MESH GENERATION ON HIGH-CURVATURE SURFACES BASED ON A BACKGROUND QUADTREE STRUCTURE

Antonio C. O. Miranda[1], Luiz F. Martha[2]

[1]*Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil - amiranda@tecgraf.puc-rio.br*
[2]*Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil - lfm@tecgraf.puc-rio.br*

ABSTRACT

This paper extends a previously proposed algorithm for generating unstructured meshes in three-dimensional and in two-dimensional domains to generate surface meshes. A surface mesh is generated in parametric space and mapped to Cartesian space. Finite elements may be stretched on parametric space, but they present a good-quality shape on the 3D surface. The algorithm uses a metric map defined by Tristano et al. to obtain correct distances and stretches. A background quadtree structure is used to store local surface metrics and to develop local guidelines for node location in an advancing-front meshing strategy.

Keywords: mesh generation, high-curvature surfaces, background quadtree, advancing-front technique

1. INTRODUCTION

This paper describes an algorithm for generating finite-element triangular meshes on surfaces of arbitrary shape and with high curvatures. It is an extension of a previously proposed algorithm for generating unstructured meshes in three-dimensional [1] and two-dimensional [2,3] domains. The mesh is generated in the surface's parametric space and mapped to 3D space. Finite elements may be stretched on parametric space, but they present a good-quality shape on the 3D surface. The algorithm uses a metric map defined by Tristano et al. [4] to obtain correct distances and stretches. A background quadtree structure is used to store local surface metrics and to develop local guidelines for the size of elements generated by means of an advancing-front technique.

The proposed algorithm presents similar characteristics to its ancestors, which are summarized as follows. First, the algorithm produces well-shaped elements, avoiding elements with poor aspect ratio. While it does not guarantee bounds on aspect ratios of elements, care is taken at each step to produce good-quality meshes. Second, the mesh conforms to an existing discretization on the surface's boundary. This is important in the generation of finite-element meshes because usually the mesh generated on a surface patch has to conform to the mesh generated on adjacent patches. Third, the algorithm presents a smooth transition between regions with elements of highly varying sizes. This a desirable feature because a finite-element analysis requires high element density in regions with high gradients of analysis response, while a low density may be used in other regions. It is not uncommon in this type of analysis to have a difference of two or three orders of magnitude in element size.

An additional requirement arises for surface mesh generation, especially when the surface presents high curvatures. In such locations, the algorithm must locally refine the mesh. Due to this fact, the smooth-transition characteristic described above is even more important in surface mesh generation.

Tristano and collaborators [4] proposed a similar algorithm for surface mesh generation that basically involves three steps: discretizing the boundary, computing a background mesh [5], and applying an advancing-front technique. This background mesh is generated by means of a Delaunay procedure. The main difference between the referred algorithm and the one presented in this paper is that here no auxiliary triangulation is generated. Instead, the current algorithm uses a quadtree structure to hold the metric information.

It should be noted that the original work of Tristano et al. also creates a quadtree in parametric 2D space for inserting internal points in the background mesh. In the present work, as in its ancestors [1,2,3], the quadtree is used to develop local guidelines for node location in an advancing-front meshing procedure. Here, the use of the quadtree was extended to store local surface metrics, avoiding the creation of a background triangulation. The background quadtree may potentially be adjusted to insert points to take into account local scalar field gradients, such as in adaptive analysis or in boundary layers problems, although this has not been done yet.

The present algorithm, as its ancestors, incorporates well-known meshing procedures [6-12] and introduces some original steps. It includes an advancing-front technique along with a background quadtree structure, taking special care to generate elements with the best possible shape. To enhance the quality of the mesh's element shape, an *a posteriori* local mesh improvement procedure is used.

One important characteristic of the algorithm presented here is the generation of internal nodes simultaneously with the elements. Some authors, e.g. Rassineux [12], use a quadtree/octree procedure to generate internal nodes prior to element generation. The current algorithm also employs a quadtree, but only as a node-spacing function during the advancing-front strategy. This approach tends to provide better control over the quality of the generated mesh and to decrease the amount of heuristic cleaning-up procedures.

2. MEASUREMENT ON SURFACE

This section describes the metric map [4] and the measurement of distances and angles on a surface space. Cuilière [13] originally devised the adopted surface metric. The metric of a tangent plane at every point P of a 3D surface is defined as:

$$[M]_p = \begin{bmatrix} E & F \\ F & G \end{bmatrix} \tag{1}$$

where

$$\begin{aligned} E &= \vec{\tau}_1 \cdot \vec{\tau}_1 \\ F &= \vec{\tau}_1 \cdot \vec{\tau}_2 \\ G &= \vec{\tau}_2 \cdot \vec{\tau}_2 \end{aligned} \tag{2}$$

and

$$\begin{aligned} \vec{\tau}_1 &= \vec{\sigma}_u(u,v) \\ \vec{\tau}_2 &= \vec{\sigma}_v(u,v) \end{aligned} \tag{3}$$

in which $\vec{\sigma}_u(u,v)$ and $\vec{\sigma}_v(u,v)$ are the gradient vectors at the point P on the surface.

