Nibble Meshing: An Algorithm for Triangulation of Non-Manifold Solid Boundary

David Marcheix Stefka Gueorguieva

Laboratoire Bordelais de Recherche en Informatique
Université Bordeaux I- LaBRI
351, Cours de la Libération
33405 Talence FRANCE

Abstract

A computational method, called *Nibble* algorithm, for triangulation of non-manifold solid boundary is proposed. The algorithm is based on an incremental boundary traversal technique. The mesh generator creates a mesh element-by-element until the whole region is covered no matter the domain complexity (faces with non convex shapes and multi-connected boundaries are treated). At each step of the algorithm, a surface boundary called *active boundary* is evaluated in such a way that it nibbles the surface to be triangulated. The fundamental feature of this process is the definition of an area, called *influence zone*, which controls the node insertion and thus avoids edge intersection tests. Further, the generated mesh is refined through an *extension* of the *Laplacian smoothing*. It allows an optimization of the smoothing quality without saturating the time complexity. A new technique for *adaptive smoothing* is also applied in order to speed up the mesh refinement.

Keywords :
Geometric modelling, non-manifold topology, boundary representation, computational geometry, triangular mesh generation, boundary triangulation.

1 Introduction

A key topic in computational geometry is the generation and the refinement of triangular meshes. Indeed, space partition into triangles and tetrahedra is the simplest subdivision that enables the discretization of any polyhedral domain. Computational analysis using various numerical methods such as the finite element method (FEM) and the boundary element method (BEM) is a classic example of application field.

Our interest is directed towards a more recent application concerning the triangulation of the boundary surface of non-manifold objects (see [9, 11, 17, 19, 18, 21]) defined in terms of the boundary representation (*BRepr*) [12].

On one hand, *BRepr* models are topologically rich representations. The description of the topology of the modeled objects leads to notable improvement of algorithm complexity and

reliability [11]. On the other, the use of solid models as input data for the triangular mesh generation considerably reduces the gap between geometric modeling and product design. As a matter of fact, solid models could be automatically transformed into mesh of finite elements represented in a way compatible to the postprocessing product analysis.

The presented work has been motivated by our researches on integration of free form surfaces into *BRepr* models. More precisely, once the solid boundary is triangulated, it is embedded on a triangular mesh of Gregory-Bezier patches with surface continuity corresponding to the application. For this purpose, triangular mesh generator should meet several requirements:

- It should operate on complex domain. Indeed, non-manifold solids have complex geometric shapes with possible cavities, holes through, protuberances and protrusions, and thus necessitating the treatment of non convex faces with multi-connected boundaries.

- It should produce a mesh as regular as possible i.e. the triangulation should define a subdivision into triangles nearly equilateral. This feature is of primordial importance for the embedment into triangulated free form surface. The more regular is the triangular mesh, the more intuitive is the manipulation of the corresponding free form surface.

- The generator should provide also a local control on the triangle size. Often, free form deformations are applied on chosen parts of the solid boundary. Continuous shape deformation could be obtained if the triangular mesh is graded[1] with node spacing controlled by an explicit continuous node spacing function.

The *Nibble* algorithm is designed with respect to the above requirements. Moreover, it is well suited for adaptive mesh refinement due to the geometric complexity[2] of most solids of practical interest.

The paper is organized as follows. First, basic notions in triangular mesh generation and refinement are reviewed. Next, we present the *Nibble* method and some associated geometric and topologic considerations. Finally, different examples are given to illustrate the performance of the proposed techniques.

