E.E.C. project MAGIC_FEAT who partially funded this work.

Table I reports some significative figures about these concrete examples. The number of constrained facets is *faini*, that of constrained edges is *arini*. The number of missing facets (after the classical enforcement step) is *faper*, that of missing edges is *arper*. *nparet* and *npface* stand for the number of intersection points added when partitioning the missing entities. In Table II, the resulting tet meshes include *npste* Steiner points, *ne* tets and *np* vertices. The cost (s) is denoted by cpu. Then *Target* is the targeted quality (that of the best tet based on the worse boundary triangle), the quality measure is that indicated in the text. It ranges from 1 (regular tet) to infinity (flat element with null volume).

The number of missing entities (after the classical enforcement method) remains low. In other words, the partition–elimination method is used for a small number of constraints. The number of intersection points remains low in these examples (whereas we have met cases where more than a thousand of such points were necessary, for only one face!) (Figures 16–19).

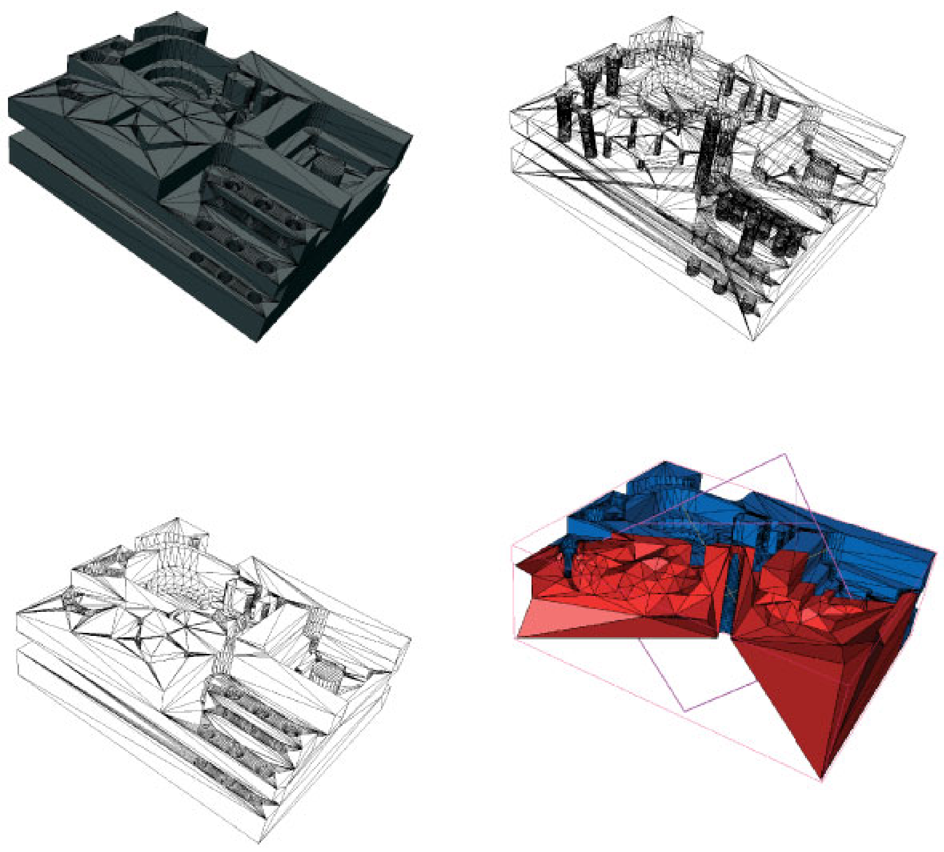

Figure 16. This polyhedron is defined by a surface mesh made up of a minimal number of triangles, so that the geometry is well captured. Top left, facets in the boundary mesh, top right, the edges boundary of these facets. Bottom left, hidden line view. Bottom right, a cut through the resulting tet mesh.

The resulting quality, Q, is often worse than the targeted value, *Target*. This results from the Steiner points (after duplication) which constrain the mesh. Cpu time may be improved (here, we paid great attention to robustness). The number of Steiner points is certainly not optimal (minimal) because robustness and mesh quality were given greater importance. Therefore, some of these points were not removed as the mesh quality was controlled.