The distance between two points, A and B, along the 3D surface can be computed using their parametric coordinates. This distance, starting from A, is:

$$\|AB\|_{MA} = \sqrt{E_A(u_B-u_A)^2 + 2F_A(u_B-u_A)(v_B-v_A) + G_A(v_B-v_A)} \tag{4}$$

For a *well-behaved* surface, the distance can be computed as the average of the distances measured from A and from B:

$$\|AB\| = \frac{\|AB\|_{MA} + \|AB\|_{MB}}{2} \tag{5}$$

For mesh generation on surfaces, some metrics are necessary to correctly compute angles and distances that might be stretched in parametric space. Figure 1 shows these measurements in parametric space.

The mid point m_{AB} is computed as

$$\vec{m}_{AB} = \frac{\vec{A}_{2D} + \vec{B}_{2D}}{2} \tag{6}$$

where $\vec{A}_{2D}$ and $\vec{B}_{2D}$ are the parametric coordinates of A e B, respectively. The vector $\vec{N}_{2D}$, normal to AB (in 3D space), is computed using the metric from equation (1):

$$\vec{N}_{2D} = \begin{Bmatrix} u_{N_{2D}} \\ v_{N_{2D}} \end{Bmatrix} = \begin{bmatrix} F_{m_{AB}} & G_{m_{AB}} \\ -E_{m_{AB}} & -F_{m_{AB}} \end{bmatrix} \cdot \vec{V}_{AB} \tag{7}$$

where $\vec{V}_{AB}$ is a unit vector pointing from A to B.

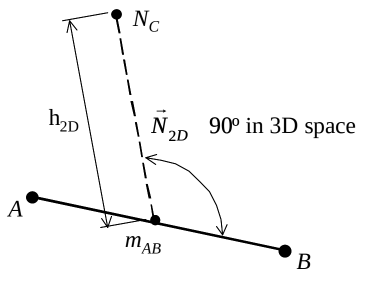

Figure 1. Measurements in parametric space.

In the advancing-front algorithm to be described in Section 3.2, it is necessary to compute the parametric coordinates of point N_C along the normal vector $\vec{N}_{2D}$ (see Figure 1). A given distance, h_{3D}, to point m_{AB}, along the 3D surface, defines the location of this point. The parametric coordinates of point N_C are computed using the equations below:

$$\vec{N}_{C_{2D}} = \vec{m}_{AB} + \vec{N}_{2D} \cdot h_{2D} \tag{8}$$

where

$$h_{2D} = \frac{h_{3D}}{\sqrt{E_{m_{AB}} u_{N_{2D}}^2 + 2F_{m_{AB}} u_{N_{2D}} v_{N_{2D}} + G_{m_{AB}} v_{N_{2D}}^2}} \tag{9}$$

3. DESCRIPTION OF THE ALGORITHM

The input data for the present algorithm is a parametric description of a surface and a polygonal description of the boundary of the surface patch to be meshed. This boundary information is given by a list of nodes defined by their parametric coordinates on the surface and a list of boundary segments (or edges) defined by their node connectivity. The input boundary must be defined *a priori*. The definition of these points is not part of the proposed algorithm and, for better results, the boundary segment sizes should be consistent with surface curvatures. Figure 2 shows an example of a surface patch and a set of input boundary segments.

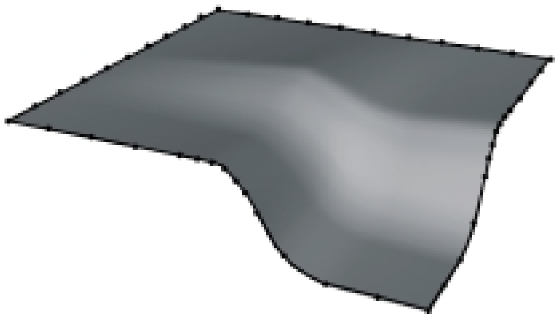

Figure 2. Surface example and input boundary segments.

From the boundary segments, a background auxiliary quadtree structure is created to control the sizes of the finite elements generated by an advancing-front technique. The given boundary edges form the initial front that advances as the algorithm progresses. At each step of this meshing procedure, a new triangle is generated for each base edge of the front. The front advances replacing the base edge with new triangle edges. Consequently, the domain region is contracted, possibly into several regions. The process stops when all contracted regions result in single triangles. The overall algorithm is presented next.

3.1 Quadtree Generation

The first phase of the algorithm involves a background quadtree generation. The steps in this phase are as follows:

- *Quadtree initialization based on given boundary edges.* Each segment of the input boundary data is used to determine the local subdivision depth of the quadtree. Additional procedures, described below, are adopted in the generation of the initial quadtree to take into account metric distortions in surface parametric space.
- *Refinement to force maximum cell size.* The quadtree is refined to guarantee that no cell in its interior is larger than the largest cell at the boundary.
- *Refinement to provide minimum size disparity for adjacent cells.* This additional refinement forces only one level of tree depth between neighboring cells and provides a natural transition between regions of different degrees of mesh refinement.
- *Refinement to force minimum curvature difference between adjacent cells.* This is explained below.

