# 3D Triangulations for Industrial Applications

Paulo Roma Cavalcanti
Department of Computer Science
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil
http://orion.lcg.ufrj.br/roma

Yalmar Ponce Atencio
Claudio Esperança
Flavio Pereira Nascimento
COPPE Sistemas
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil
http://www.lcg.ufrj.br

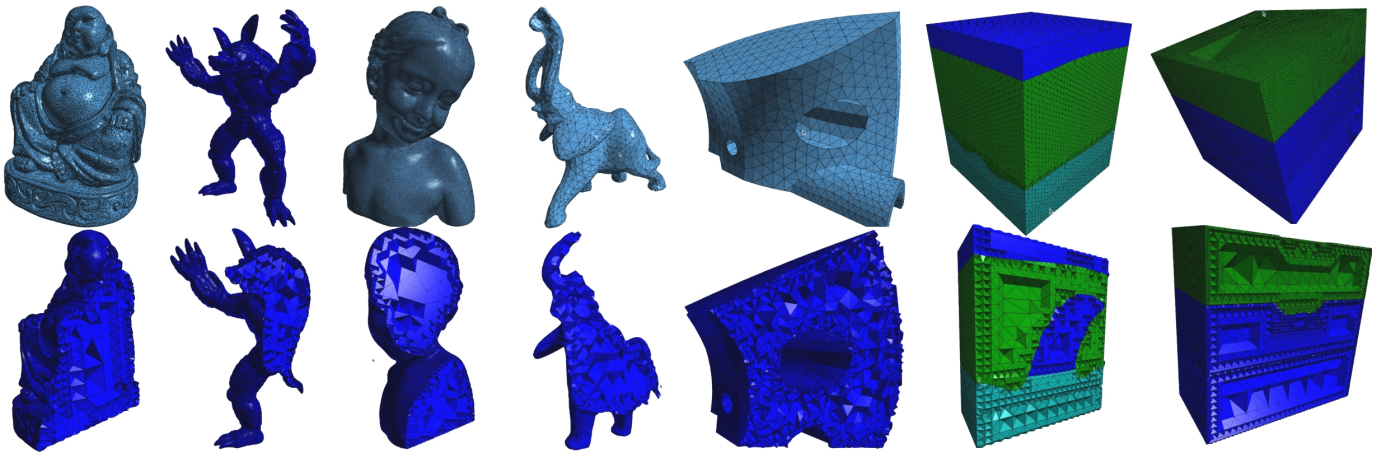

Fig. 1. Mesh Gallery.

***Abstract*—This paper presents a methodology for creating 3D non-structured meshes for industrial applications, which honor all the boundaries of the domain. The input for the system is a multi-region 3D non-manifold model, possibly containing dangling faces, which poses a number of difficulties for standard 3D triangulation frameworks. The main contribution of this work consists of a detailed description of how these difficulties were overcome in a real implementation.**

***Keywords*-3D Triangulations, mesh optimization, multi-region triangulations.**

## I. Introduction

The robust generation of non-structured 3D meshes is fundamental in several areas of the applied sciences, such as Finite Element Analysis, Computer Graphics, Geo-sciences, Engineering, etc. The mesh generation is the process of decomposing a domain into a set of simplices satisfying certain geometrical criteria, in order to allow its usage in several kinds of simulations, for example, the evolution of sedimentary basins and multiphase fluid flow within sediments (Figure 1).

Algorithms for mesh generation have been an active research topic in the past three decades [1], and three main families of algorithms have emerged: Octree based methods [2], [3], Delaunay based methods [4], [5], [6], [7] and Advancing Front [8], [9], [10], [11].

Depending on the purpose of the application, the mesh generation algorithm may or may not honor the boundary of the underlying model, which means that the boundary of the original model can only be approximated in the final mesh. Nonetheless, one of the main sources of difficulties in any Computational Geometry algorithm is the finite precision nature of the calculations performed by digital computers. Therefore, the robustness of the algorithms is a main concern in any implementation used for production.

This paper presents a 3D mesh generator, which can be used for triangulating any 3D non-manifold model given as a set of polygonal surfaces. The implementation uses the CGAL library (http://www.cgal.org) for several of its data structures in order to achieve robustness. Although the CGAL library has procedures to produce a 2D or 3D Delaunay triangulation from a cloud of points, the creation of constrained triangulations is still limited, mainly because CGAL does not have a general way of inputting complex geometric models. As a consequence, CGAL is only able to create meshes of domains given by a manifold surface mesh, or multi-region domains given by the evaluation of an implicit (scalar) function. In any case, the boundary of the domain is only approximated in the resulting mesh (Figure 2).

To cope with such limitations, we developed a methodology for inputting and representing 3D non-manifold models, which partitions the space into a set of 3D regions, by using only the tools available in the CGAL library. Our algorithm first creates

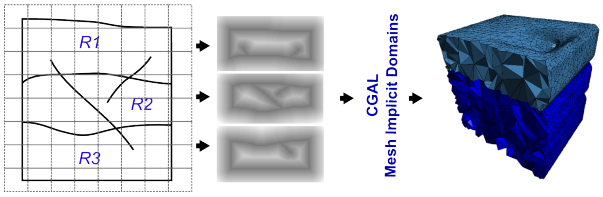


Fig. 2. Implicit Domain generated from a multi-region model and the corresponding triangulation.

a Delaunay triangulation of the vertices of the input model, and then recovers all of the missing edges and faces of the original model. We have also implemented a mesh improvement phase to increase the quality of the final mesh.