9. CONCLUSION

We have presented a partition–elimination method that makes it possible to mesh an arbitrary polyhedron while conforming to the surface mesh defining its geometry. The existence and complexity of the method have been investigated. Technical computer issues have been addressed mentioning a number of new techniques (generalized insertion, optimization tools, etc.). The robustness of the method has been analysed and demonstrated by a number of

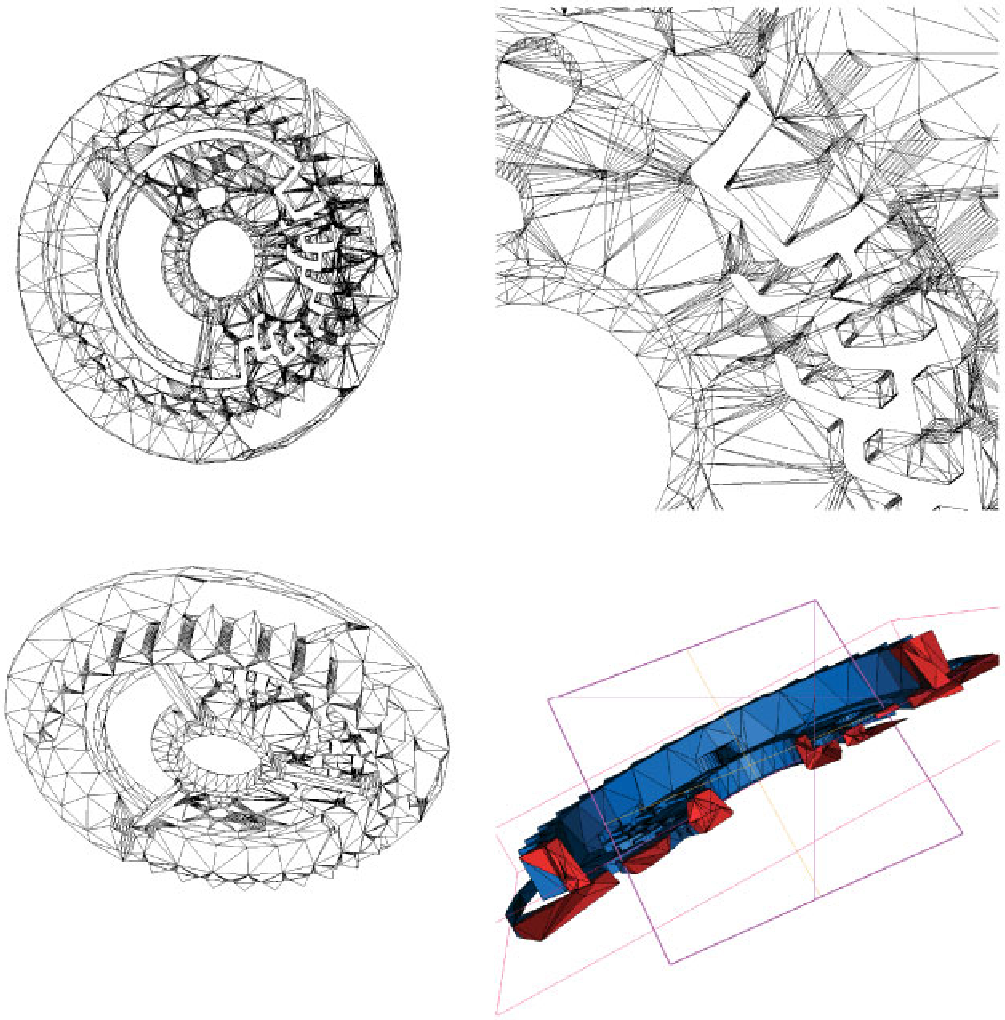

Figure 17. The same views as in the previous figure for a different polyhedron (geometric model of a gear).

tedious application examples. Concrete applications of the proposed method are multiple and the underlying ideas can be re-used in various engineering problems.

An easy to understand application is to combine this method with a classical mesh generation method such as a Delaunay based method. For cases where the classical method fails to form a boundary facet, the present method allows for a valid solution. Then, the surface facets being formed, other steps can be successfully performed including the generation of the necessary field points. In this way meshes of high quality can be completed even for an ill-shaped definition of the domain boundaries. Dealing with strongly anisotropic