The first step has some modifications in relation to the original 2D algorithm [2,3]. The second and third steps have not changed. The fourth step was added to take into account high surface curvatures.

There are some additional procedures in the first step of the quadtree generation. The algorithm creates a two-dimensional space to store coordinates of quadtree points. This space is necessary to avoid excessive metric distortions that might occur in the parametric space of a surface. Distances in the quadtree space approximate distances in 3D space along the surface.

Firstly, the algorithm determines the center of the input boundary data in parametric coordinates ($Center_x$, $Center_y$). From this center, the algorithm computes the longest horizontal and vertical distance to the boundary, using equations (4) and (5). Then, it computes ratios R_x and R_y between the longest distance and the corresponding parametric distances. These ratios are used as proportional factors to transform from surface parametric space to quadtree space. The center of the quadtree is considered at the center of the input boundary data. The initial quadtree cell size, which includes the whole model, is computed as twice the longest distance.

Each segment of the input boundary data is used to determine the local depth of tree subdivision. The midpoint of each input segment in parametric space (Mid_x, Mid_y) is obtained. The corresponding values in quadtree space (Mid_{quad_x}, Mid_{quad_y}) are obtained by:

$$\begin{aligned} Mid_{quad\_x} &= \left(Mid_x - Center_x\right)\cdot R_x + Center_x \\ Mid_{quad\_y} &= \left(Mid_y - Center_y\right)\cdot R_y + Center_y \end{aligned} \tag{10}$$

The quadtree cell containing the midpoint is determined. If the length, computed using equations (4) and (5), of this cell's edge is larger than the length of the boundary edge, then this cell is subdivided into four smaller cells. This process is repeated recursively and finishes when the length of the cell's edge is smaller than the length of the boundary segment.

The fourth step of the quadtree generation refines the quadtree to force a minimum curvature difference between adjacent cells. Firstly, the algorithm stores gradient vectors in each cell of the quadtree – equation (3) – evaluated at the center of the cell. Then, it computes the vector normal to the 3D surface at each cell. Finally, the algorithm obtains the cosine of the angle between the normal vectors, N_A and N_B, of the two adjacent cells:

$$\cos\theta = \frac{N_A \cdot N_B}{\|N_A\|\|N_B\|} \tag{11}$$

and compares it to a minimum value, $\cos\theta_{\min}$. If $\cos\theta$ is less than $\cos\theta_{\min}$, then a new cell size, H_{new}, is obtained from the current size, H_{old}, as:

$$H_{new} = \frac{H_{old}}{\cos\theta_{\min}} \cdot \cos\theta \tag{12}$$

This new size is used to locally refine both adjacent cells of the quadtree. This process is repeated recursively to every cell. After the whole quadtree is refined according to the curvature criterion, the third step of the algorithm is repeated to provide minimum size disparity for adjacent cells.

Figure 3 shows the final quadtree obtained for the surface in Figure 2 after all four steps. The quadtree is used as a node-spacing function in the advancing-front technique to be described in next section. In addition, each cell of the quadtree stores local metric information (in this case, gradient vectors) that is used to compute distances and angles necessary for mesh generation.

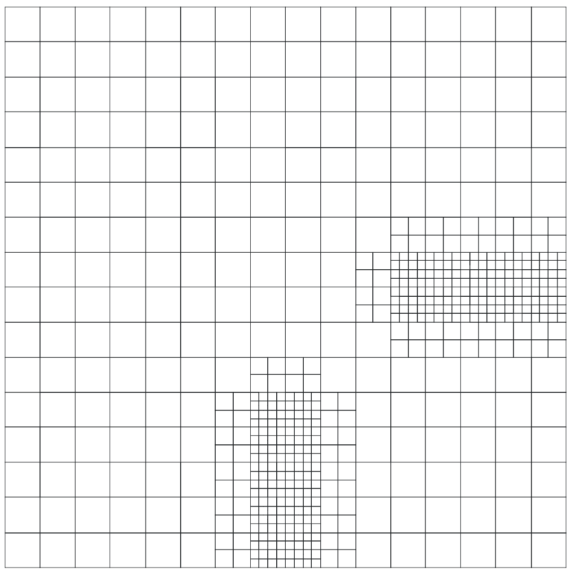

Figure 3. Background quadtree of surface in Figure 2.

3.2 Advancing-front Procedure

The advancing-front technique starts with a boundary that bounds a region to be filled with a triangulation. Triangular elements are “extracted” or “pared” from the region one at a time. As each element is extracted, the boundary is updated and the process is repeated. The procedure terminates when the entire region is meshed. Therefore, the boundary of the region to be meshed is formed by edges of the triangles created in the contraction process. These edges are referred to as *boundary edges*.