## II. Input Model

The CGAL library has functions for inputting models, either bounded by a 2D manifold surface mesh, or defined by an implicit function. Unfortunately, multi-region models defined by several non-manifold surface meshes can not be input directly, but only implicitly, by means of a scalar field. This is due to the fact that surface boundaries are routinely represented by Half-Edge data structures in the CGAL library, which are not amenable to describing non-manifold domains. Nonetheless, there are several important applications where the domain must be represented by a 3D non-manifold model, possibly with dangling faces, which cannot be appropriately described by an implicit function. To deal with such kinds of complex models, we propose a feasible solution by breaking the model into a set of simple surface patches, we call "fragments".

Furthermore, the 3D triangulation of multi-region models requires an scheme for attaching attributes to every surface or region presented in the domain, so one can set properties to be used by a simulation process.

Therefore, our data structure defines the following elements:

- Vertex: represents a vertex of the model, with its coordinates, $x$, $y$, $z$, and possesses a unique identifier, called the vertex index.
- Face: is a list of vertex indices, which describes a face of the model, and has a unique identifier called the face index.
- Fragment: is a list of connected faces identified by a unique fragment index. This way, each fragment corresponds to a surface patch used in the construction of the input model. In the particular case of a Geo-science model, it can be a horizon, a fault, or a face of the model bounding box. Each face between two adjacent regions belong to the same fragment. Also, every face in a fragment must have the same attribute set.
- Region: is composed by a list of fragment indices, corresponding to faces with the same orientation. This way, for each region, a fragment list has faces oriented in the clockwise order, while its adjacent region is traversed in counter-clockwise order. The region is identified by a unique region index.

## III. 3D Constrained Triangulation

Our triangulation algorithm employs the CGAL Delaunay engine to triangulate the convex hull of the vertices of the model. Therefore, there can be a set of edges and faces present in the input model, but absent in the Delaunay triangulation. These missing elements are introduced one by one to produce a constrained triangulation, which contains all of the vertices, edges and faces of the input surfaces.

The algorithm is described in [12], which can be summarized in five steps:

1) Convex hull: the triangulation of the convex hull defined by the vertices of the input model is created.
2) Edge recovery: the missing edges, that is, the edges of the input model that are not in the Delaunay triangulation, created in step 1, are inserted into the triangulation.
3) Face recovery: the missing polygonal faces of the input model are inserted into the triangulation.
4) Classification: all of the simplices of the triangulation are classified relatively to the input model. This way, all tetrahedra will be associated to the region of the space in which it is contained.
5) Mesh improvement: a series of transformations are applied to the mesh to improve the shapes of its faces/tetrahedra.

Some auxiliary data structures are necessary to keep the consistency between the original model and the triangulation. The input model is represented by a set of fragments, which must be kept up-to-date during the triangulation, so that any new vertex, edge or face created is also inserted into the model.

## IV. Auxiliary Points

The regions of the model must be filled with auxiliary (Steiner) points, so the point distribution permits the generation of good shaped tetrahedra. For this purpose, a grid is defined in the model bounding box, and points keeping a minimum distance from all surfaces are inserted into the model. The position of these points can be determined by three types of lattices: cubic, hexagonal [13], and adaptive. In any case, an octree is kept for avoiding the insertion of points too close to a face of the model. The subdivision goal is to achieve a configuration where there are at most three faces per cell. Each point is then tested against the faces of the cells it interferes with [14], before being added to the model, as depicted in Figure 3. The threshold is a function of the distance between two adjacent grid vertices.

## V. Edge Recovery

Edge recovery implies checking which constrained edges are missing from the triangulation of the model's convex hull, and forcing its appearance in the final triangulation.

The process of edge recovery we used is known as *stitching* [15], [16] and, is based on the insertion of vertices on a missing edge. This process works because in a Delaunay triangulation each vertex is connected to its closest neighbor by an edge. However, to avoid an infinite loop in certain configurations,

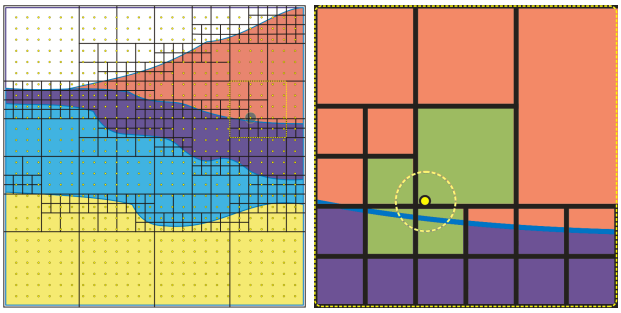

Fig. 3. Auxiliary Octree and the threshold (depicted as a yellow circle), used to find out the closest face to the given point.

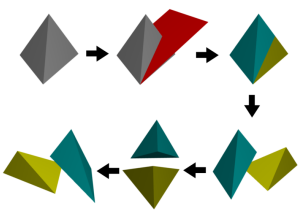

Fig. 4. Tet-Tet.

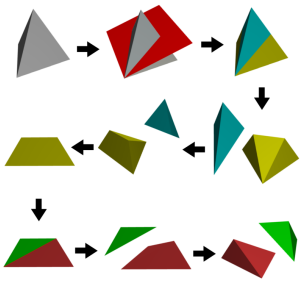

Fig. 5. Tet-Pyramid.

where a new vertex causes the elimination of a previously inserted edge [17], protecting spheres centered on the vertices of the edges are used.

