Two triangulations methods based on edge refinement

Marc Vigo, Núria Pla, Dolors Ayala

Research Report LSI-03-10-R

Computer Graphics Section — Dept. LSI.
Universitat Politècnica de Catalunya

Research report LSI-03-10-R

Two triangulations methods based on edge refinement

Marc Vigo, Núria Pla, Dolors Ayala

Escola Tècnica Superior d'Enginyeria de Barcelona
Universitat Politècnica de Catalunya

February 2003

Abstract

In this paper two curvature adaptive methods of surface triangulation are presented. Both methods are based on edge refinement to obtain a triangulation compatible with the curvature requirements. The first method applies an incremental and constrained Delaunay triangulation and uses curvature bounds to determine if an edge of the triangulation is admissible. The second method uses this function also in the edge refinement process, i.e. in the computation of the location of a refining point, and in the re-triangulation needed after the insertion of this refining point. Results are presented, comparing both approaches.

Keywords: Delaunay triangulation, regular triangulation, meshing, trimmed surfaces, curvature adaptive

1 Introduction

The problem we are focusing in this paper is the approximation of a solid boundary by a triangulation. The solid boundary is composed by trimmed surfaces and each surface is composed by several parametric patches.

The approximation of surfaces by a mesh of triangles is a process which is used in many fields such as surface rendering, rapid prototyping and finite element mesh among others. Many approaches have been presented to deal with this problem. There are approaches oriented to visualization as [12, 6, 10, 11], and to FEM generation [16, 15, 2, 4, 8] and there are also general purpose approaches [14, 3, 21, 7].

While the earlier methods were global [12, 14], most recent methods are adaptive to several measures – mostly related with the curvature. The method [2] is adaptive to a user defined density that can be the curvature. In [3] the authors present a method adaptive to an alternative curvature measure, the root mean square curvature, and for each triangle they extract the corresponding triangular sub-patch and use its convex hull to evaluate the upper bound. The approach presented in [16] uses the bubble mesh

method, similar to a particle system, and obtains a triangulation which is adaptive to the analyzed magnitude such as heat or torsion.

In [15] the bubble mesh method is extended to anisotropic triangulations. The problem of anisotropic or directional triangulations is also addressed in [7], [20] and [21].

Several methods use advanced front strategies to obtain an adaptive triangulation [4, 8, 7], whereas others are based on Delaunay triangulation (most of already cited methods).

In this paper two curvature adaptive methods of surface triangulation are presented. We are given a set of trimmed parametric surfaces describing a closed solid. Then, we obtain function $R(p)$ defined at each parametric point, that indicates an upper bound for an edge length in order to follow a determined curvature criterion. Both methods we propose are based on edge refinement to obtain a triangulation compatible with the curvature requirements. The first method, presented in the next section, applies an incremental and constrained Delaunay triangulation and uses function $R(p)$ to determine if an edge of the triangulation is admissible. The second method, presented in section 3, uses this function also in the edge refinement process, i.e. in the computation of the location of a refining point, and in the re-triangulation needed after the insertion of this refining point.

2 The first triangulation algorithm

The algorithms we propose approximate a set of surface patches that bound a closed object with a triangular mesh. the triangulation must be conformal (i.e. no gaps between neighbor triangles must be created) and admissible (i.e. it must not be farther than at a tolerance value ε supplied by the user). Formally, we define an *admissible triangle* as a triangle such that for each point on t the maximum distance form the point to the surface it approximates is smaller than ε.

The scheme of our triangulation algorithms is the following:

1. Retrieve and preprocess input data.
2. Analyze the surface curvature and compute the admissibility bounds.
3. Set vertices and discretize edges (trimming curves), by bisection.
4. Triangulate the interior of each face, by edge refinement.
5. Map triangles to image space.

The two algorithms proposed in this paper follow the same scheme. Essentially, they differ in the method used for triangulating the edges and the interior of the faces: while the kernel of the edge refinement algorithm of the first proposal is a constrained Delaunay triangulation, the second approach uses a regular triangulation.

Next, we describe the main steps of the above algorithmic scheme; steps 2 and 4, that require a more precise description, are treated in the following subsections.

The first step reads the data from a formated file and stores it using a computer model of a closed object. The model we use is a surface boundary representation. Faces of the object are trimmed parametric patches, and edges are curves on a patch that trim the boundary of the face. Although this step seems relatively easy, we found some problems while retrieving data stored in standard file CAD formats by commercial systems. The preliminary treatment of the patches can include preprocesses, such as

conversion from degenerate patches into non-degenerate trimmed patches [19] or a linear reparameterization of the surfaces [18].