In this algorithm, as in its ancestors [1,2,3], the advancing-front process is divided into two phases to ensure the generation of valid triangulations. In the first phase, a geometry-based element generation is pursued to generate elements of optimal shapes. After this ideal phase is exhausted and no more optimal elements can be generated, a topology-based element generation takes place, creating valid, but not necessarily well shaped, elements in the remaining region. The required steps for the advancing-front procedure are as follows.

Front initialization based on given boundary edges. The process starts with the creation of the initial advancing front, which is formed by the given boundary segments. The current boundary edges are stored in two separate doubly linked lists. The first is a list of active edges, which includes all boundary edges that have not been used in an attempt to generate valid triangles. The other is a list of rejected edges, that is, it stores the edges that failed in the generation of elements for the current phase. Initially, all segments of the given boundary refinement are stored in the first list, which is the one used in the geometry-based generation phase.

The initial list of active edges on the boundary is sorted by the length of the edges. This has been recommended by other authors [11] to prevent large elements from penetrating regions with small-length edges. This criterion is only used in the initial boundary edge list.

It was also found convenient for some steps in the algorithm to have an additional data structure storing a list of adjacent boundary edges for each node on the current advancing front. This data structure is initialized for all nodes of the given boundary, and is updated as the boundary-contraction procedure progresses.

In addition, all nodes (from the input boundary as well as new internal nodes) store their corresponding gradient vectors, obtained from the quadtree cells, as they are created. As seen in Section 2, these vectors are used in the surface measurements required by the advancing-front procedure. This avoids excessive queries to the quadtree structure, which could be expensive.

Front contraction (geometry-based element generation). Ideally, the entire mesh could be generated in the geometry-based phase. This depends on the geometry and topology of the given boundary model and, as observed, is strongly related to the segment-size disparity of the given boundary refinement. In this phase, for each base edge on the advancing front, the following is performed (see Figure 4):

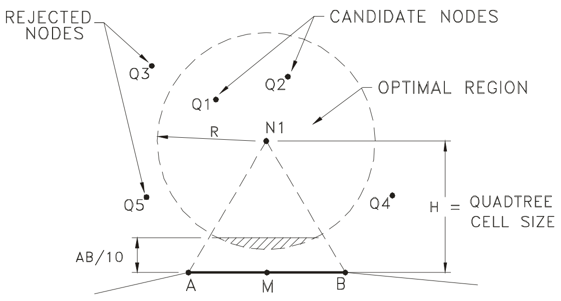

Figure 4. Determination of a triangle.

- The optimal location *N1*, in parametric space, for the vertex of the triangle to be formed is determined with the help of the quadtree. The quadtree cell containing the midpoint *M* of the base edge is determined using equation (10). The optimal point *N1* lies on a line perpendicular (in the 3D space) to the base edge passing through this midpoint. The distance from the optimal point to the base edge midpoint is computed using equations (6), (7), (8) and (9), in which h_{3D} is equal to the quadtree cell size, *H*. The gradient vectors of point *M* are considered as the average between the values of points *A* and *B*.

- The optimal point defines an optimal region where the vertex of the triangle to be generated is located. This region is a sector of the circle whose center is the optimal point and whose radius is proportional to the quadtree cell size. In the current implementation, a factor of 0.85 was adopted. Radius and distances are computed using equations (4) and (5). This circle defines an upper bound for the distance between the target vertex of the triangle and the centroid of the base edge. A lower bound is defined to ensure that the generated triangle will have area greater than the smallest acceptable area. In the current implementation, this lower bound is defined by a triangle with height equal to 1/10 of the base edge. The optimal region is used for two reasons: first, to ensure shape quality of the elements to be generated; and, second, to ensure that new internal nodes will be created only when it is strictly necessary and always in good positions.

- If no existing node is inside the optimal region, a new node is inserted at the optimal location *N1* and an element is generated using this node. The values of gradient vectors to point *N1* are held from the quadtree. If only one node exists in the region, this node is used to generate the element. If more than one node is found in the region, they are ranked according to the included angle with respect to the base edge. The node with the maximum included angle is used to generate the element. A heap list is used to efficiently rank the nodes.

- Additional geometric checks are performed to ensure that the edges of the new element do not intersect any existing edge of the advancing front. If this is the case, the element is rejected.

- Once a valid triangle is generated for the current base edge, the list of active edges is updated. This is done through the following steps: first, the base edge is removed from the list; then, for the other edges of the element, the edge is either deleted, if it coincides with an edge already in the list, or inserted in the list as a new one.

- Due to geometric bounds imposed by the current advancing front, there are situations in which the algorithm fails in forming a valid triangle for the current boundary's base edge. In these cases, the current base edge is removed from the list of active edges and is stored in the separate list of rejected edges. It might happen that an edge is subsequently removed from this latter list if it is used as part of a valid triangle for an adjacent base edge.