This process is repeatedly executed for each sub-segment of a missing edge, and in the end, the edge is recovered, maybe as a collection of triangulation edges.

## VI. Face Recovery

Face recovery implies checking which constrained faces are missing from the triangulation of the model convex hull, and forcing its appearance into the final triangulation. This process is more complex than edge recovery, mainly because even if all edges on the boundary of a missing face are present in the triangulation, the face itself maybe missing.

Hazlewood [18] has shown it is possible to produce constrained triangulations, by just re-triangulating tetrahedra, which are intersected by a missing polygonal face. First, all intersection points between a missing polygonal face and the edges of the triangulation are found. Then, all intersected tetrahedra are locally re-triangulated. Since all constrained edges have been already recovered, the local triangulation can be performed through the use of the operators described by Weatherill and Hassan [16].

These operators deal with the cases where the face completely intersects a tetrahedron, which is divided into a set of new polyhedra. The new polyhedra can be tetrahedra, pyramids with quadrilateral bases, or prisms with two triangular and three quadrilateral faces.

Since each tetrahedron is processed separately, it is not possible to assure that all intersected tetrahedra will be re-triangulated consistently by these operators. The reason is that a prism may have all of its quadrilateral faces already triangulated by its adjacent tetrahedra. This problem is solved by inserting a new point in the interior of the prism, and connecting this new point to the edges on the prism.

## VII. Re-triangulation Operators

*1) Tetrahedron-Tetrahedron Triangulation:* In this case, only one edge of a tetrahedron intersects the missing face. Therefore, a vertex is inserted in the intersection point between the edge and the face, thus splitting the tetrahedron in two new tetrahedra sharing the face. (Figure 4).

*2) Tetrahedron-Pyramid Triangulation:* In this case, two edges of a tetrahedron intersect the missing face. Therefore, two vertices are inserted on the intersection points between the two edges and the face, thus splitting the tetrahedron in a new tetrahedron and a pyramid sharing the face. The pyramid is also re-triangulated conforming to its neighborhood (Figure 5).

*3) Tetrahedron-Prism Triangulation:* In this case, three edges of a tetrahedron intersect the missing face. Therefore, three vertices are inserted on the intersection points between these three edges and the face, thus splitting the tetrahedron into a new tetrahedron and a prism, sharing the face. In this particular case, the prism can be re-triangulated based on its neighborhood (Figure 6), or not. Nonetheless, a new vertex can always be inserted into the prism in such a way it can be re-triangulated conforming to its neighborhood (Figure 7).

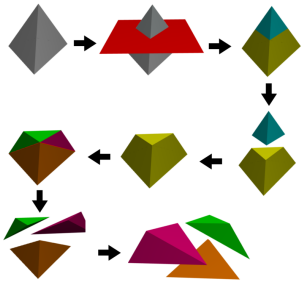

Fig. 6. Tet-Prism.

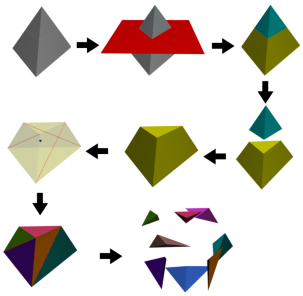

Fig. 7. Point insertion.

*4) Prism-Prism Triangulation:* In this case, four edges of a tetrahedron intersect the missing face. Therefore, four vertices are inserted on the intersection points between the edges and the face, thus splitting the tetrahedron into two prisms, sharing the face. The prisms have at least one degree of freedom, because of the common face (its diagonal can be chosen arbitrarily). This condition may allow the re-triangulation of both prisms, conforming to their neighborhoods, or not. (Figure 8). If a prism cannot be re-triangulated, a new vertex can always be inserted into it, so it can be re-triangulated conforming to its neighborhood (Figure 9).

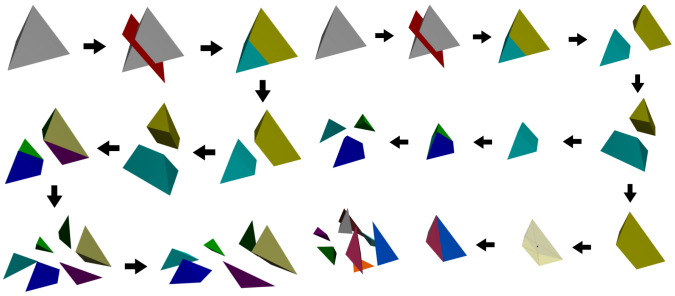

Fig. 8. Prism-Prism.



Fig. 9. Point insertion.

## VIII. Simplex Classification

After all faces have been recovered, there is a constrained triangulation of the model convex hull. At this point, the triangulation has to be carved by removing all tetrahedra into the convex hull, but outside the model. The carving involves traversing all external tetrahedra (outside the triangulation domain) and checking whether its faces are marked. By definition, an external tetrahedron is one that does not possess any face-adjacent tetrahedron in the triangulation. If there is any marked face $F$ in an external tetrahedron $A$, which does not possess an adjacent tetrahedron $B$, then tetrahedron $A$ should be kept in the triangulation. Otherwise, tetrahedron $A$ should be removed and all its adjacent tetrahedra which were not checked yet, should be considered external tetrahedra. This procedure ends when all external tetrahedra have been evaluated and removed.