The second step of the algorithm consists on subdividing the patches into minimal subdivision regions in the parametric plane and bounds of the second derivatives are computed for each minimal region. The bounds are summarized using a scalar function $R : \mathbb{R}^2 \rightarrow \mathbb{R}$ that limits the size of the admissible triangles and edges (see next section for a formal definition of an admissible edge). Function R, which allows us to easily test the admissibility of a given triangle from its vertex coordinates, is more accurately described in Section 2.1. The usage of these bounds is the clue for the resulting triangulation to be adaptive to the surface curvature.

The triangulation of the object is performed following the topological hierarchy: first, points are placed on the vertices, next trimming edges are approximated, and finally the interior of the faces is triangulated. This technique, also followed by other meshing algorithms such as [16] and [4], ensures that the output mesh will be conformal without having to sew triangulations of neighboring patches. Instead, trimming curves are approximated before the interior of patches, taking into account the bounds computed on the two meeting faces (in the same way than [1]). This discretization of the edges (trimming curves of the faces) is performed by bisection, i.e. halving non-admissible edges by adding its middle point. Thus, the third step produces admissible polygonal edges, and the interior of the faces is ready to be filled with triangles.

Triangulating the faces is the main course of the algorithm. Patches are triangulated in parametric space following a Delaunay-based refining method. The method starts from a constrained Delaunay triangulation of the contour and adds points one by one in strategic places so that the resulting triangulation is composed of a reduced set of admissible triangles. This step, the crux of our triangulation method, is accurately described in the next section. In a certain way, this approach is similar to the one proposed in [2] (it is also an incremental refining method), but in our case the algorithm entirely works in parametric space, produces an admissible triangulation, and admits sets of neighbor patches as well as open boundaries. We also managed to achieve a theoretical bound on the number of vertices added by our method (see Section 2.3).

Last step of the algorithm produces the final mesh in 3D. Since the approximations of edges and curves work in parametric space, all that remains is to project the triangles into image space, in other words, to evaluate 3D coordinate vertices. Special care has to be taken in the case of the vertices of the object and the polygonalized edges, which have to be projected in the same way for the different patches converging at them. In case that the output triangulation must be used for visualization purposes, normal vectors at the mesh vertices can also be computed in order to apply Phong shading.

2.1 Curvature bounds

The main goal of this section is to obtain conditions for easily testing the admissibility of triangles, and dependent on the curvature of each point of a patch (*i.e.*, adaptive). Following the results given in [5] and [14], two functions are defined:

Definition 1.

$$\Omega(p) = 3\sqrt{\frac{\varepsilon}{M(p)}} = 3\sqrt{\frac{\varepsilon}{2\|M_{uu}(p)\| + 4\|M_{uv}(p)\| + 2\|M_{vv}(p)\|}}$$

being M_{uu}, M_{uv} and M_{vv} are the second derivatives of surface being analyzed and p is a point in parametric space.

Definition 2.

$$R(p) = \min\{f_p(q),\ \forall q \text{ t.q. } \|pq\| \leq \Omega(p)\}$$

where

$$f_p(q) = \max\{\Omega(q), \|pq\|\}$$

As it is proved in [18], function R bounds the length of the largest edge of an admissible triangle with point p as one of its vertices. An admissible edge is defined as follows:

Definition 3. A segment AB is admissible if and only if

$$R(A) \geq \|AB\| \quad \text{or} \quad R(B) \geq \|AB\|$$

This results in a simple condition for a triangle to be admissible, that can be tested based only on the vertices of the triangle:

Proposition 1. *A sufficient (but not necessary) condition for a triangle to be admissible is that their three edges are admissible.*

Definition 4. $C_R(p)$ is a circle centered at p and with radius $R(p)$.

It is also proved in [18] that function R is Lipschitz continuous, with Lipschitz constant equals to the unit:

Lemma 1.

$$\forall\ a, b \in \mathbb{R}^2 \qquad |R(a) - R(b)| \leq \|ab\|$$

This fact is exploited to bound the number of internal vertices when triangulating each face by edge refinement.

2.2 The refining edges method

We start from a solid whose edges have yet been discretized, approximating the trimming curves with admissible (straight) segments. Thus, the aim of the refining edges method is to produce a triangulation of the interior of the polygon, trying to minimize the number of internal vertices. Since this polygon corresponds to the face, it may contain holes – internal closed cycles of segments. Recall that the admissibility of an edge can be tested simply comparing its length with the R value at its endpoints.
The refining edges algorithm is the following:

CDT := Constrained Delaunay triangulation of the polygon
L := EmptyListOfEdges()
Append all the non-admissible edges of CDT to L
While L is not empty **do**
    Extract e, the first edge from L
    If e must be refined **then**
        x := Refining point of e
        InsertPointCDT(x, CDT)
        Actualize L according with the non-admissible edges of CDT
    endIf
endWhile