- When there are no more edges in the list of active edges, the algorithm tries to generate elements using the edges that were previously rejected. It might be the case that base edges that previously failed may now work because the front has changed with the addition of elements. The geometry-based element-generation phase ends when either there are no edges left in the boundary-contraction lists (in which case an optimal mesh was generated) or when a rejected edge fails for a second time.

Front contraction (topology-based element generation). The objective of this phase of the algorithm is to force the generation of valid triangles, even if the new elements do not satisfy the bounds used in the previous phase for element shapes.

The topology-based element-generation phase starts when a boundary edge fails twice in trying to generate an optimal element. The list of rejected edges of the previous phase is transformed into a list of active edges and, similarly to the geometry-based phase, a list of rejected edges is created for edges that eventually fail in generating valid triangles.

In the topology-based element-generation phase, any node close to the current base edge is selected and stored in the local heap list of candidate nodes. The node that has the maximum included angle with respect to the base edge is chosen for the generation of the new triangle. If the edges of this triangle do not intercept any other edge of the current advancing front, the element is created and the boundary is contracted accordingly. The topology-based phase ends when the lists of active and rejected edges are empty. This phase always generates a valid mesh (although not optimal) because it is always possible to triangulate a region defined by its boundary edges [14].

3.3 Local Mesh Improvement

A smoothing technique is used to improve mesh quality by relocating nodes within a patch. A general formulation for this technique is given by equation (13), which is a generic form of a weighted Laplacian function [15]:

$$X_0^{n+1} = X_0^n + \phi \frac{\sum_{i=1}^{m} w_{i0}(X_i^n - X_0^n)}{\sum_{i=1}^{m} w_{i0}} \tag{13}$$

In this equation, m is the number of nodes connected to node O, X_0^{n+1} is the parametric position of node O at smoothing iteration $n+1$, w_{i0} is the weighted function between nodes i and O, and ϕ is a relaxation parameter which is normally set in the interval *(0,1]*. In this work, a value of $\phi = 0.7$ is defined and w_{i0} is adopted as the ratio between the 3D distance from node i to node O and the corresponding parametric distance. Therefore, smoothing is done in parametric space but takes into account size distortion metrics between parametric and 3D spaces. The smoothing procedure is repeated 5 times for all internal nodes.

Figure 5 presents the final mesh obtained for the surface example in Figure 2.

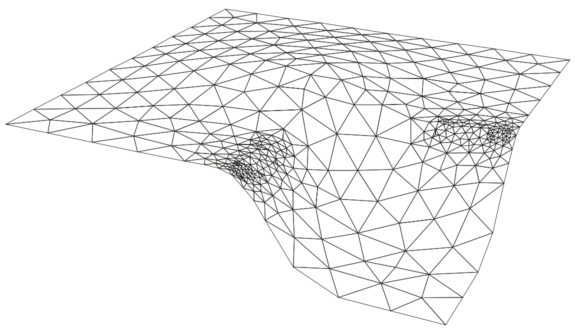

Figure 5. Generated mesh of surface of Figure 2.

4. EXAMPLES

This section provides some examples of finite-element meshes generated on 3D surfaces using the proposed algorithm. Figure 6 shows an example a mesh generated on a conic surface and its representation in parametric space. It may be observed that in parametric space the elements are stretched, while on the 3D surface the elements have a good shape.

The second example is a single-folded surface, shown in Figure 7. The mesh at the top of this figure was generated without taking into account surface curvatures, i.e., without considering the fourth step of the background quadtree generation described in Section 3.1. The mesh at the bottom of Figure 7 was generated for the same surface now considering the fourth step in the algorithm.

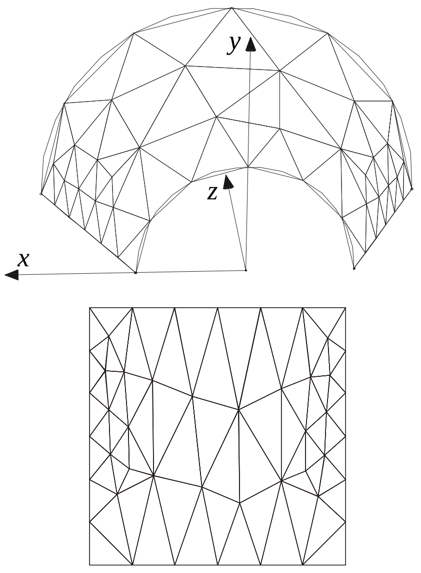

Figure 6. Mesh generated on a conic surface and its representation in parametric space.