After the carving, each simplex in the mesh should be classified to have its attributes set [19], [20], according to the properties corresponding to the regions in a multi-region model. For this purpose, it is necessary to keep the data structure representing the input model updated during all the triangulation process, as new vertices, edges and faces are created in the phases of edge and face recovery. As a consequence, the triangulation faces on the surfaces of the model are always known. These faces can be used to set boundary conditions for running simulations.

The simplex classification involves a point in region testing for a tetrahedron, to determine the region it is into, which can be used as a seed to a flood-fill algorithm. By using the mesh adjacency information, all tetrahedra in a region can be determined and marked during a recursive traversal, which stops when no unmarked tetrahedron can be reached without crossing the boundary of the region.

## IX. Mesh Quality

After the mesh generation, heuristic methods are commonly used to improve the quality of the mesh, (in fact, the quality of its tetrahedra). The quality of a tetrahedron is generally expressed by a number, which estimates its effect (positive or negative) on the interpolation error, the discretization error, and on the stiffness matrix conditioning. The mesh quality is strongly influenced by its worst elements, that is, the worst tetrahedra have more influence than the average tetrahedra.

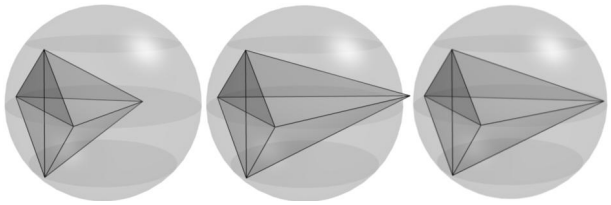

Fig. 10. Slivers. The triangulation on the left does not satisfy the Delaunay (empty sphere) criterion. The other two triangulations pass the empty sphere criterion, and are Delaunay triangulations with slivers.

A sliver is a tetrahedron whose four vertices are very close to a plane, and whose perpendicular projection onto this plane is a convex quadrilateral without a short edge. Slivers are always undesirable in tetrahedral meshes, which were generated for being used in applications of the Finite Element Method. Even when the set of points is well distributed in space, slivers may show up in 3D Delaunay triangulations (Figure 10). Therefore, sliver removal algorithms are always applied during the mesh generation process.

There are two main methods for mesh improvement: smoothing and topological transformations. Smoothing based methods move some vertices to improve the shape of the incident tetrahedra, and they do not change the mesh topology (connectivity). Topological transformations, on the other hand, remove tetrahedra from the mesh and replace them by a new set of tetrahedra that fill the same space. As a consequence, the topology of the mesh can be changed.

Smoothing methods belong to the numerical optimization domain, while topological transformations belong to the combinatorial optimization domain. Both techniques are more effective when applied together.

Smoothing and topological transformations are used by hill-climbing methods to optimize the quality of a mesh. A special function maps each mesh onto a set of values, which describe the mesh. Hill-climbing methods evaluate the application of a certain operation in an specific region of the triangulation. If the quality of the modified mesh is greater than the quality of the original mesh, then the operation is effectively applied to the mesh, and the hill-climbing method looks for another operation, which improves the quality of the new mesh. Operations that do not improve the objective-function value are not applied to the mesh. Therefore, the final mesh cannot be worse than the original mesh. The hill-climbing method ends when no operation is able to produce any further improvement.

Freitag and Ollivier-Gooch [21] presented a hill-climbing method that combines smoothing, based on optimization, with several topological transformations, such as flips 2-3, flips 3-2 and an operation called edge removal (Figure 12). They also describe the performance of several different scheduling strategies on a set of meshes. It should be noted that scheduling refers either to the order and number of operations or the criteria for selecting the tetrahedra that should be improved. They also show that their best scheduling remove most of the badly shaped tetrahedra, and offer some empiric recommendations about what makes a scheduling better than another.

Klingner and Shewchuk [22] propose some new topological operations, beyond the smoothing applied to vertices on the boundary, to empower the repertoire of operations of the hill-climbing method. They also presented a new scheduling that employs these new operations, and show that, by applying all operations together, a better result is produced than applying them separately. Even taking speed into account, they make an analysis of the operations with greater impact on the mesh improvement, either individually or combined with other operations.

## X. Smoothing

The most famous smoothing technique is the Laplacian smoothing [23], which moves a vertex to the centroid of the vertices it is connected to (Figure 11). Typically, the Laplacian smoothing applies several smoothing steps to each vertex in sequence, where each step moves just one vertex at a time. The Laplacian smoothing, although popular and effective on bidimensional meshes, may produce a high number of bad tetrahedra on tridimensional meshes.

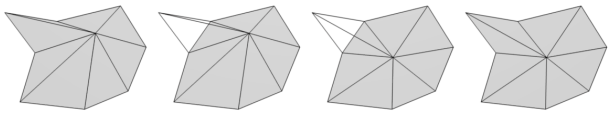

Fig. 11. Bidimensional smoothing example. There can be seen that the smoothed vertex should stay in the gray region to avoid topological (connectivity) issues in the mesh.

Freitag *et al.* [24] proposed a smoothing algorithm to optimize the worst tetrahedron in the set. In practice, this algorithm maximizes the smaller angle among all of the tetrahedra incident on a given vertex. This smoothing is applied only to vertices inside the mesh, that is, vertices on the boundary of the mesh are not smoothed. However, Klingner and Shewchuk [22] have extended this smoothing to vertices on the boundary of the mesh. In this case, it is necessary that the faces on the boundary of the mesh incident on a vertex lie within a small distance of a given plane. This way, these vertices can be smoothed along this plane.