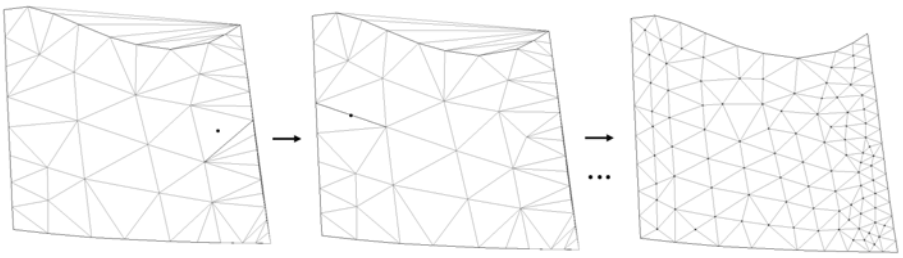

Figure 1: The edge refinement algorithm in action. *left and middle,* two intermediate steps, showing the edge that is refined and the new point added. *Right,* when the refinement converges, an admissible triangulation is obtained.

The method used for obtaining a constrained Delaunay Triangulation (CDT) of a planar graph is the one presented in [17]. Since this CDT algorithm works incrementally, it allows to efficiently actualize a CDT when single points and restricting edges are added on it.

Since the incremental inclusion of a new point X in a CDT changes the triangulation locally (only some new edges connecting x will appear), the step that actualizes the list of edges L is efficient because it only has to rearrange things in a neighborhood of x. Figure 1 shows an example of the edge refinement process being applied to a face, with two intermediate steps and the final state.

Whether an edge needs to be refined or not, and the position of the refining point x depends on the configuration of the triangles adjacent to that edge. From now on, the non-admissible edge to be refined will be called ab, the adjacent triangles will be $T(a, b, c)$ and $T(a, d, b)$, and the refining point will be x. Since function R classifies ab as a non-admissible edge,

$$R(a) < \|ab\| \quad \text{and} \quad R(b) < \|ab\| \tag{1}$$

We have two possible configurations of adjacent triangles to a candidate edge to be refined (see Figure 2): in the first case, the edge ab cuts its dual in the Voronoi diagram; in the second case, this circumstance does not occur. The new point inserted in the first case will be the midpoint of edge ab, and in the second case, x will be the circumcenter of one of the two adjacent triangles, but only if this point is interior to the region that is going to be triangulated; otherwise we will prove that no additional point is needed.

Case 1: Edge ab cuts its dual. In this case x is the midpoint of ab, $x = \frac{a+b}{2}$, which coincides with the intersection between ab and its dual edge in the Voronoi diagram.

When x is inserted into the CDT, the new triangulation will contain edges ax, bx, cx and dx, while edge ab will be removed since x is interior to the circumcircles of triangles $T(a, b, c)$ and $T(a, d, b)$. Furthermore, since x is on the Voronoi edge that splits tiles associated to a and b, there is no point closer to x than a and b themselves. In consequence, a circle with radius $r = \frac{\|ab\|}{2}$ centered on x does not contain any point of the original CDT, thus new edges that will appear when inserting x into the triangulation must have at least length r. From equation 1 and lemma 1, one gets

$$R(a) < \|ab\| \Longrightarrow R(x) < \|ab\| + \|ax\| = 3r \Longrightarrow r > \frac{1}{3}R(x)$$

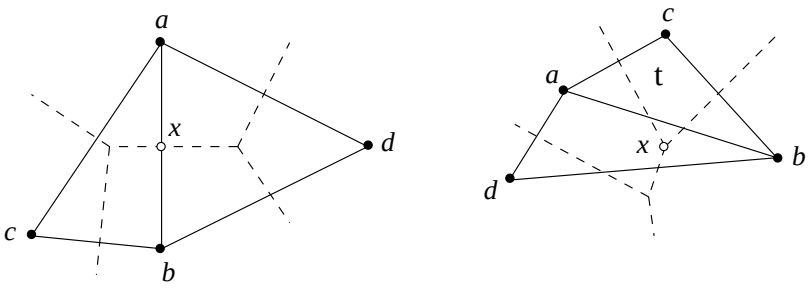

Figure 2: The two cases of the refining edge method, depending on the configuration of adjacent triangles. Voronoi diagrams are drawn using dashed lines.

Case 2: Edge ab does not cut its dual. The endpoints of the Voronoi edge dual to ab are the circumcircles of triangles $T(a, b, c)$ and $T(a, d, b)$. Since ab does not cut its dual, the endpoints must lie outside one of these triangles; let's suppose, without loss of generality, that this triangle is $t = T(a, b, c)$ (see Figure 2). In this case, x, the candidate point to be inserted, is the circumcircle of t. Remark that in this case the midpoint of ab would not be a good candidate because it can be arbitrarily close to c, thus a lower bound for the length of edge cx could not be found.

