Curvature Adaptive Triangulations of Surfaces

Marc Vigo, Núria Pla, Pere Brunet
Secció d'Informàtica Gràfica
Dept. Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Abstract

Many application areas in CAD/CAM are concerned with triangulation of surfaces, and require an error bounded approximation of a parametric surface model. In this paper we present an overview of the authors results on surface meshing. Two methods are described for triangulating objects bounded by trimmed parametric surfaces. The algorithms approximate the object with a predefined tolerance, and are adaptive to the surface curvature. The main difference between the two algorithms lies on the way that adaption is achieved. While the first method takes into account isotropic bounds of the surface curvature associated to each parametric point, the second algorithm uses bounds that do depend on the curvature direction. Emphasis is made on the computation of the curvature-based bounds and on the triangulation of the interior of the faces. Examples are presented, and results are analyzed comparing the two algorithms. Finally, open problems are enumerated.

Keywords: Parametric surfaces, trimming curves, curvature adaption, Delaunay triangulations.

1 Introduction

Many application areas in CAD/CAM are concerned with triangulation of surfaces. Surface triangulation is required for instance in surface rendering, rapid prototyping and finite element methods (FEM). In all these applications, the initial parametric surface model must be approximated by a discrete mesh of triangles. Moreover, many geometrical methods for modeling could be optimized if the input data are only triangles. Therefore, an efficient solution for dealing with surfaces lays on first converting them into an error bounded triangulation.

An important application is the visualization of a parametric surface. In this case, the speed of the involved algorithms is crucial. Since hardware can render polygons very quickly, the most spread solution consists on triangulating the surface and passing the results to the graphical library or hardware. Algorithms for this conversion also has to avoid cracks within neighboring patches or surfaces, since discontinuities would produce unpleasant visual effects.

Many other authors have addressed the problem of approximating CAD parametric surfaces for different Computer Graphics applications. Their works can be classified according to the approach taken, which in most cases depends on the main application area. Meshing methods for rendering usually treat single patches, producing quadrilateral or triangular elements that cover the patch. Trimming curves are approximated by line segments, and the resulting polygonals are connected to the surface mesh using different techniques. The main drawback of tesselating single patches independently is that cracks can occur in neighboring patches. While the majority of these render meshing approaches use global bounds for the whole patch that limit the size of the elements to be produced, some methods are adaptive to surface curvature, and some others take into account visualization criteria such as surface tangency, distance to observer and contour lines. In this category we include references [KM95, PR95, PT98, RHD89]. Other approaches use quadtree data structures to achieve adaption [DM90, HB87, KS95].

Another strategy consists on using advancing-front methods for obtaining an adaptive triangulation of the surface. The works that use these approaches usually come from the field of FEM, therefore one of their main objectives is the quality of the resulting mesh [Cui98, MG98].

Finally, another group of works use Delaunay-based approaches for triangulating CAD parametric surfaces. These works are concerned not only with curvature adaption but also with the shape of the resulting triangles and the minimization of the number of elements produced. These are usually general purpose triangulation algorithms (although the majority of recent meshing algorithms are also presented as general purpose). In this category we can include the proposals [Che93, SH92] and also the phisycally-based method of [SG98]. The algorithms presented in this paper also falls into this cathegory.

In this paper we present an overview of the authors results on surface meshing. Two methods are described for triangulating objects bounded by trimmed parametric surfaces. The algorithms approximate the object with a predefined tolerance, ε, and are adaptive to the surface curvature. The main difference between the two algorithms lies on the way that adaption is achieved. While the first method takes into account isotropic bounds of the surface curvature associated to each parametric point, the second algorithm uses bounds that do depend on the curvature direction.

The rest of the paper is organized as follows: Section 2 formalizes the problem and discusses the multiple objectives; the algorithmic scheme, valid for both proposals, is presented in Section 3. Section 4 describes more accurately the main steps of the isotropic algorithm, that is, questions related to curvature bounds and the triangulation of the interior of the faces, and Section 5 presents the directional version of the algorithm. In Section 6 some examples are presented and the results are analyzed. Finally, conclusions and open problems are discussed in Section 7. Readers can find a more detailed description of the algorithms in references [BV95, Vig97, Vig98, VN00, VPB99].