The smoothing technique chosen in this work is known as smart smoothing [21]. Should a smoothing operation compromise the quality of the corresponding tetrahedra, then the operation is not applied. Therefore, the quality of the mesh never gets worse because of the smoothing.

## XI. Topological Transformations

Topological transformations encompass operations that modify the connectivity of the mesh, while preserving its volume. New tetrahedra replace some tetrahedra of the mesh, which occupy the same space. In this phase of the mesh generation process, this is done in such a way that the quality of the mesh is improved by means of tetrahedra replacement. Should the new tetrahedra have a lower quality compared to the tetrahedra to be replaced, then the operation is not performed. The following topological transformations are implemented: flip 2-3, flip 3-2 and flip 4-4 ([6]).

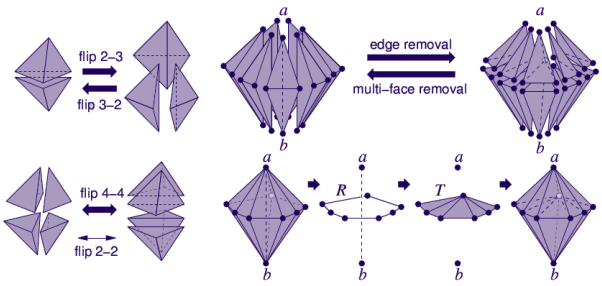


Fig. 12. Some topological transformations employed in the mesh improvement.

## XII. Sliver Removal

Cavendish *et al.* [25] have shown how the presence of slivers is prejudicial to tridimensional triangulations. Sliver removal is not a simple task. Furthermore, even Delaunay triangulations generated from well distributed points can contain slivers, as noted by Talmor [26].

The first positive result in sliver removal was obtained by Chew [27], who adds new points to obtain a uniformly dense mesh. Cheng *et al.* [28] have shown how to specify weights to the points, during the creation of a weighted Delaunay triangulation, in such a way that slivers do not appear. This method does not add new points. Unfortunately, though, it is not completely effective for an arbitrary model. Edelsbrunner *et al.* [29] propose some mesh improvement techniques to be applied in a post-processing phase to remove slivers, mainly using smoothing operations.

In the mesh generator implemented in this work, some operations to remove slivers in several phases along the process have been implemented. However, certain operations can only be performed after recovering the missing faces, since these operations do not honor, globally, the Delaunay criterion. The sliver removal operations include: vertex smoothing, flip 2-3, flip 3-2 and flip 4-4. After the phase of simplex classification, where tetrahedra outside the model are removed, a new sliver removal operation is possible: sliver peel off.

### A. Sliver Peel Off

This operation detects slivers with a face on the boundary of the triangulation, and simply removes them. The four vertices of a sliver are almost co-planar, which means its volume can be considered null. Therefore, the peel off operation will not introduce any change inside the triangulation, because the corresponding tetrahedra are on the boundary.

## XIII. Mesh Improvement

The success of the Finite Element Method depends on the shape of the tetrahedra in the mesh. Large dihedral angles (near 180 degrees) cause severe interpolation errors and reduce the precision of numerical simulations [30], [31]. On the other hand, small dihedral angles cause stiffness matrices, associated to the Finite Element Method, to be ill-conditioned [31]. In some cases, a few bad tetrahedra can ruin a whole simulation.

For a typical model, the constrained Delaunay triangulation generated may have a huge amount of undesired badly shaped tetrahedra. To cope with this issue, a very common approach is just adding points into the model bounding box, positioned at vertices of a regular grid. Nonetheless, this grid may cause the generation of an elevated number of tetrahedra. As a consequence, the appropriate approach is using an adaptive grid [22], thus allowing the generation of regular tetrahedra (good quality) into the model. During the phase of the generation of the Delaunay triangulation, the points of the grid are inserted, according to the Delaunay criterion, in the same way as the points of the model.

The quality of a tetrahedron can be accessed by a single numerical value, called the quality measurement. There are several ways of determining this quality measurement [22], [31], and we chose the least value of the sine, between the six dihedral angles of a tetrahedron, also known as the smallest sine measurement. This measurement penalizes either large or small angles, and Freitag and Ollivier-Gooch [21] consider it the most effective among all tested measurement criteria.

Klingner and Shewchuk [22] employ a hill-climbing method that processes all of the tetrahedra in the mesh, so the worst tetrahedra in the resulting mesh are as good as possible. Although giving excellent results with respect to the quality of tetrahedra, its running time is very high. Therefore, since badly shaped tetrahedra have a higher influence than average shaped tetrahedra, the hill-climbing method used in this work will process only badly shaped tetrahedra. A tetrahedron is considered bad if its dihedral angles, minimum and maximum, are outside a pre-defined range, defined in the method implementation.

Operations, such as smoothing, flip 2-3, flip 3-2 and flip 4-4, are also employed in our hill-climbing method. We also use the point insertion scheme [22].

### A. Point Insertion

This operation computes a position in a bad shaped tetrahedron, and then tries to insert a point at this position (Figure 13). As a consequence, the best possible set of tetrahedra to be removed from the triangulation needs to be calculated. This set of tetrahedra is called a cavity. The triangulation is seen as a visibility graph, relative to the position of the inserted point, and the cavity computation is performed as a search in this graph.

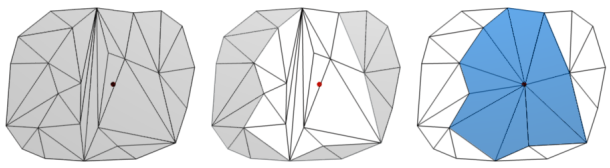

Fig. 13. Point insertion: a bi-dimensional example. On the left, there can been seen the vibility relative to the inserted point $p$. On the right, there can be seen the cavity (in blue) calculated using the visibility graph.