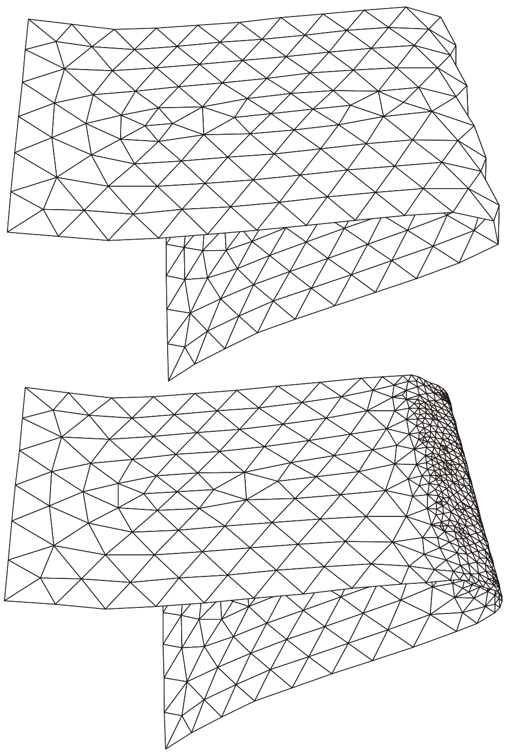

Figure 7. Mesh generated on a single-folded surface without and with consideration of surface curvatures.

The third example is a double-folded surface, shown in Figure 8 without and with local refinement to consider high surface curvatures.

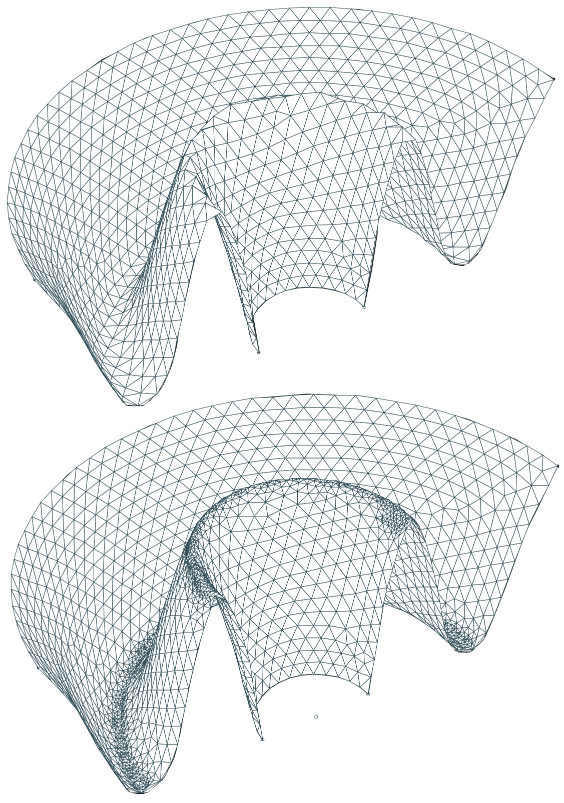

Figure 8. Meshes generated on a double-folded surface.

Finally, Figure 9 illustrates the so-called Utah teapot, which is composed of several surface patches. It can be seen that the mesh generated on a patch conforms to the meshes generated on adjacent patches. This is accomplished because all the common curves at the patches' boundaries are discretized *a priori*.

5. MESH QUALITY AND PERFORMANCE

In this section, a study on the quality of the meshes generated by the proposed algorithm is presented. The adopted shape quality measure is a normalized ratio γ/γ* [2], in which γ is the ratio between the root mean square of the lengths (S_i) of a triangle's edges and the triangle's area, and γ* is the value for the equilateral triangle:

$$\gamma = \sqrt{\tfrac{1}{3}\sum_{i=0}^{3} S_i^2} \Big/ Area \qquad (14)$$

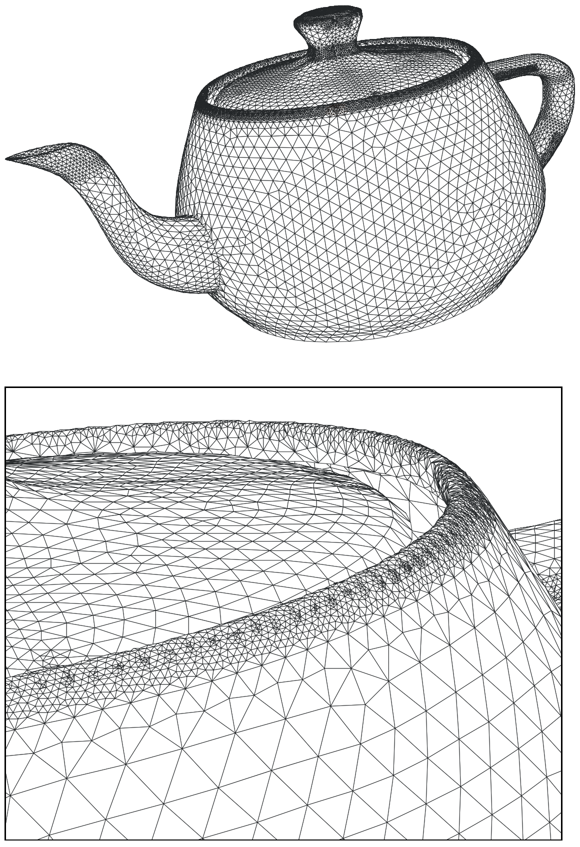

Figure 9. Surface mesh generated for the Utah teapot.