Point x can be outside the original polygon (the region that it is going to be triangulated), so we have two sub-cases, depending on whether cx cuts the boundary polygon.

Case 2.A: Segment cx does not cut the boundary polygon. Since x is the circumcenter of t, from equation (1) we have

$$\|ax\| = \|bx\| = \|cx\| \geq \frac{1}{2}\|ab\| > \frac{1}{2}R(a)$$

When x is inserted, edge ab will be removed and in its place will appear the new edges ax, bx and cx (see Figure 3).

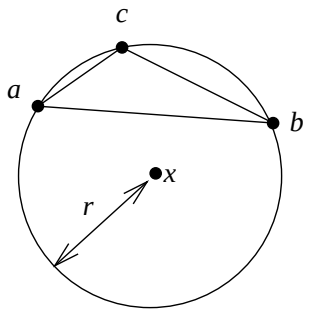

Figure 3: Case 2.A: The new point is the circumcenter of $T(a, b, c)$.

Furthermore, since t is a Delaunay triangle, the empty circumcircle criterion ensures that circle through a, b and c, of radius $r = \|cx\|$, does not contain any point of

the CDT. Applying lemma 1 to the previous expression, the length of new edges can be bounded:

$$R(a) < 2\|cx\| \Longrightarrow R(x) < 2\|cx\| + \|ax\| = 3r \Longrightarrow r > \frac{1}{3}R(x)$$

Case 2.B: Segment cx cuts the boundary polygon. In this case, the configuration is the one shown in Figure 4, where a boundary edge jk splits c from x.

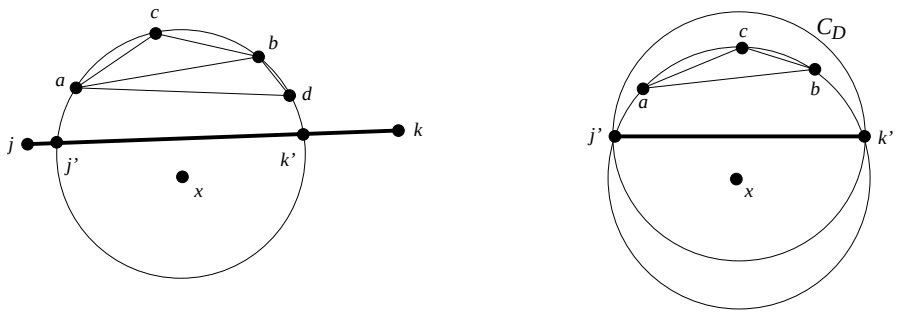

Figure 4: Case 2.B: Boundary edge jk splits c from x. *Left*, configuration; *right* $\|ab\| \leq \|j'k'\|$ and $T(a,b,c) \subset C_D$.

Since all boundary edges are admissible, $R(j) \geq \|jk\|$ or $R(k) \geq \|jk\|$. Let us suppose, without loss of generality, that $R(j) \geq \|jk\|$.

Segment $j'k'$ is defined as the part of jk inside the circumcircle of $T(a,b,c)$, and C_D is the diametral circle of $j'k'$ – *i.e.*, the circle with diameter $j'k'$ (see Figure 4). Remark that $C_D \subset C_R(j)$, because k is interior to $C_R(j)$. The following two facts will be proved:

$$\|ab\| \leq \|j'k'\| \quad \Longrightarrow \quad \|ab\| \leq \|jk\| \tag{2}$$

$$T(a,b,c) \subset C_D \quad \Longrightarrow \quad T(a,b,c) \subset C_R(j) \tag{3}$$

Since $j'k'$ is a secant of circumcircle of $T(a,b,c)$ that splits point x (the center of the circle) from c, angle $\angle j'xk'$ has to be lower than π (see Figure 4,right). As a consequence, the diametral circle C_D contains the portion of the circumcircle where c is included; specifically, it contains triangle $T(a,b,c)$. Furthermore, since ab is a secant of the arc cut by $j'k'$, segment ab has to be shorter than $j'k'$.

By equations 2 and 3 and due to the admissibility of segment jk,

$$\begin{aligned} R(j) \geq \|jk\| \Longrightarrow \forall p \in C_R(j),\ \Omega(p) \geq \|jk\| \Longrightarrow \\ \Longrightarrow \forall q \in T(a,b,c),\ \Omega(q) \geq \|jk\| \geq \|ab\| \end{aligned}$$

i.e., triangle $T(a,b,c)$ is admissible, although we cannot ensure this just evaluating function R at its vertices. Therefore, the refining edges algorithm can remove the edge from the list without inserting a new point in the CDT.