2 Triangulations of parametric surfaces

The problem we are facing is to obtain an approximation of a closed object bounded by a set of parametric patches by triangles. Therefore, it can be viewed as a conversion from an exact surface model to an approximate piecewise linear one. While the first representation model has the advantage of being more concise, the second one is in some sense less complex, because it is entirely made of simple elements (triangles). As stated in the introduction, the main application fields of this conversion process include visualization surfaces, geometric algorithms (collision detection, boolean operations, etc.), rapid prototyping and FEM meshing. Keeping this in mind, we can enumerate the ideal traits of the output triangulation:

1. The triangulation must be *admissible*, i.e., the distance between the triangulation and the original surface must not exceed a tolerance value, ε.
2. The triangulation must describe a closed object and must be *conformal*,

i.e., neighboring triangles must connect through whole edges. This requires taking special care when discretizing neighboring patches.

3. The number of resulting triangles – or, equivalently, the number of vertices – must be *minimal*.
4. Triangles have to be *well shaped*, that is, as equilateral as possible.

Since requirements 3 and 4 are contradictory — typically, a triangulation with few points will contain bad shaped triangles, while optimizing the shape of the triangles can imply obtaining a triangulation with a large amount of small triangles — a priority policy between these requirements must be established. In our case, since the main filed are to apply geometric algorithms and to visualize surfaces, we prefer minimizing the number of triangles, that is, the quality of the triangles is a secondary requirement.

The requirement to minimize of the number of triangles leads us to another important objective: the curvature adaptation. The input model is a set of trimmed parametric patches, which is a widely spread model in CAGD that allows to represent many surface features and geometries. It is thus reasonable to suppose that the patches and trimming curves can be of high degree, having flat regions as well as other regions of high curvature. Therefore, if an small amount of elements has to be obtained, triangulation must be *adaptive* to curvature, i.e., the size of the resulting triangles have to depend on the curvature of the original patches. Many small triangles will be placed in curved regions, and large triangles will cover planar regions.

3 Overview of the algorithms

Two different surface triangulation algorithms have been developed to achieve the objectives presented in the previous section. Both algortihms are based in the same scheme which can be divided in the following steps:

1. Retrieve and preprocess input data.
2. Analyze of surface curvature and computation of the admissibility bounds.
3. Approximate vertices and edges (trimming curves).
4. Triangulate the interior of each face.
5. Map triangles to image space.

The main difference between the two algorithms is on step (2). The first algorithm, called the *isotropic algorithm*, computes admissibility bounds based on the maximum curvature on each point of the surface. Instead, the second algorithm uses curvature bounds that also depend on the parametric directions, and thus it was called the *directional algorithm*. The two algorithms will be presented in the following sections.

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 (B-Rep). 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 [VPB97] or a linear reparameterization of the surfaces (see Section 6).

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. These bounds are used to limit the size of the triangles and edges that can be produced to ensure admissibility, thus they are the clue for the resulting triangulation to be curvature adaptive. The main difference between the two proposed algorithms lies in these limits for triangles and edges (see Sections 4.1 and 5). Once computed, the bounds are stored using a quadtree-like data structure. Similar of neighboring minimal regions are unified in such a way that leaf nodes of the quadtree correspond to a region of the surface homogeneous enough with respect to these values (or ellipses in the directional case). The limits stored in this quadtree allow us to test the admissibility of a given triangle in a simple way. A triangle can be tested for admissibility taking only into account the bounding values/ellipses assigned to its three vertices.

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 are triangulated. This technique, also followed by other meshing algorithms such as [SG95] and [Cui98], 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. 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. In the directional case, the constrained Delaunay method had to be modified according to the directionality of the bounds [VN00]. This step is accurately described for the isotropic algorithm in Section 4.2.

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.

4 The isotropic algorithm

4.1 Isotropic bounds

The objective of this section is to present conditions for easily testing the admissibility of a given triangle that approximates a piece of a surface. Remark that these conditions must be curvature adaptive. This is achieved by obtaining bounds that depend on the curvature on a parametric point of a patch. We start with some definitions and obtain the desired conditions. Proofs of properties and theorems presented in this section can be found in [Vig98].

A patch is a function $S(u,v)$ from the parametric space $\mathbb{D} \subset \mathbb{R}^2$ to image space $\mathbb{R}^3$. Let $T = (A, B, C)$ be a triangle in parametric space with vertices A, B and C, then function $l(u,v)$ is the planar triangle parameterized such that $l(A) = S(A)$, $l(B) = S(B)$ and $l(C) = S(C)$.