2 Basic notions

There is a wide variety of methods for construction of irregular computational meshes ([1, 2, 3, 7, 10, 13, 16, 20, 22, 23]). Irregular meshes are attractive because they allow nodes on curved boundaries of irregularly shaped domain. Moreover, nodes could be distributed in the interior of the domain as desired for variable resolution. Most of the existing automatic mesh generators are focussed on triangulation. Two basic tasks have to be distinguished during the triangulation process: *node placement* and *triangulation of a set of nodes*. Usually, *node placement* precedes the triangulation and is involved in generating or modifying a node list where nodes are identified with the corresponding coordinate information. The *triangulation* consists of constructing a triangle list identifying each triangle and its vertices. Once a triangulation has been created, it is refined in order to improve its geometric and topological mesh irregularity (resp. ϵ_g and ϵ_t) defined as follows:

$\epsilon_g = \sum |0.5 - \frac{r_i}{R_i}|/m$, where $i = 0 \ldots m$, m is the number of elements, and r_i and R_i are the radii of inscribed and circumscribing circles.

$\epsilon_t = \sum |d_i - 6|/n$, where $i = 0 \ldots n$, n represents the total number of interior nodes in the domain, d_i is the degree or the number of neighboring nodes connected to the ith interior node.

[1] Meshes with varying element sizes.

[2] Physically based considerations are beyond the scope of the present discussion.

An important feature of mesh generator is the *size* of mesh elements. A specified element size could be achieved by controlling either the *node spacing* (the distance between neighboring nodes) or the *node density* (the number of nodes per unit length or area).

The *boundary traversal* method is a broadly used method for automatic mesh generation [2, 6, 3, 4] that uses boundary node spacing to guide the node placement.

It is summarized as follows. Initially the domain boundary is defined with a desired node spacing. Then, each boundary edge is considered in turn. A new interior node is placed alongside the current edge so as to form nearly equilateral triangles with the given edge and any other edge close enough to be affected. Next, the boundary of the domain is readjusted removing the triangles just formed. The readjusted boundary is usually called *active boundary*. This process is iterated until the active boundary shrinks down as the last triangle is formed.

In the present article attention is paid to the application of the *boundary traversal* method for meshing non-manifold solid boundary. With this respect an important inconvenience of this method is the loose of the size element control. In fact, the algorithm tends to reduce the size of the triangles when we move away from the domain boundary. Moreover, often non-convex domains (see for example [3]) are initially subdivided into a set of convex regions with simply connected boundaries. Further, each region is triangulated separately and the mesh covering the entire domain is an assembly of the meshes spread over different convex regions. Despite of the fact that the subdivision into convex parts is a time consuming process, the assembly provokes irregularities along the common boundaries that cannot be smoothed (see [5]).

The *Nibble* algorithm presented in the next section follows the boundary traversal approach but it offers several advantages:

- It gives the opportunity to manipulate the size of the triangle in every point of the domain without being only influenced by the boundary. Such a process permits to eliminate the triangle shrinking phenomenon.
- It supports the triangulation of any polygonal domain with no restriction on the geometric shape and boundary connection.
- It offers the possibility to develop the active boundary without having to calculate visibility [4] or edge intersection. An *influence zone* is introduced (see section 3) that avoids these cumbersome calculations.

3 Nibble Algorithm

3.1 Method overview

The proposed *Nibble* algorithm could be classified as a boundary traversal method. To start up, the boundary of the domain to be triangulated is specified along with a function $f(x)$, called *distance function*, defining the size of the mesh elements in each point of the domain. The algorithm could be schematized as follows:

Step 0 : An active boundary is associated to each connected component of the domain boundary.

Step 1 : Each active boundary is subdivided according to the distance function.

Step 2 : Among all edges of all active boundaries, the longest edge (n_i, n_j) is selected.

Step 3 : An isosceles triangle with base the longest edge (n_i, n_j) and height $d=f(\frac{n_i+n_j}{2})$ is constructed. Corresponding to this triangle a zone, called *influence zone* is built up. Its definition will be given in the next subsection. The influence zone enables to validate or to reject the newly created triangle as an element of the generated mesh. If the influence zone does not contain any other node, the triangle is accepted. Otherwise, it is rejected and a new triangle is

constructed linking n_i, n_j and the node lying in the influence zone. This process could lead to the destruction or the subdivision of an active boundary.