The γ/γ* quality measure has a valid interval between 1.0 and infinity, and the value for the equilateral triangle is 1.0. It is desirable to have elements with values close to 1.0.

The quality of generated meshes is presented in the form of a histogram such as the one shown in Figure 10. In this histogram, the horizontal axis corresponds to the γ/γ* quality measure in intervals represented by triangular shapes that are shown below the histogram. The vertical axis corresponds to the percentage of elements in each interval of the quality measure.

The results of two examples are shown in Figure 10: the single-folded mesh of Figure 7 and the double-folded mesh of Figure 8 (both considering surface curvatures). These results demonstrate that the proposed algorithm generates meshes with good quality for the great majority of elements.

An estimate of the expected performance of the 3D and 2D versions of the proposed algorithm has been presented in previous works [1,2]. Surface mesh generation requires additional computations of surface metrics. Therefore, it is worthwhile to present timing data outlining time spent in each phase of the algorithm. Table 1 shows processing times for the single-folded and double-folded meshes. The performance of the algorithm was measured running on a

Pentium-650MHz PC with 128 MB of RAM, under Windows 2000 operating system.

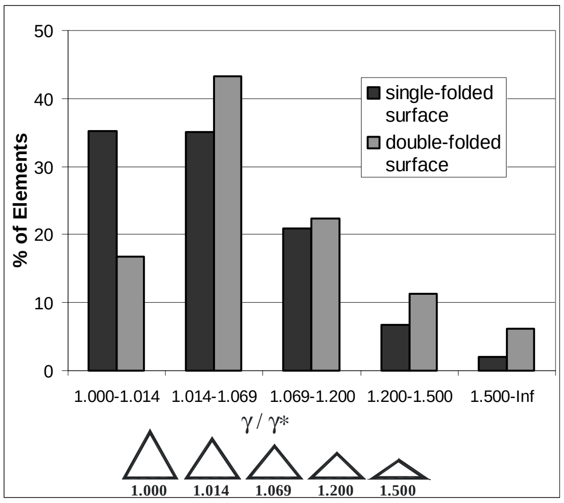

Figure 10. Histogram of element quality.

Table 1. Processing time for single-folded and double-folded surfaces.

Folded Surface	Single	Double
Quadtree Generation [sec.]	0.09	0.11
Surface Metrics [sec.]	1.95	6.59
Boundary Contraction [sec.]	0.16	2.03
Total Time [sec.]	2.11	8.63
Number of Elements	1356	5669
Number of Elements / second	641.7	657.5

In Table 1, it is clear that most of the computational effort is spent in surface metrics calculation – equations (1), (2), and (3). The computation of these metrics depends on the underlying surface parametric representation that provides surface gradients. This is not part of the proposed meshing algorithm, any surface parametric representation may be used. The present implementation uses a public domain NURBS library [16].

The total time of surface metric computation depends on the complexity of the surface. An almost flat surface requires a smaller amount of quadtree refinement than a surface with high curvatures. Since surface metrics are computed for each cell of the quadtree, a greater amount of metrics computation is required for a surface with high curvatures.

It is also interesting to observe in Table 1 that the processing time for quadtree generation is minimum compared to the other phases of the algorithm.

6. CONCLUSION

This paper has described an algorithm for generating finite-element triangular meshes on surfaces of arbitrary shape and with high curvatures. The algorithm incorporates aspects of well-known meshing procedures and includes some original steps.

This algorithm is an extension of a previously proposed algorithm for generating unstructured meshes in three-dimensional and two-dimensional domains. The mesh is generated in the surface's parametric space and mapped to 3D space. Finite elements may be stretched on parametric space, but they present a good-quality shape on the 3D surface.

Previous works have demonstrated the computational efficiency of the proposed meshing scheme [1,2], which should be maintained in the present case of surface mesh generation. However, it was shown that surface metrics calculation is an expensive additional computation.

The algorithm uses a metric map defined by Tristano et al. to obtain correct distances and stretches. Differently from their work, the present algorithm uses a quadtree structure, instead of a background triangulation, to hold surface metric information. The background quadtree is also used to develop local guidelines for the size of elements, which are generated by means of an advancing-front technique.

The input data for the present algorithm is a parametric description of the surface and a polygonal description of the boundary of the surface patch to be meshed. The steps in the algorithm are as follows:

- A background quadtree is generated to control the distribution of node points generated in the interior. The quadtree refinement is defined by the given boundary discretization and by surface curvatures.
- A two-pass advancing-front procedure is used to generate elements. On the first pass, elements are generated based on geometrical criteria, which produce well-shaped elements. On the second pass, elements are generated based only on the criterion that they have valid topology.