The triangulation is reconstructed by connecting the inserted point to the faces on the boundary of the cavity, followed by a smoothing of the new vertex. If the quality of new tetrahedra is superior to the quality of the tetrahedra previously in the cavity, then the operation is executed. Otherwise, the mesh remains the same.

It is possible that a vertex is removed from the mesh, if all tetrahedra incident to it are part of the cavity. Therefore, sometimes this operation reduces the number of vertices in the mesh.

### B. Scheduling

Scheduling encompasses the order and number of operations to be executed by the hill-climbing method, and also defines on which set of tetrahedra the operations are going to be performed. In general, the scheduling is performed in steps, which apply a set of operations on all of the selected tetrahedra. The scheduling schemes described in the literature are heuristic, and have been developed by means of trial and error. However, they offer valuable clues based on practical tests.

Joe's algorithm [32] visits each face and checks whether any transformation from its repertoire can improve, locally, a set of tetrahedra. The algorithm checks all faces and ends when a step does not cause any improvement in the mesh.

The scheduling of Freitag and Ollivier-Gooch [21] starts with two steps of flips 2-3, followed by a step of edge removal and two steps of smoothing on all tetrahedra of the mesh. Then, it executes a step of flip 2-3 and a face removal operation only on the worst tetrahedra of the mesh. This scheduling ends with two steps of smoothing on all vertices of the mesh.

Klingner and Shewchuk [22] developed a scheduling without a fixed number of steps. They begin with a smoothing step on all vertices and a topological transformation without point insertion. All transformations try edge removal first and then face removal. The scheduling proceeds with a loop to smooth all vertices. If the smoothing is not able to improve the mesh quality, then topological transformations are tried. If no improvement is achieved, then point insertion is executed. Every time one of these operations improves the quality of the mesh, the loop is reset. The scheduling ends when three consecutive steps of the loop are not able to improve the quality of the mesh. Their scheduling offers very good results, for improving the quality of the mesh. However, the run time of the method is high.

In this work, the running time for improving the mesh is an important requirement. Therefore, the scheduling chosen tries to balance the running time of the mesh improvement algorithm and the quality of tetrahedra in the triangulation.

The goal of the improvement process implemented is not to obtain a mesh whose worst tetrahedra are as good as possible. Rather, the goal is to reduce the number of bad tetrahedra.

The scheduling mechanism has the effect of traversing all of the mesh, always improving its tetrahedra, even those acceptable from the point of view of the Finite Element Method. Since bad tetrahedra are really undesirable, the improvement operations, inside the loop, are performed only on them.

Our scheduling algorithm starts with a smoothing step applied on all vertices inside the mesh. Then, it performs a

loop that tries to improve a bad tetrahedron at a time. The order in which the operations are executed, in the loop, is the following: vertex smoothing, flip 3-2, flip 4-4, flip 2-3 and point insertion. This way, the performance of the scheduling is not a critical factor, as long as the quality of the mesh is not prejudicial to the Finite Element Method.

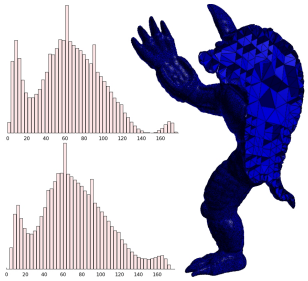

Fig. 14. Armadilo.

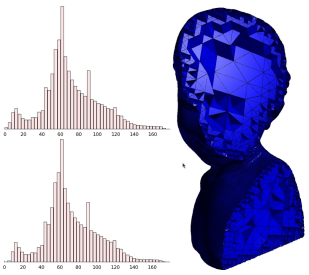

Fig. 15. Bimba.

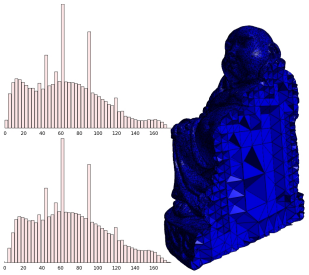

Fig. 16. Budha.

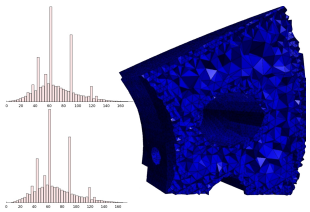

Fig. 17. SPX.

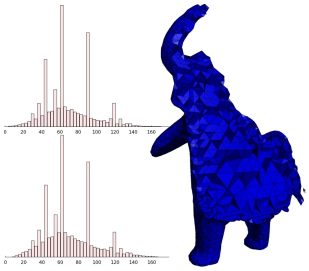

Fig. 18. Elephant.

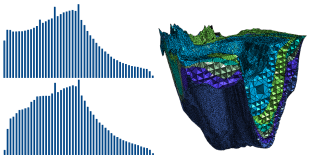

Fig. 19. Gulf of Mexico with six regions and a fault.

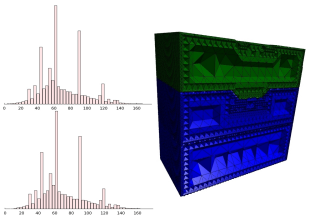

Fig. 20. Synthetic.