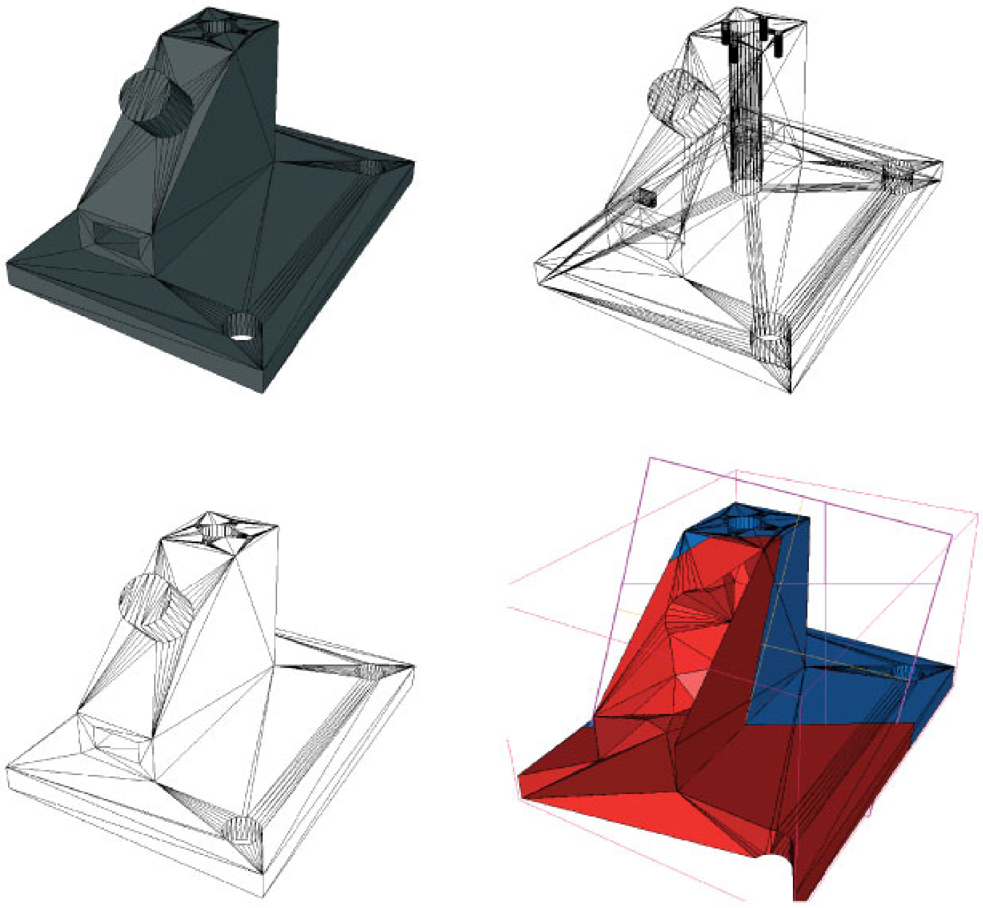

Figure 18. The same views as in the previous figures for a different polyhedron (geometric model of a mechanical part).

surface meshes is also an issue which, in essence, is similar to the previous case. Anisotropic facets are indeed considered to be ill-shaped for a Delaunay style mesh generation method.

Exotic applications can be envisaged including the mesh construction of domains defined by the so-called STL surfaces (where the triangles in the surface may be extremely flat). Detecting self-intersection in a surface mesh can be envisaged thus enabling us to correct such a mesh, while not suppressing the added points that are strictly required to ensure the mesh conformity. Making a given surface mesh conformal (where edges are not common to two facets) is also a possible application together with pasting together two meshes which share a common region which is not triangulated in the same way in each of them. Therefore, this paper may have various interseting applications.