Some examples have demonstrated the quality of the generated meshes and compared results with and without local refinement to consider high surface curvatures.

Finally, it should be mentioned that the use of a background quadtree structure makes the proposed algorithm well suited for adaptive finite-element analysis. An additional step is required in the generation of the quadtree to account for the refinement due to estimated numerical errors. This has been used in two-dimensional analysis [17] and the implementation for adaptive surface mesh generation is straightforward.

ACKNOWLEDGEMENTS

The first author acknowledges a doctoral fellowship provided by CNPq. The second author acknowledges financial support by CNPq (Project 300.483/90-2). The authors acknowledge the financial support provided by agency Finep/CTPetro, processes numbers 65999045400 and 2101033200. The authors are grateful to Carolina Alfaro for the manuscript copydesk. The present work has been developed in Tecgraf/PUC-Rio (Computer Graphics Technology Group).

REFERENCES

[1] J. B. Cavalcante Neto, P. A. Wawrzynek, M. T. M. Carvalho, L. F. Martha, and A. R. Ingraffea, “An Algorithm for Three-Dimensional Mesh Generation for Arbitrary Regions with Cracks”, *Engineering with Computers,* Vol 17(1) pp.75-91 (2001).

[2] A. C. O. Miranda, J. B. Cavalcante Neto, and L. F. Martha, “An Algorithm for Two-dimensional Mesh Generation for Arbitrary Regions with Cracks”, *SIBGRAPI’99 – XII Brazilian Symposium on Computer Graphics, Image Processing and Vision*, IEEE Computer Society Order Number PRO0481, ISBN 0-7695-0481-7, Eds.: J. Stolfi & C. Tozzi, pp.29-38 (1999).

[3] A. C. O. Miranda, M. A. Meggiolaro, J. T. P. Castro, L. F. Martha, and T. N. Bittencourt, “Fatigue Life and Crack Path Predictions in Generic 2D Structural Components”, accepted for publication in *Engineering Fracture Mechanics* (2002).

[4] J. R. Tristano, S. J. Owen, and S. A. Canann, “Advancing Front Surface Mesh Generation in Parametric Space Using Riemannian Surface Definition”, *7th International Meshing Roundtable Proceedings*, pp.429-445 (1998).

[5] S. J. Owen and S. Saigal, “Neighborhood-Based element sizing Control for Finite Element Surface Meshing”, *6th International Meshing Roundtable Proceedings*, pp.143-154 (1997).

[6] C. T. Chan and K. Anastasiou, “An Automatic Tetrahedral Mesh Generation Scheme by the Advancing Front Method,” *Communications in Numerical Methods in Engineering*, Vol 13 pp.33-46 (1997).

[7] H. Jin. and R. I. Tanner, “Generation of Unstructured Tetrahedral Meshes by Advancing Front Technique”, *International Journal for Numerical Methods in Engineering*, Vol 36 pp.1805-1823 (1993).

[8] S. H. Lo, “A New Mesh Generation Sheme for Arbitrary Planar Domains”, *International Journal for Numerical Methods in Engineering*, Vol 21 pp.1403-1426 (1985).

[9] R. Lohner and P. Parikh, “Generation of Three-dimensional Unstructured Grids by the Advancing-front Method”, *International Journal for Numerical Methods in Fluids*, Vol 8 pp.1135-1149 (1988).

[10] P. Moller and P. Hansbo, “On Advancing Front Mesh Generation in Three Dimensions”, *International Journal for Numerical Methods in Engineering*, Vol 38 pp.3551-3569 (1995).

[11] J. Peraire, J. Peiro, L. Formaggia, K. Morgan, and O. C. Zienkiewicz, “Finite Euler Computation in Three-Dimensions”, *International Journal for Numerical Methods in Engineering*, Vol 26 pp.2135-2159 (1988).

[12] A. Rassineux, “Generation and Optimization of Tetrahedral Meshes by Advancing Front Technique”, *International Journal for Numerical Methods in Engineering*, Vol 41 pp.651-674 (1998).

[13] J. C. Cuilière, “And Adaptive Method for the Automatic Triangulation of 3D Parametric Surfaces”, *Computer Aided Design*, Vol 30 pp.139-149 (1998).

[14] J. O'Rourke, *Art Gallery Theorems and Algorithms.* Oxford University Press (1987).

[15] T. A. Foley and G. M. Nielson, “Knot Selection for Parametric Spline Interpolation”, *Mathematical Methods in CAGD*, Ed. L. Schumaker, Academic Press, pp.445-467 (1989).

[16] P. Lavoie, *NURBS++: The Nurbs Package - User’s Reference Manual*, http://yukon.genie.uottawa.ca/~lavoie/software/nurbs.

[17] G. H. Paulino, I. F. M. Menezes, J. B. Cavalcante Neto, and L. F. Martha, “A Methodology for Adaptive Finite Element Analysis: Towards an Integrated Computational Environment”, *Computational Mechanics*, Vol 23(5/6) pp. 361-388 (1999).