Let $S(u,v)$ be a parametric patch, $p \in \mathbb{D}$ a point and T a triangle parametric space, then the following notations will be used:

$$M_{uu}(p) = \left(\left.\frac{\partial^2 S_x}{\partial u^2}\right|_p, \left.\frac{\partial^2 S_y}{\partial u^2}\right|_p, \left.\frac{\partial^2 S_z}{\partial u^2}\right|_p \right),$$
$$M_{uv}(p) = \left(\left.\frac{\partial^2 S_x}{\partial uv}\right|_p, \left.\frac{\partial^2 S_y}{\partial uv}\right|_p, \left.\frac{\partial^2 S_z}{\partial uv}\right|_p \right),$$
$$M_{vv}(p) = \left(\left.\frac{\partial^2 S_x}{\partial v^2}\right|_p, \left.\frac{\partial^2 S_y}{\partial v^2}\right|_p, \left.\frac{\partial^2 S_z}{\partial v^2}\right|_p \right),$$

$$M_1 = \sup_{p \in T} \|M_{uu}(p)\|, \qquad M_2 = \sup_{p \in T} \|M_{uv}(p)\|, \qquad M_3 = \sup_{p \in T} \|M_{vv}(p)\|.$$

The following theorem bounds the error between a surface $S(u,v)$ and its linear approximation $l(u,v)$:

Theorem 1. *Given a parametric surface $S(u,v)$ and a triangle $T=(A,B,C)$ parameterized as $l(u,v)$, then*

$$\sup_{(u,v)\in T} \| S(u,v) - l(u,v) \| \leq \frac{2}{9}\Omega^2 (M_1 + 2M_2 + M_3)$$

being Ω the length of the longest edge of T in parametric space.

A proof of this theorem can be found in [SH92]. This result gives global bounds to limit the maximum edge length of any admissible triangle:

Corollari 1. *A triangle with maximum edge length Ω is admissible if*

$$\Omega \leq 3\sqrt{\frac{\varepsilon}{2M_1 + 4M_2 + 2M_3}}$$

However, since we are looking for a triangulation adaptive to curvature, instead of a global bound for the whole patch, we define the $\Omega(p)$ function for each point $p \in \mathbb{D}$:

Definition 1.

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

This allows us to obtain the following sufficient condition for a triangle T to be admissible:

$$\Omega(p) \geq \Omega \qquad \forall p \in T \tag{1}$$

being Ω the length of the longest edge of T. We define circle $C_\Omega(p)$ to be the circle centered on p with radius $\Omega(p)$. However, based on this condition, the admissibility of a triangle T would imply computing $\Omega(p)$ for all interior points of T. To simplify the admissibility test we need some additional definitions. A second function $R(p)$ is defined for each parametric point p:

Definition 2.

$$R(p) = \min_{q \in C_\Omega(p)} \{\max\{\|pq\|, \Omega(q)\}\}$$

and circles $C_R(p)$ are defined as a circle centered on p with radius $R(p)$. This leads us to the following theorem:

Theorem 2. *Let T be a triangle in $\mathbb{D}$ and let AB be its longest edge. Then, a sufficient condition for T to be admissible is that*

$$R(A) \geq \|AB\|.$$

This theorem allows us to test the admissibility of a triangle in a simple and fast way, only the value R must be evaluated at the vertices of its edges. Notice that the theorem does not depend on which endpoint of the edge AB is chosen.

The isotropic analysis consists on evaluating the function R on the patch. Since the analytic form of this function would be too complex to be evaluated, a discretization is computed. The patches are subdivided until it is found that the second derivative values, and therefore the Ω and R values, are homogeneous enough on that region or until a maximum subdivision level is reached. Finally, R values are stored in a quadtree, as it has been explained. This admits a fast way of locating $R(p)$ value for a point p on the patch.