2.3 A bound for the number of points

Thanks to the way the edge refinement works and using the fact that function R is Lipschitz continuous, we can bound the number of new vertices inserted on a face by the refining edges algorithm.

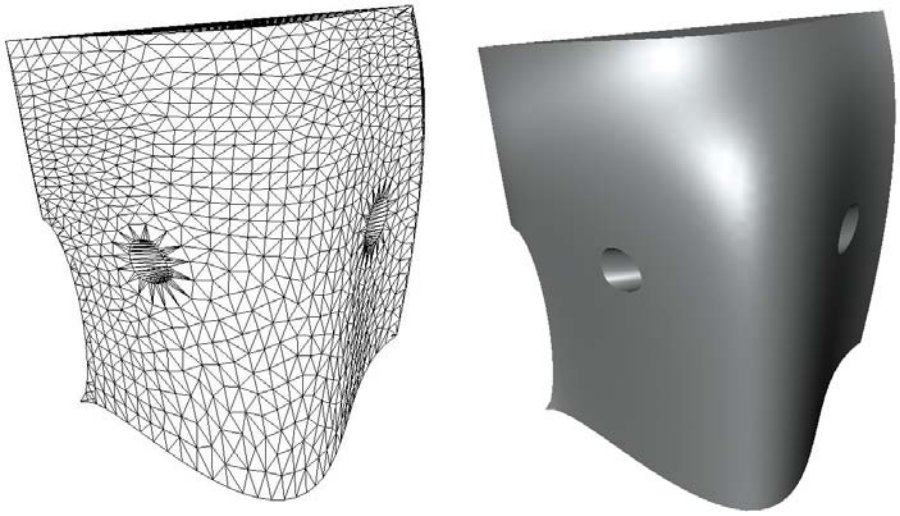

Figure 5: Surface and triangulation obtained applying the first method.

Theorem 1. *The number of points in the output triangulation is at most*

$$\frac{18^2}{\pi} \int_B \frac{1}{R^2(x)}\, dx$$

where B is the area of the bounding box of the face.

For the interested reader, the proof of this theorem can be found in [18]. The proof follows a similar technique to that of [13] (we have a CDT where new edges are created in such a way that its length is lower bounded; this bound depends on a Lipschitz continuous function R, thus new points are at a minimum distance from the old ones, that is, we can bound the number of new vertices by integrating over the entire region being triangulated).

It should be noticed that this is only a theoretical bound that demonstrates that the refining edges algorithm converges in a finite number of iterations. Since the algorithm is presented giving freedom to which non-admissible edge is chosen to be refined, in practice efficiency can be improved by imposing an order to the way edges are refined. We have tested two artificial orders: longest non-admissible edge first, and "least admissible" edge first – i.e first refine the edge ab whose length most differs from $max\{R(a), R(b)\}$. Both orderings work well, resulting in a similar number of triangles. In any case, the number of points is much smaller than the one predicted by the theoretical limit.

3 The regular triangulation algorithm

In this section a new triangulation algorithm is presented. This algorithm follows the same scheme than the one presented in Section 2. The main novelty is the more intensive use of the R function and, in consequence, the more intensive use of the geometry

of the surface being triangulated. The R function was computed to determine the admissibility of any edge of a triangulation of the surface. This function gives information about the flatness of the surface in a neighborhood of any parametric point, but it was only used to determine the admissibility of the edges of the triangulation. The information supplied by this function could also be useful in the edge refinement step and in the triangulation algorithm.

Regular truiangulations give a basic tool to incorporate the geometric information in this process. Now, given a set of parametric points, the regular triangulation of these performed and the resulting triangles are mapped to the image space. Moreover, edge refinement is modified, using a weight attached to any parametric point as regular triangulations do. The new algorithm can be summarized as follows:

1. Retrieve and preprocess input data.
2. Analyze the surface curvature and compute the admissibility bounds.
3. Set vertices and discretize edges (trimming curves), using the R function to place vertices on the edges.
4. Attach a weight to any parametric point, and triangulate the interior of each face using the regular triangulation. Regular edge refinement adds new vertices in case of non-admissible edges.
5. Map triangles to image space.

In the rest of the section, only steps 3 and 4 of the algorithm, which are different from the previous one, are detailed. Before, a brew summary of regular triangulations is presented.

3.1 Regular triangulations

Regular triangulations are a generalization of Delaunay triangulation. In this case, a real valued weight ω_p is assigned to each point p of $S \subset R^d$. This weight can be interpreted as a sphere $\mathcal{C}_{\omega(p)}$ with center p and radius $\sqrt{\omega_p}$. Usually, as we do, positive weights are assumed, although there is no theoretical inconvenient in having negative weights and thus spheres with imaginary radius.