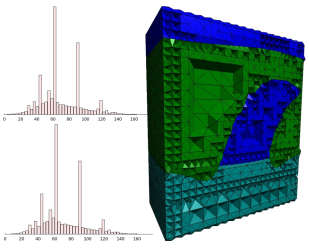

Fig. 21. Salt.

TABLE I
SOME MESH STATISTICS GENERATED WITH OUR METHODOLOGY.

| Model | Vtx | Tet | Bad tet before | Bad tet after | Time (gen.) | Time (imp.) |
|---|---|---|---|---|---|---|
| Budha | 52158 | 160387 | 9121 | 294 | 698sec | 56sec |
| Bimba | 78635 | 205587 | 2606 | 55 | 874sec | 67sec |
| Armadilo | 92555 | 243115 | 16898 | 1442 | 1052sec | 89sec |
| SPX | 36635 | 181996 | 527 | 0 | 715sec | 59sec |
| Elephant | 69834 | 336681 | 632 | 0 | 985sec | 101sec |
| Salt | 147242 | 818903 | 1697 | 14 | 5981sec | 563sec |
| Synthetic | 131801 | 693492 | 425 | 0 | 3745sec | 279sec |

## XIV. CGAL

CGAL (Computational Geometry Algorithm Library) is an open source project aiming at providing an easy access to efficient C++ geometric algorithms. The CGAL library contains data structures, algorithms, and predicates, which can be applied on geometric objects, such as: points, vectors, segments, polygons, etc. These objects and predicates compose the geometric nucleus (kernels) of the library.

Applications requiring an accurate numerical precision should employ a kernel supporting exact arithmetic, for instance, a kernel based on the GMP library (*GNU Multi-Precision Library*). GMP is an open source library for applications demanding arithmetic with an arbitrary precision, and can be applied onto several numerical representations. In our mesh generator, the chosen kernel uses exact predicates and exact geometry for the objects, therefore being robust in all phases of the mesh generation. Exact predicates are essential in the 3D Delaunay mesh generation and the exact geometry is necessary when performing intersection operations.

## XV. Limitations

It should be noted that if the input model has small dihedral angles between some constrained faces, the corresponding (bad) tetrahedra cannot be removed, since the mesh generator honors the boundaries by design. This is a common situation in Geo-science models, for instance Figure 19, where all surfaces meet at an internal fault. Another requirement is that the regions of the input model must be water-tight, so the mesh does not ”leak” from one region to another.

The memory consumption is mainly dictated by CGAL data structures. In particular, the 3D triangulation data structure represents each tetrahedron by four vertex pointers and four neighbor pointers, plus four coordinate values for each vertex. Fragments are represented by a Half-edge data structure, but these tend to add little to the memory footprint of the application.

## XVI. Conclusions and Future Work

The main goal of this work was obtaining a robust implementation of a constrained mesh generator, suited for industrial applications. In the Geo-science area, for instance, the generation and maintenance of numerical meshes can be quite complex for basins that have undergone extensive changes in geometry, as a result of compaction, diapirism, and fault motion [33].

The mesh generator is based upon the 3D Delaunay triangulation paradigm, and is able to apply a set of operations in order to improve the quality of its tetrahedra.

The whole system has been implemented with portability as a requirement, and it runs either on 32 or 64 bit operating systems, such as Linux, Mac OS and Windows, using the same source code. The code is object oriented and was written in C++.

The implementation used the CGAL library and the main contribution of the paper was the creation of a scheme to break the model into fragments, thus allowing the input of complex, multi-region domains. We also combined several scheduling mechanisms available in the literature, in order to produce a mesh in a reasonable time and with a small number bad elements, as can be seen in table I. The table depicts the number of vertices, tetrahedra, bad elements (before and after mesh optimization), and the run time (for generating and improving the mesh), corresponding to the models in Figures 14, ... 21. Those figures also present the histogram of minimum dihedral angles, before and after optimizing the mesh.

The sequence of this work will evolve in the direction of obtaining the best mesh as possible, given that the surfaces defining the domain (e.g., a sedimentary basin) may have some structural small dihedral angles, commonly known as pintch-out. Therefore, better sliver removal and faster scheduling algorithms will have to be implemented. We are also working to be able to generate and process some huge meshes, with hundred of million tetrahedra.

## REFERENCES


[1] S. Owen, "A survey of unstructured mesh generation technology," in *Proceedings of the Seventh International Meshing Roundtable*. Dearborn, Michigan: Sandia National Laboratories, October 1998.

[2] W. J. Schroeder and M. S. Shephard, "A combined octree/Delaunay method for fully automatic 3-d mesh generation," *International Journal for Numerical Methods in Engineering*, vol. 29, pp. 37–55, 1990.

[3] M. S. Shephard and M. K. Georges, "Automatic three-dimensional mesh generation by the finite octree technique," *International Journal for Numerical Methods in Engineering*, vol. 32, pp. 709–749, 1991.

[4] D. F. Watson, "Computing the n-dimensional Delaunay tesselation with application to Voronoi polytopes," *The Computer Journal*, vol. 24, no. 2, pp. 167–172, 1981.

[5] B. Joe, "Three-dimensional triangulations from local transformations," *SIAM J. Sci. Stat. Comput.*, vol. 10, no. 4, pp. 718–741, July 1989.

[6] B. Joe, "Construction of three-dimensional Delaunay triangulations using local transformation," *Computer Aided Geometric Design*, vol. 8, pp. 123–142, 1991.

[7] J. R. Shewchuk, "Delaunay refinement mesh generation," Ph.D. dissertation, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1997.