4.2 Triangulation of faces

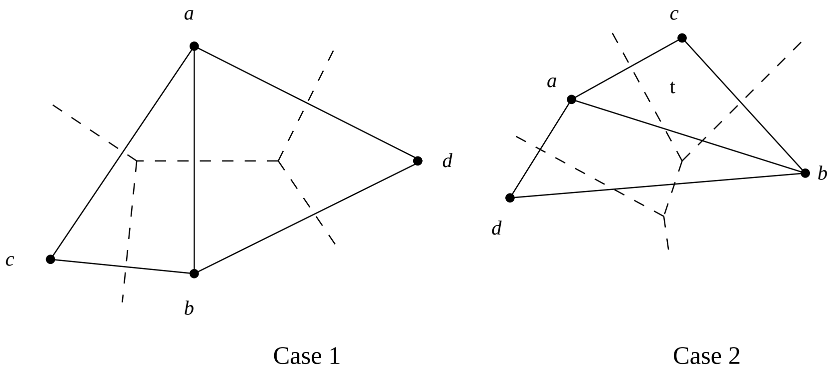

Figure 1: The two cases of the refining edge method, depending on the configuration of adjacent triangles. Voronoi diagrams are drawn using dashed lines.

We start from a solid whose edges have yet been discretized, approximating the trimming curves with admissible (straight) segments. Triangulating the faces thus means producing a triangulation of the interior of the polygonal region, 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 a triangle can be tested simply comparing the length of its edges with the R value at its vertices. The algorithm we developed for this purpose is called the refining edges method, and it follows a scheme similar to the algorithms of [Che93] and [Rup92]. 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 new non-admissible edges of CDT
    endIf
endWhile
```

When the refining edges method finishes, all the triangles will be admissible, and a reduced set of interior points will have been introduced. The usage of a Delaunay triangulation and the fact that points are added based on an empty circumcircle criterion ensures, as a sub-product, that the shape of the triangles will be reasonably good. Triangulation (CDT) of a planar graph was specially developed (see [Vig97]). This CDT algorithm works incrementally, allowing fast insertion of single points and edges in the CDT.

Whether an edge needs to be refined and the position of the refining point 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,

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

We have two possible configurations of adjacent triangles (see Figure 1): in the first case, the edge ab cuts its dual of the Voronoi diagram; in the second case, this 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 it can be proved that no additional point is needed.

5 The directional algorithm

The bounds Ω and R computed in the previous section are not directional. In order to add directionality to the bounds, instead of dealing with single values we treat with elliptical regions. The computation of the elliptical bounds is based on a more accurate analisys of the results presented in Theorem 1 and Corollary 1.

Let T be a triangle in parametric space and let t be an interior point to T. The triangle is divided into three regions by connecting its center of gravity with the midpoints of all three edges (see FIgure 2), each region corresponds to a vertex of T. Then, the *site vertex* of t is the vertex p of T corresponding to the region where t lies.

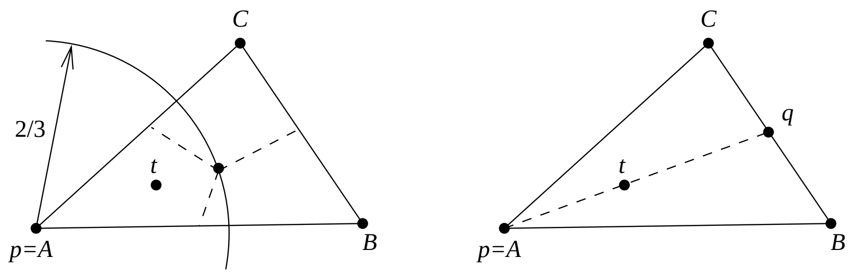

Figure 2: Illustration for definition of site vertex and for Theorem 3.

Using this notation, the following results are obtained. They give more accurate bounds than the ones obtained in the isotropic case and include directionality.

Theorem 3. *Given a C^2 parametric patch $S(u,v)$, a triangle T in parametric space, and $l_T(u,v)$ as before, then the maximum distance between $S(u,v)$ and $l_T(u,v)$ satisfies*

$$\sup_{\{u,v\}\in T} \|S(u,v) - l_T(u,v)\| \leq \frac{2}{9} \sup_{t\in T} \|(\overrightarrow{pq}^T Q_x(t)\, \overrightarrow{pq},\ \overrightarrow{pq}^T Q_y(t)\, \overrightarrow{pq},\ \overrightarrow{pq}^T Q_z(t)\, \overrightarrow{pq})\|$$

where $t = (u,v)$, p the site vertex of t, and q is the intersection point between the edge of T opposite to p and the line through t and p (see Figure 2).

Corollari 2. *A triangle T is admissible if*

$$\sup_{t\in T} \|(\overrightarrow{pq}^T Q_x(t)\, \overrightarrow{pq},\ \overrightarrow{pq}^T Q_y(t)\, \overrightarrow{pq},\ \overrightarrow{pq}^T Q_z(t)\, \overrightarrow{pq})\| \leq \frac{9}{2}\varepsilon$$

being p, q and t as in Theorem 3.