Step 4 : A smoothing, called *extended Laplacian smoothing*, is applied using an *adaptive smoothing technique.*

Step 5 : Go back to step 2 until no active boundary is left.

A basic novelty of the method is the use of the influence zone detailed in the next subsection.

3.2 Geometric considerations

As summarized above, the elaborated algorithm for triangular mesh generation is compounded by three main steps : looking for the longest edge (step 2), creating of a new triangle along this edge (step 3) and smoothing a mesh during its evaluation (step 4).

3.2.1 Longuest edge detection

The use of the longuest edge interferes with the self-intersection problems detailed in 3.2.2. From computational point of view, looking for the longest edge is fast insofar as just the edges that belong to the active boundary are considered. On the initial step this information is directly available as long as the underlying geometric model, the *Radialmodel* [8, 14], contains an explicit representation of the boundary edge loops for each solid face. Further this information is maintained for each active boundary using the corresponding modeling tools (see [15]).

3.2.2 Node insertion

The creation of a new triangle is a more delicate task because an erroneous choice could provoke undesired edge intersections and thus could invalidated the generated mesh.

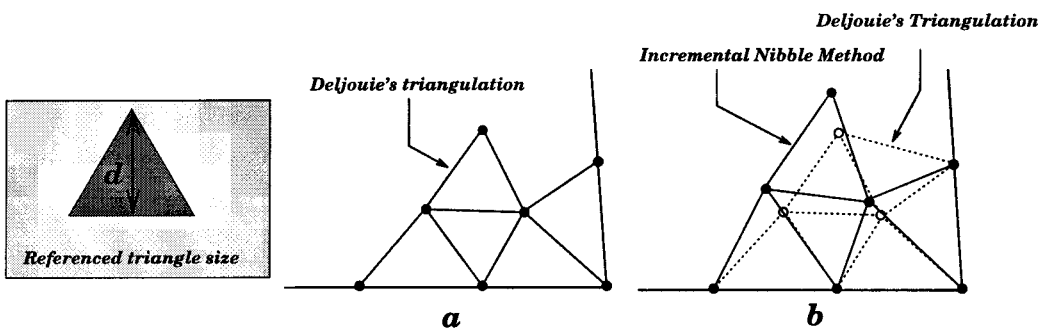

Fig 1 : *(a) An example of triangle shrinking phenomenon with the Deljouie's algorithm [3]. (b) A comparison with the Nibble method.*

This step consists of creating the more adapted triangle. We propose to construct a triangle T with base the longest edge chosen in step 2, called *triangle base*, and height $d = f(\frac{n_i+n_j}{2})$. In the rest of this article, the distance d will be called *triangle size*. This choice enables to avoid the triangle shrinking phenomenon (see fig.1).

In some cases new node insertion can invalidate the generated mesh as shown in fig.2. The insertion of n_x provokes the intersection of two mesh elements. Similarly, the creation of a new node n_y is inappropriate because of the proximity of n_i that will cause the appearance of a degenerated triangle (see [20]).

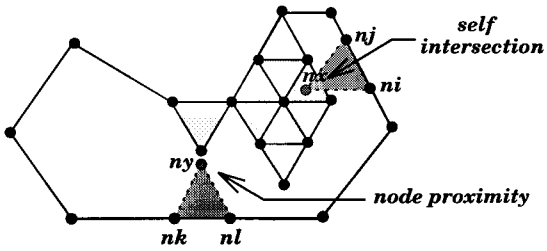

Fig 2 : *Problems arising during the evaluation of the active boundary.*

In order to avoid erroneous node insertion we introduce an *influence zone*. This zone is deduced from the following considerations:

(1) The Delaunay triangulation guarantees triangles as well shaped as possible for given set of nodes. This triangulation verifies the so called empty circumdisc property i.e. for every triangle in the mesh, the open circumdisc of the triangle contains no nodes. With this respect if there is a node n_k that belongs to the circumdisc of the triangle T it is preferable to join n_k to the base of T than to insert n_x. We apply this property for boundary traversal of non-convex domain. Then, just one part of the circumdisc of T is of interest, the one that lies in the interior of the domain. In the opposite case, the enclosed triangle (n_i, n_j, n_k) would not belong to the domain as illustrated in fig.4. Let us denote this zone Z_s.