Regular triangulations can be defined as the dual graph of the Voronoi decomposition, in a graph-theoretical sense, obtained using the power distance, $\| \cdot \|_\omega$ [9]. The power distance from a point $q \in \mathbb{R}^d$ to a weighted point p is defined as:

$$\|pq\|_\omega = \|pq\|^2 - \omega_p$$

This Voronoi distance can be interpreted geometrically as the square of the length of a segment from q to the point r in $C_\omega(p)$. The Voronoi diagram of a set of weighted points defined by the power distance is called the power Voronoi diagram. *Redundant points* are the ones whose Voronoi region is empty, that is, points $p \in S$ such that any point q of R^d is closer to another point $p' \in S$, different form p, using the power distance. Redundant points are not vertices of the resulting triangulation.

3.2 Weights and contour discretization

The new algorithm uses weights attached to the vertices that depend on the R function. The weight associated to a point p is chosen as $R(p)/k$, where k is a positive integer constant. This definition assures that geometric information is incorporate in

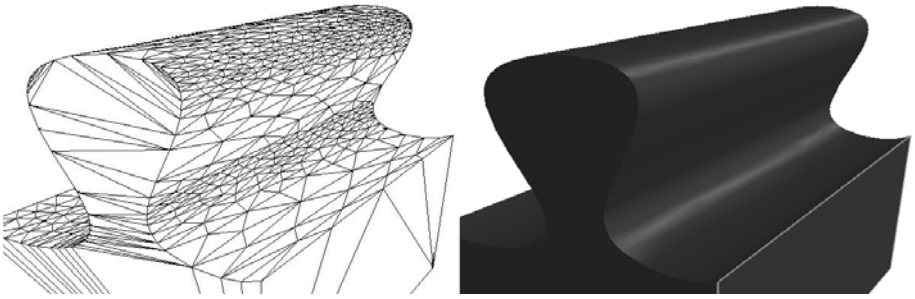

Figure 6: Triangulation using the second algorithm.

the attached weight. Moreover, a large enough k value allows to prove that no redundant point will appear during the regular edge refinement. Experimental results show that $k = 4$ is a sufficient value to avoid redundant points. notice that this value is the one required to prove Theorem 1, which is based on the condition that circles $C_{R/4}(p)$ do not contain other triangulation vertices apart from p. In a regular triangulation, a necessary condition for a vertex to be redundant is that its weight circle C_ω must be contained in the interior of the weight circles of the other vertices.

The contour edges are discetized using the weights attached to the vertices. The main difference with previous algorithm is that now contour edges are not bisected; instead, new points are located at a distance proportional to the weights of the edge endpoints.

3.3 Regular Edge Refinement

In this section, a new edge refinement method is proposed which uses the geometric information given by the R function. The present edge refinement method differs on the previous one on the computation of the location of a new vertex refining a fixed edge, which, in the current case, is a non admissible edge. In fact, the *regular edge refinement* locates new vertices using a weighted Voronoi decomposition.

When a non admissible edge is selected, two possible configurations of adjacent triangles are distinguished: in the first, the non admissible edge cuts its dual of the weighted Voronoi diagram, whereas in the second, the edge is not cut by its dual. These cases are analogous to the previous edge refinement method. However, now the Voronoi diagram is computed with the weights depending on the R function as explained above.

In case 1, when the non admissible edge ab cuts its dual in the weighted Voronoi diagram, the point to insert in the triangulation is the intersection point x between the edge ab and its dual. Now this point is not in the middle of the edge; in fact, it is closer to the endpoint of the segment with smaller weight (in the same way that happens when contour edges are discretized).

Case 2 occurs when the current non admissible edge ab and its dual do not intersect. In this case, among the weighted Voronoi points corresponding to the two adjacent triangles to ab, $T(a, b, c)$ and $T(a, b, d)$, the one which is closer to ab is inserted in the triangulation (see Figure 2).

A drawback for applying regular triangulations to our edge refinement scheme is that they cannot be restricted – as far as we know, no author has tried to define them.

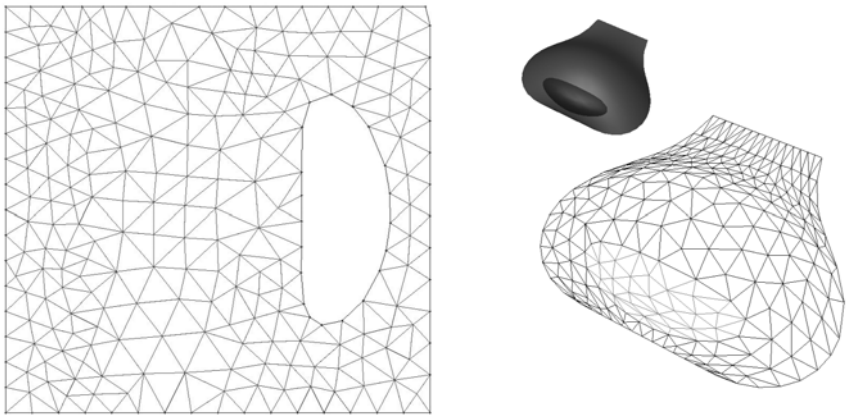

Figure 7: Adaptiveness of the mesh to the curvature of the surface. *Left* Parametric space; *right* image space (surface and approximation).