[8] R. Lohner, "Generation of three-dimensional unstructured grids by the advancing front method," in *Proceedings of the 26th AIAA Aerospace Sciences Meeting*, Reno, Nevada, 1988.

[9] T. Barth and D. Jesperson, "The design and application of upwind schemes on unstructured meshes," in *Proceedings of the 27th AIAA Aerospace Sciences Meeting*, Reno, Nevada, 1989.

[10] D. J. Mavriplis, "An advancing front Delaunay triangulation algorithm designed for robustness," ICASE, Technical Report 92-49, October 1992.

[11] R. Lohner, "Progress in grid generation via the advancing front technique," *Engineering with Computers*, vol. 12, pp. 186–210, 1996.

[12] P. R. Cavalcanti and U. T. Mello, "Three-dimensional constrained Delaunay triangulation: a minimalist approach," in *Proceedings of the 8th International Meshing Roundtable*. Lake Tahoe, CA: Sandia National Laboratories, October 1999, pp. 119–129.

[13] U. T. Mello and P. R. Cavalcanti, "A point creation strategy for mesh generation using crystal lattices as templates," in *Proceedings of the 9th International Meshing Roundtable*. New Orleans, LA: Sandia National Laboratories, October 2000, pp. 253–261.

[14] L. Stocco and G. Schrack, "Integer dilation and contraction for quadtrees and octrees," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers, and Signal Processing*, 1995, pp. 426–428.

[15] N. P. Weatherill, "Delaunay triangulation in computational fluid dynamics," *Computers and Mathematics with Applications*, vol. 24, no. 5/6, pp. 129–150, September 1992.

[16] N. P. Weatherill and O. Hassan, "Efficient three-dimensional Delaunay triangulation with automatic point creation and imposed boundary constraints," *International Journal for Numerical Methods in Engineering*, vol. 37, pp. 2005–2039, 1994.

[17] J. Ruppert, "Results on triangulation and high quality mesh generation," Ph.D. dissertation, Department of Computer Science, University of California at Berkeley, Berkeley, CA, 1992.

[18] C. Hazlewood, "Approximating constrained tetrahedralizations," *Computer Aided Geometric Design*, vol. 10, pp. 67–87, 1993.

[19] P. R. Cavalcanti, P. C. Carvalho, and L. F. Martha, "Nonmanifold modeling: An approach based on spatial subdivisions," *Computer-Aided Design*, vol. 29, no. 3, pp. 209–220, March 1997.

[20] U. T. Mello and P. R. Cavalcanti, "A topologically-based framework for simulating complex geological processes," in *Proceedings of the AAPG Hedberg Conference-Basin Modeling*. Colorado Springs, CO: American Association of Petroleum Geologists, May 1999.

[21] L. Freitag and C. Olliver-Gooch, "Tetrahedral mesh improvement using face swapping and smoothing," *International Journal for Numerical Methods in Engineering*, vol. 40, pp. 3979–4002, 1997.

[22] B. M. Klingner and J. R. Shewchuk, "Agressive tetrahedral mesh improvement," in *Proceedings of the 16th International Meshing Roundtable*, Oct. 2007, pp. 3–23.

[23] L. R. Hermann, "Laplacian-isoparametric grid generation scheme," *J. of the Eng. Mechanics Div. of the American Soc. of Civil Engineers*, vol. 102, pp. 749–756, 1976.

[24] L. Freitag, M. Jones, and P. Plassmann, "A parallel algorithm for mesh smoothing," *SIAM J. Sci. Comput.*, vol. 20, no. 6, pp. 2023–2040, 1999.

[25] J. C. Cavendish, D. A. Field, and W. H. Frey, "An approach to automatic three-dimensional finite element mesh generation," *Int. J. Numer. Meth. Eng.*, vol. 21, pp. 329–347, 1985.

[26] D. Talmor, "Well-spaced points for numerical methods," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, August 1997, cMU CS Tech Report CMU-CS-97-164.

[27] L. P. Chew, "Guaranteed-quality delaunay meshing in 3d (short version)," in *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*. New York, NY, USA: ACM, 1997, pp. 391–393.

[28] S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S.-H. Teng, "Sliver exudation," in *SCG '99: Proceedings of the fifteenth annual symposium on Computational geometry*. New York, NY, USA: ACM, 1999, pp. 1–13.

[29] H. Edelsbrunner, X.-Y. Li, G. Miller, A. Stathopoulos, D. Talmor, S.-H. Teng, A. Üngör, and N. Walkington, "Smoothing and cleaning up slivers," in *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*. New York, NY, USA: ACM, 2000, pp. 273–277.

[30] M. Křížek, "On the maximum angle condition for linear tetrahedral elements," *SIAM J. Numer. Anal.*, vol. 29, no. 2, pp. 513–520, 1992.

[31] J. R. Shewchuk, "What is a good linear element? interpolation, conditioning, and quality measures," in *In 11th International Meshing Roundtable*, 2002, pp. 115–126.

[32] B. Joe, "Construction of three-dimensional improved-quality triangulations using local transformations," *SIAM Journal on Scientific Computing*, vol. 16, no. 6, pp. 1292–1307, 1995.

[33] U. T. Mello and P. R. Cavalcanti, "A topologically-based framework for three-dimensional basin modeling," in *Multidimensional Basin Modeling*, S. Düppenbecker and R. Marzi, Eds. Tulsa: AAPG/Datapages Discovery Series No. 7, 2003, pp. 255–269.