(2) Let us consider two same length edges (n_i, n_j), (n_j, n_k) and the angle θ between them. We suppose that $0° < \theta < 180°$ (see fig.3). Three cases are possible: $(0° < \theta < 90°, \theta = 90°, 90° < \theta < 180°)$. The angle α_1 denotes the smaller angle when a new node is inserted by subdividing θ thus creating two new triangles (n_i, n_j, n_x) and (n_j, n_k, n_x). The angle α_2 is the smaller angle in the alternative case when a new triangle is closed with the creation of the edge (n_k, n_i). It could be shown (see [14]) that:

- if $0° < \theta < 90°$ then $\alpha_1 < \alpha_2$
- if $\theta = 90°$ then $\alpha_1 = \alpha_2$
- if $90° < \theta < 180°$ then $\alpha_1 > \alpha_2$

As long as we look for a maximization of the smallest angles of the triangles, the new node insertion should be avoided in the first case. Thus, if there exists a node n_k in the square with side (n_i, n_j), it is preferable to reach n_k rather than to insert a new node n_x.

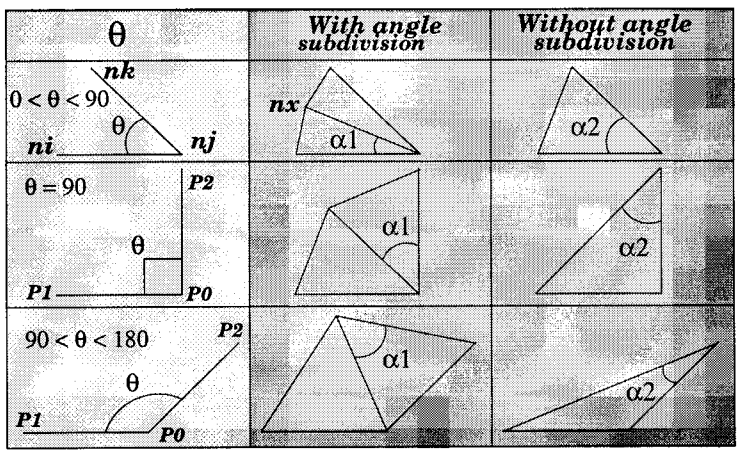

Fig 3 : *Test of the minimum angle : 3 typical situations according to both θ and the operation carried out.*

This observation is extended when (n_i, n_j) and (n_j, n_k) are not equal. Let us denote the resulting rectangular zone Z_r.

(3) During the mesh creation a smoothing is carried out. Then, each newly created node n_x is centered in relation to its neighbor nodes. Let (n_i, n_j) is the base edge found in step (2) and let n_x is the node that we want to insert and create a new triangle (n_i, n_j, n_x) of size d. One should calculate how long should be the distance between n_x and the nearest boundary node, in order to obtain around n_x triangle size close to d. It is proved (see [14]) that if there is a boundary node n_k that belongs to the disc Z_c, centered at n_x with radius $\frac{d}{3}$, then n_k should be joined and no new node should be created.