The existence of redundant points difficults the inclusion of restricting edges in a regular triangulation (what happens when one of the endpoints of the restricting edges is redundant?). Although no vertices are redundant in our triangulations, we still had problems when applying our regular edge refinement algorithm to some patches with holes or concave boundaries.

4 Results

In this section we present some examples showing the output of the two algorithms, the non-regular and the regular. We also compare them in terms of the number of vertices and adaptiveness.

The capacity of the first method to be adaptive to the local curvature of the original surface can be verified on Figure 5, which shows the triangulation of an object both in parametric an image space. Notice the triangle size variation on the different zones of the larger patch. The example also shows the ability to deal with a set of neighboring trimmed patches without producing cracks. Figure 7 shows another example of application of the first method to an object composed of several trimmed parametric patches.

Figure 8 left shows a more notorious example of the adaptiveness of the triangulation (compare triangle sizes on left middle and right of the patch). In this case, the second algorithm was applied (i.e., the regular triangulation was used). The triangulation on the right of the figure is an example where a small tolerance value was required. The triangulation in Figure 6 was also obtained by the second approach.

Finally, we made some tests to compare the two proposed algorithms. As expected, we found that the number of triangles when applying the second algorithm was smaller. However, this improvement was more noticeable in some test objects than in others. In fact, we can conclude that the second approach was slightly better than the first one for surfaces with a strong curvature variation, obtaining up to 25% less triangles. Figure 9 shows two output triangulations of a very curved surface. Note that although both

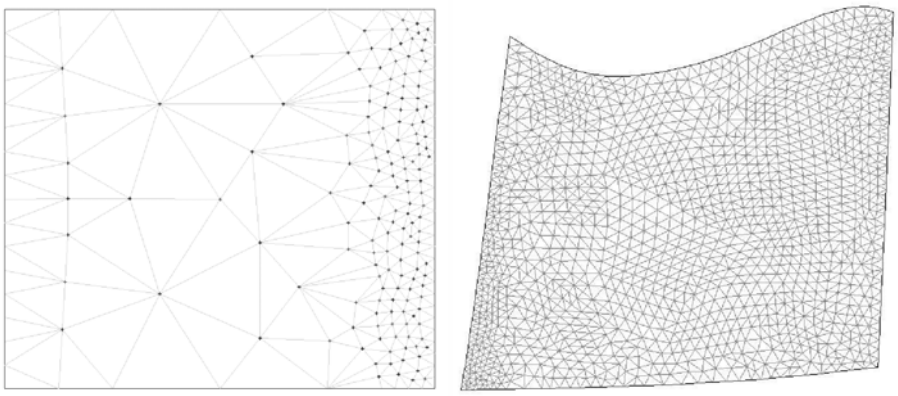

Figure 8: *Left,* output triangulation built by the second approach for a patch with strong curvature variation. *Right* A mesh obtained imposing a small tolerance.

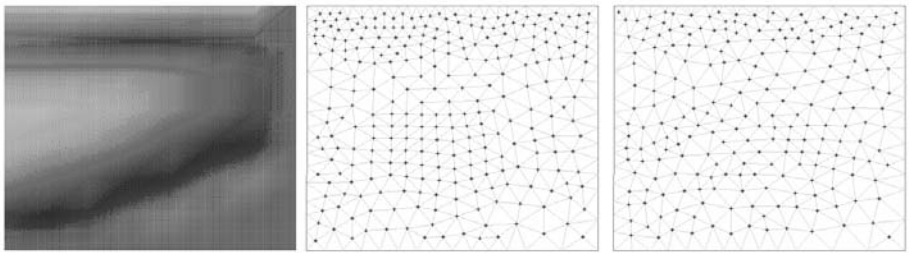

Figure 9: Comparison between the first and the second approach. *Left*, curvature plot; *middle* output of the first algorithm; *right*, output of the second algorithm.

triangulations are admissible and adaptive, the second algorithm places fewer points in the interior (and thus, it produces less triangles). Notice also that the triangulation obtained by the second algorithm looks less structured (triangle shapes are more variable). This is because the refinement algorithm in the second approach does not tend to place internal vertices exactly in the midpoint of edges, since bisectors in a power Voronoi diagram are not placed at the same Euclidean distance from the points.