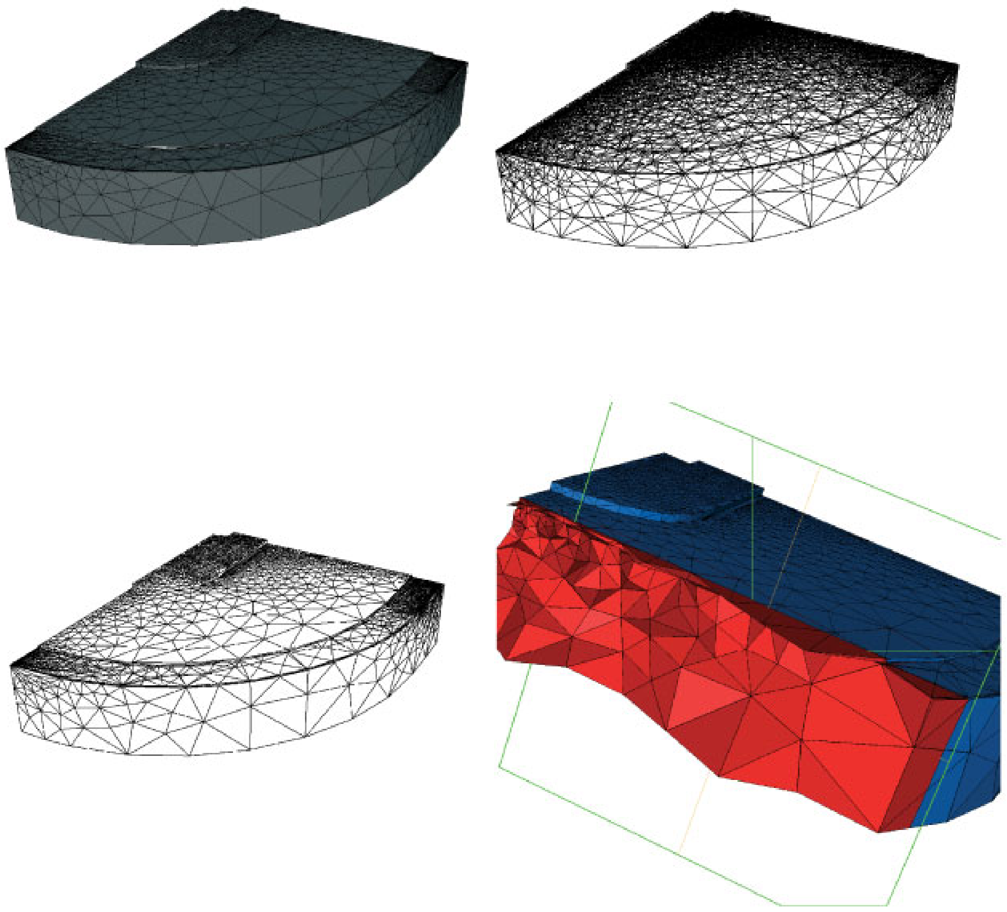

Figure 19. Domain related to a semiconductor device including 4 materials thus with non-manifold prescribed edges (one being common to more than two triangles). Note that some regions are rather thin compared with their neighbouring regions. The same views as in the other figures.

ACKNOWLEDGEMENTS

This work has been partially funded by the E.E.C, under the MAGIG_FEAT project, no. 1999-11433, whom we thank here. Within this project, we faced a number of meshes with delicate pathologies. We also thank the people in charge of the 'drexel' depository from which we picked a number of examples.

REFERENCES

1. Preparata FP, Shamos MI. *Computational Geometry, An Introduction*. Springer: Berlin, 1985.
2. Boissonnat JD, Yvinec M. *Algorithmic Geometry*. Cambridge University Press: Cambridge, 1997.
3. Chazelle B. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry* 1991; **6**:485–524.
4. Cook WA. Body oriented coordinates for generating 3-dimensional meshes. *International Journal for Numerical Methods in Engineering* 1974; **8**:27–43.

5. George A. Computer implementation of the finite element method. *Ph.D. Thesis*, Department of Computer Science, Stanford University, 1971.
6. Yerry MA, Shephard MS. A modified quadtree approach to finite element mesh generation. *IEEE Computer Graphics and Applications* 1983; **3**(1):39–46.
7. Frey PJ, George PL. *Mesh Generation*. Hermes: Paris, 2000.
8. Ruppert J, Seidel R. On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete and Computational Geometry* 1992; **7**:227–253.
9. Chazelle B, Palios L. Triangulating a nonconvex polytope. *Discrete and Computational Geometry* 1990; **5**:505–526.
10. George PL, Hecht F, Saltel E. Automatic mesh generator with specified boundary. *Computer Methods in Applied Mechanics and Engineering* 1991; **92**:269–288.
11. Liu A, Baida M. How far flipping can go towards 3D conforming/constrained triangulation. In *Proceedings of the 9th International Meshing Roundtable*, New Orleans, 2000; 307–315.
12. Weatherill NP, Hassan O. Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints. *International Journal for Numerical Methods in Engineering* 1994; **37**: 2005–2039.
13. Murphy M, Mount DM, Gable CW. A point-placement strategy for conforming Delaunay tetrahedralization. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms*, 2000; 67–74.
14. Hazlewood C. Approximating constrained tetrahedrizations. *Computer Aided Geometric Design* 1993; **10**:67–87.
15. Pebay Ph P. Delaunay-admissibilité a priori en dimensions 2 et 3. *Thèse de l'Université Pierre et Marie Curie*, Paris, 2000.
16. Löhner R, Parikh P. Three-dimensional grid generation by the advancing front method. *International Journal for Numerical Methods in Fluids* 1988; **8**:1135–1149.
17. Shephard MS, Georges MK. Automatic three-dimensional mesh generation by the finite octree technique. *International Journal for Numerical Methods in Engineering* 1991; **32**:709–749.
18. Schönhardt E. Über die Zerlegung von Dreieckspolyedern. *Mathematish Annalen*, 1928; **98**:309–312.
19. Chazelle B. Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm. *SIAM Journal on Computing* 1984; **13**(3):488–507.
20. Borouchaki H, George PL, Lo SH. Boundary enforcement by facet splits in Delaunay based mesh generation. in *7th International Conference on Numerical Grid Generation in Computational Field Simulations*, Whistler, BC, Canada, September 2000, 203–221.
21. George PL. Improvement on Delaunay based 3D automatic mesh generator. *Finite Elements in Analysis and Design* 1997; **25**(3–4):297–317.
22. Coupez Th. Grandes transformations et remaillage automatique, *Thèse ENSMP*, *CEMEF*, Sophia Antipolis, 1991.
23. George PL, Borouchaki H. Maillage simplicial d'un polyèdre arbitraire. Partie 1: Existence et coût. *RR INRIA*, No. 4397, 2002.
24. George PL, Borouchaki H, Saltel E. Maillage simplicial d'un polyèdre arbitraire. Partie 2: Construction et exemples. *RR INRIA*, No. 4398, 2002.