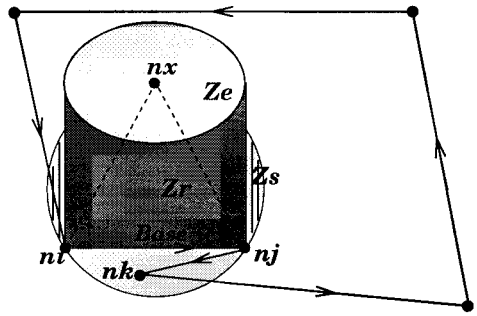

Fig 4: *The influence zone*

Combining the remarks discussed above the *influence zone* is defined as the union of Z_s, Z_r and Z_e, where Z_e is an ellipse with minor radius $\frac{d}{3}$ and major radius equal to the triangle base (n_i, n_j) such that it makes the transition from Z_c to Z_r as shown in fig.4. Then, when we reach step (3) of the algorithm, we test if there is an active boundary node n_k located in the influence zone. If n_k exists then it is connected to the ends of the longest edge and thus forming the new triangle. In the opposite case, a new node n_x is inserted.

One should remark that in step (3), several active boundary nodes could be located in the influence zone. In this case the linear time algorithm of [4] is applied on nodes belonging to the influence zone in order to choose the node to be joined up.

The use of the influence zone permits on the one hand to accelerate the computations. Indeed, following the classical approach visibility nodes are calculated for each potential new node. By contrast, in the proposed method these calculations are made for a candidate node just in case when there are several active boundary nodes in its corresponding influence zone. During algorithm tests this situation occurred very rarely when there are important variations of the distance function. On the other hand, if no boundary node is detected in the influence zone, no self-intersection could occur. In fact, if an edge with endpoints lying outside the influence zone intersects T it should be longer than the triangle base. Such an edge cannot exist because the base is the longest edge according to step. 2 (see [14]).

3.2.3 Mesh smoothing

A broadly used technique for triangular mesh smoothing is the so called Laplacian smoothing [5]. We propose an extension of this method, called *extended Laplacian smoothing*. Our objective is to be able to smooth the generated mesh according to the given distance function at any iteration of the *Nibble* algorithm and in any node including the active boundary. Moreover, by contrast with the classic method, when the mesh is smoothed in chosen node n_0, the nodes $(n_1, \ldots, n_m)$ adjacent to n_0 do not necessarily form a closed boundary around n_0.

The principle can be summarized in the following way. Let n_0 is a node and $(n_1, \ldots, n_m)$ is a list of nodes adjacent to n_0 (i.e. connected to n_0 through an edge). We propose to calculate the new position n_0' of n_0 after the smoothing as the average of all the tops n_{xi} of T_i, $i = 1, \ldots, m-1$, where T_i are the triangles of size $d_i = \frac{n_{i+1}+n_i}{2}$ and base (n_{i+1}, n_i).

The smoothing is applied when a new triangle is constructed. As long as this step is the most expensive in computational cost it is worthwhile to improve it. The elaborated *adaptive smoothing* technique is based on two observations. Firstly, smoothing provokes important mesh modifications locally around a newly inserted triangle. Secondly, depending on the technique used for the construction of the new triangle, the induced modifications are propagated on a variable sized area. It is proved (see [14]) that when active boundaries meet and when the triangle creation does not imply node insertion, the smoothing could induce important mesh modifications. So, the area of smoothing is defined according to the kind of operation that is performed.

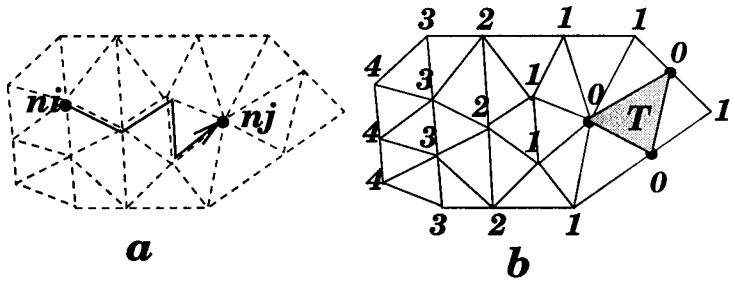

Fig 5 : *(a) A path (n_i, n_j) of length $m = 4$. (b) Node distance to triangle T.*

For this purpose we define a *path of length* m as a sequence of m edges such that each vertex is adjacent to at most two edges as illustrated in fig.5a. If T is the triangle (n_0, n_1, n_2) then all the nodes which can be reached from n_0, n_1 or n_2 by a path of length m and which can not be reached by a path of length smaller than m are said to be at an *influence distance* m to T. An example of triangulation, where the nodes are numbered according to their distance to T, is shown in fig.5b.