5 Conclusions and future work

We have presented two adaptive methods of surface triangulation, based on edge refinements. The first method uses curvature bounds to determine the admissibility of a triangulation edge while the second one uses these bounds also in the edge refinement and in the posterior re-triangulation processes.

Concerning the second method, we have decided to use regular triangulations because they allow to consider geometric properties of the surface by assigning weights to the points and we have used the presented function $R(p)$ as a weight. We have de-

vised a generalized edge refinement method and implemented an incremental regular triangulation algorithm. Comparing both methods presented we can conclude that the regular method behaves better than the first one, since it approximates the surfaces with a lower number of elements, specially for patches with high curvature variations.

Nevertheless, there still remain several open problems. The first one is related with regular triangulations and restricting edges. We only have tested the regular method with simple patches (most of the not trimmed), so as a future work we need to devise an algorithm for constrained regular triangulations. We also want to study whether there is another function better than $R(p)$ that could also avoid redundant points and to obtain a theoretical bound of the number of points for the regular triangulation.

References

[1] P. Brunet and M. Vigo. Piecewise linear approximaton of trimmed surfaces. In G. Farin H. Hagen and H. Noltemeier, editors, *Geometric Modelling*, Computing Suppl. 10, pages 341–356. Springer Verlag, 1995.

[2] L.P. Chew. Guaranteed quality mesh generation for curved surfaces. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 274–280, 1993.

[3] Wonjoon Cho, Takashi Maekawa, Nicholas M. Patrikalakis, and Jaime Peraire. Topologically reliable approximation of trimmed polynomial surface patches. *Graphical Models and Image Processing*, 61(2):84–109, March 1999.

[4] J.C. Cuillère. An adaptive method for the automatic triangulation of 3D parametric surfaces. *Computer Aided Design*, 30(2):139–149, 1998.

[5] D. Filip, R. Magedson, and R. Markot. Surface algorithm using bounds on derivatives. *Computer Aided Geometric Design*, 3:295–311, 1986.

[6] S. Kumar and D. Manocha. Efficient rendering of trimmed NURBS surfaces. *Computer Aided Design*, 27(7):509–521, 1995.

[7] Xiang-Yang Li, Shang-Hua Teng, and Alper Üngör. Biting: Advancing front meets sphere packing. *the International Journal of Numerical Methods in Engineering (IJNME)*, 1999.

[8] D. Marcheix and S. Gueorguieva. Nibble meshing: incremental triangulation of non-manifold solid boundary. *Computers & Graphics*, 22(2-3):181–188, 1998.

[9] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations. Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons Ltd, 1992.

[10] L.A. Piegl and A.M. Richard. Tesselating trimmed NURBS surfaces. *Computer Aided Design*, 27(1):16–26, 1995.

[11] L.A. Piegl and W. Tillert. Geometry-based triangualtion of trimmed NURBS surfaces. *Computer Aided Design*, 30(1):11–18, 1998.

[12] A. Rockwood, K. Heaton, and T. Davis. Real-time rendering of trimmed surfaces. *Computer Graphics*, 23(3):107–116, 1989.

[13] J. Ruppert. A new and simple algorithm for 2-dimensional mesh generation. Technical report, Computer Science Division, Univ. of California at Berkeley, 1992.

[14] X. Sheng and B.E. Hirsh. Triangulation of trimmed surfaces in parametric space. *Computer Aided Design*, 24(8):437–444, August 1992.

[15] K. Shimada. Anisotropic triangular meshing of parametric surfaces via close packing of ellipsoidal bubbles, 1996.

[16] K. Shimada and D.C. Gossard. Automatic triangular mesh generation of trimmed parametric surfaces for finite element analysis. *Computer Aided Geometric Design*, 15:199–222, 1998.

[17] M. Vigo. An improved incremental algorithm for constructing restricted Delaunay triangulations. *Comput. & Graphics*, 21(2):215–223, 1997.

[18] M. Vigo. *Aproximació facetada de superfícies paramètriques retallades*. PhD thesis, Universitat Politècnia de Catalunya, Nov. 1998.

[19] M. Vigo, N. Pla, and P. Brunet. From degenerate patches to triangular and trimmed patches. In A. Le Mehaute and A.L. Allgower, editors, *Curves and Surfaces*, 1997.

[20] M. Vigo, N. Pla, and P. Brunet. Directional adaptive surface triangulation. *Computer Aided Geometric Design*, 16:107–126, 1999.

[21] M. Vigo, N. Pla, and P. Brunet. Curvature adaptive triangulations of surfaces. In *ECCOMAS'2000*, Sept. 2000.