The adaptive smoothing technique begins with a chosen triangle T. Starting with T the refinement is propagated to all nodes with influence distance less or equal to a given value depending on the nature of the construction of T. For a comparative illustration see fig.6.

3.3 Topological considerations

The data structure that underlies the *Nibble* algorithm is a non-manifold boundary representation, called Radial Model (see [8, 15]). Each object is described as a complex where each cell is defined in terms of its boundary. Each dimensionally homogeneous part of the object that does not contain singular points[3] is encoded as a *primitive*. Thus an object is represented as an assembly of wireframes , shells or volumes resp. iD primitives, where $i = 1, 2, 3$ specifies the primitive dimension.

For the current application, we are interested in shell and volume primitives. The boundary surface of each volume is a shell and thus is subdivided into $2D$ cells, called faces. The triangulation algorithm operates on faces using their boundary definition as oriented loops of edges ($1D$ cells), each edge being defined as a couple of vertices ($0D$ cells). At the initial step of the algorithm, the boundary of the domain to be triangulated is given by the boundary edge loops of the corresponding faces. Consequently, both the subdivision of the active boundary and the face subdivision are performed using a set of basic operators. The use of this set of basic operators guarantees the validity of the underlying geometric model. See [15] for a detailed discussion. The topological information needed for the triangulation is the boundary edge connection for

[3] Points of non-manifold conditions, internal boundaries or cracks.

each individual face and the face adjacencies for shells and boundary surfaces of volumes when they share common boundaries. Indeed, boundary subdivision should be propagated to all the primitives with common boundaries. This process is illustrated in fig.9.

4 Implementation results

The proposed *Nibble* algorithm is used for automatic triangular mesh generation of non-manifold object boundaries. It is incorporated in *NEMO* [4], a solid modeler running on SG workstations.

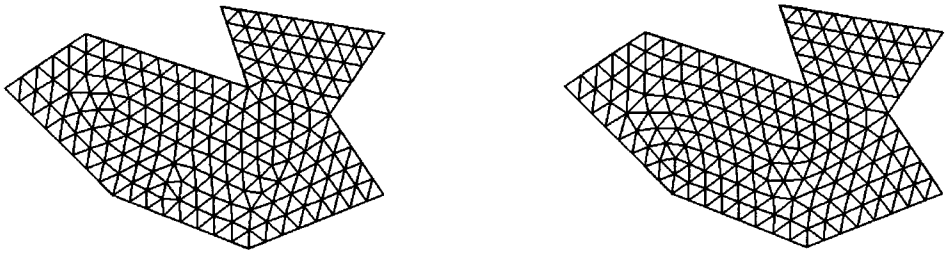

Fig. 6 *Mesh refinement with extended Laplacian smoothing and adaptive smoothing*

The first example in fig.6 illustrates the triangulation with a constant element size of a single face shell. In fig.6a the extended Laplacian smoothing is applied 3 times for each iteration of the *Nibble* algorithm. The resulting triangular mesh is characterized by $\epsilon_t = 0.074$ and $\epsilon_g = 0.0098$. The triangulation in fig.6b uses the adaptive smoothing with the same number of applications and an influence distance=4. The results are ϵ_t=0.058 and ϵ_g=0.0092. The generation of the triangulation in the first case is three times slower than in the second. As one can see, the gain in time complexity using adaptive smoothing is significative for comparative mesh qualities.

Results for the test example given by [20] are shown in fig.7. We start up a "bad" triangulation of the donkey ($\epsilon_t = 0.405$ and $\epsilon_g = 0.046$). The mesh is improved to $\epsilon_t = 0.261$ and $\epsilon_g = 0.018$. using the *Nibble* method and an adaptive smoothing applied 2 times for each iteration with influence distance=3.

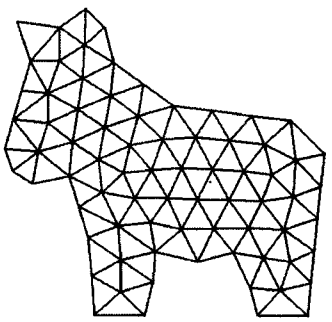

Fig. 7 *Nibble meshing convergence*

The example in fig.8 shows mesh generation with variable triangle size using the linear distance function $f(x)$.

[4] Non-manifold Environment for Modeling of 3D Objects.

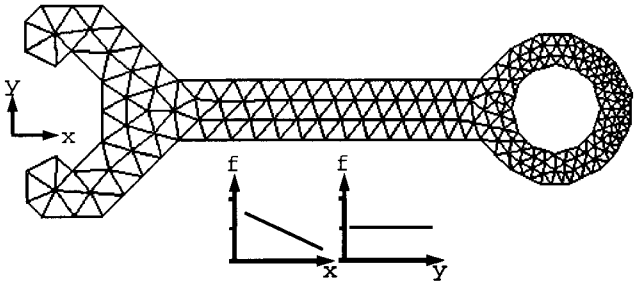

Fig. 8 *Triangulation with linear distance function*

Finally, the embedment of the boundary surface of an object A onto triangular Gregory-Bézier patches is illustrated in fig.9. The underlying graded triangular meshes are generated using the *Nibble* algorithm. Local deformations are applied on the edge e with desired continuity. The deformation continuity is preserved across e for all the primitives sharing e.

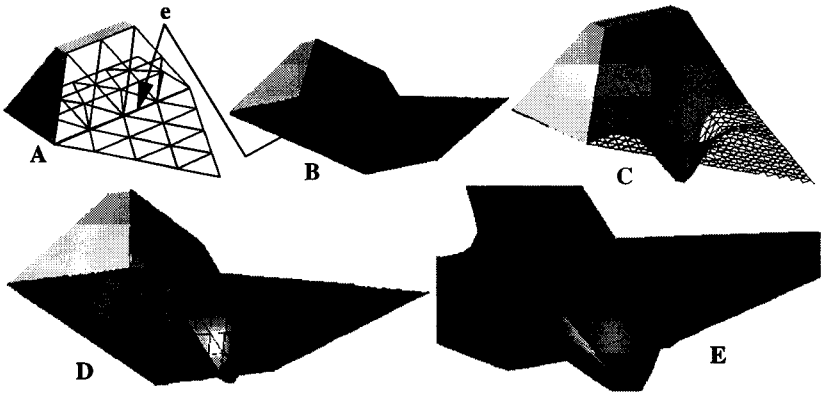

Fig. 9 *Free form deformations using the surface boundary triangulation*

5 Conclusion

A new method for automatic triangular mesh generation, called *Nibble* method, has been introduced. It enables the regular triangulation of non convex domains with multi-connected boundaries without preliminary subdivision into convex regions. The developed algorithm uses a central mechanism called influence zone that offers two basic advantages.

First, the mesh element size is controlled thus avoiding the shrinking phenomenon of the classical boundary traversal methods. Second, no more tests for self-intersections are needed to ensure the mesh validity. In order to refine the generated mesh, an extension of the Laplacian smoothing is proposed. Moreover, an adaptive smoothing technique is elaborated to speed up the computations. The experiments show that quality mesh is improved for a sensible reduction of the time complexity.

The *Nibble* method is incorporated in a solid modeling system as a tool for boundary surface triangulation. The generated triangular meshes are used as control points for the construction of triangular Gregory-Bezier patches. Further research will exploit the distance function variation to produce continuous deformation on solid boundary.

References

[1] A. Bykat. Design of a recursive shape controlling mesh generator. *International Journal for Numerical Methods in Engineering*, 19:1375–1390, 1983.

[2] J.C. Cavendish. An approach to automatic three dimensional finite element mesh generation. *International Journal for Numerical Methods in Engineering*, 21:329–347, 1985.

[3] K. Deljouie-Rakhshandeh. An approach to the generation of triangular grids possessing few obtuse triangles. *International Journal for Numerical Methods in Engineering*, 29:1299–1321, 1990.

[4] K. Deljouie-Rakhshandeh. Point visibility of general polygonal regions. *Comp. J*, 1990.

[5] D.A. Field. Laplacian smoothing and Delaunay triangulation. *Commun. appl. numer. methods*, 4:709–712, 1988.

[6] H.W. Frey. Selective refinement: a new strategy for automatic node placement in graded triangular meshes. *International Journal for Numerical Methods in Engineering*, 24:2183–2200, 1987.

[7] H.W. Frey and D.A. Field. Mesh relaxation: a new technique for improving triangulations. *International Journal for Numerical Methods in Engineering*, 31:1121–1133, 1991.

[8] S. Gueorguieva and D. Marcheix. Non-manifold boundary representation for solid modeling. In *Proc. of the International Computer Symposium*, 1994.

[9] E.L. Gursoz, Y. Choi, and B. Prinz. Vertex-based representation of non-manifold boundaries. In *Geometric modeling for product engineering.*, 1990.

[10] K. Ho-Le. Finite element mesh generation methods: A review and classification. *Computer-Aided-Design*, 20(1):27–38, 1988.

[11] Ch.M Hoffmann. *Geometric and Solid Modeling: An Introduction.* Morgan Kaufmann, 1989.

[12] P. Lienhardt. Topological models for boundary representation: A comparison with n- dimensional generalized maps. *Computer-Aided-Design*, pages 59–82, January/February 1991.

[13] S.H. Lo. A new mesh generation scheme for arbitrary planar domains. *International Journal for Numerical Methods in Engineering*, 21:1403–1426, 1985.

[14] D Marcheix. Geometric modeling of non-manifold objects: Representation, construction and deformation. In *PhD Thesis, University Bordeaux I*, 1996.

[15] D. Marcheix and S. Gueorguieva. Topological operators for non-manifold modeling. In *Proc. of the Third International Conference in Central Europe on Computer Graphics and Visualisation 95*, 1995.

[16] F. Preparata and M. Shamos. *Computational Geometry: An Introduction.* Springer Verlag, 1985.

[17] A.G. Requicha and J.R Rossignac. Solid modelling and beyond. *IEEE ComputerGraphics&Applications*, pages 31–44, Septembre 1992.

[18] J.R. Rossignac. Representations, design and visualisation of solids and geometric structures. In *Tutorial notes, Eurographics'93*, 1993.

[19] J.R. Rossignac and A.O. Connor. Sgc: A dimension independant model for pointsets with internal structure and incomplete boundaries. In *Geometric Modeling for Product Engineering*, 1989.

[20] K. Shimada and D.C. Gossard. Bubble mesh: automated triangular meshing of non-manifold geometry by sphere packing. In *ACM Solid Modeling'95*, 1995.

[21] T. Takala. A taxonomy on geometric and topological models. In *Computer graphics and topological models.*, 1992.

[22] W.C. Thacker. A brief review of techniques for generating irregular computational grids. *International Journal for Numerical Methods in Engineering*, 15:1335–1341, 1980.

[23] W.C. Thacker, A. Gonzalez, and G.E. Putland. A method for automating the construction of irregular computational grids for storm surge forecast models. *Journal of Computational Physics*, 37:371–387, 1980.