This result allows us to define a function $B(t, \overrightarrow{v})$ which gives a sufficient condition for a parametric triangle T in the parametric space to be admissible.

$$B(t, \overrightarrow{v}) = \sqrt{\frac{9}{2}\varepsilon}\ [\ (\overrightarrow{v}^T Q_x(t) \overrightarrow{v})^2 + (\overrightarrow{v}^T Q_y(t) \overrightarrow{v})^2 + (\overrightarrow{v}^T Q_z(t) \overrightarrow{v})^2\ ]^{-\frac{1}{4}}$$

Thus, Theorem 3 can be reformulated in a different way:

Theorem 4. *A sufficient condition for a triangle T in parametric space to be admissible is that*

$$\forall t \in T \qquad B(t, \overrightarrow{v})\ \geq\ \|\overrightarrow{pq}\|$$

being p the site vertex of t, q the intersection point between the edge of T opposite to p and the line through t and p, and $\overrightarrow{v} = \overrightarrow{pq}/\|\overrightarrow{pq}\|$.

For any point p, we define a region $B(p)$ centered on the point p which contains the points q such that:

$$B(p, \frac{\overrightarrow{pq}}{\|\overrightarrow{pq}\|}) \geq \|\overrightarrow{pq}\|$$

The region $B(p)$ has the drawback of not having a simple shape, therefore $B(p)$ is simplify by an ellipse $E_\Omega(p)$, which is defined as the ellipse centered in p contained inside $B(p)$ such that it contains the circle $C_\Omega(p)$ and has maximum area.

The next step is to obtain ellipses $E_R(p)$, in order to test the admissibility of triangles using a simple condition as in the isotropic case. Definition of ellipses E_R from E_Ω is done generalizing the definition of function R from Ω, dealing with elliptical regions instead of single values :

Definition 3 (E_R).

$$E_R(p) = \min_{q \in E_\Omega(p)} \{\max\{Red(E_\Omega(p), q)\ ,\ E_\Omega(q, p)\}\}$$

Where $E_\Omega(q, p)$ is the ellipse centered in p obtained by translation of $E_\Omega(q)$, and $Red(E, q)$ is the ellipse interior to the given ellipse E and that has the same axes as that of E such that the point q is on its boundary and such that it has maximum area.

Using this definition, a straightforward way to test admissibility to a triangle using its boundaries is obtained. This result also includes directionality.

Theorem 5. *A triangle T with vertices (A, B, C) will be admissible if*

$$T \subset E_R(A) \cap E_R(B) \cap E_R(C)\ .$$

As in the isotropic case, a discretization of $E_R(p)$ function, which assigns an ellipse to each parametric point, is evaluated. The E_R ellipses are also stored in a quadtree-like data structure, using the same reasoning than in the isotropic algorithm. The rest of the algorithm to obtain a triangulation of a parametric patch follows the scheme given in Section 3; this means taking into account the new bounds and the different way to test the admissibility of triangles, and using the CDT algorithm adapted to admit elliptical information presented in [VN00]. Although the refining edges algorithm inserts points taking into account the elliptical bounds, if a classical Delaunay triangulation is used, edges may not always be oriented according to minimal curvature directions. This is the reason why the modified CDT algorithm has to be used.

6 Results

Figures 3-5 show examples of triangulations obtained using the algorithms we developed. The object of Figure 3 is a block cut by a Bézier patch of order 5×8, which presents a strong variation of the surface curvature. The result shows the adaptability of the size of triangles and edges to the curvature, even though the approximation is made of a quite small amount of triangles (only 262 triangles for the whole object). Edge lengths of triangles approximating the upper face vary between 12.2 and 61.5. Notice also that the shape of triangles of this face is quite good, thanks to the fact that the relaxation postprocess was applied.

The object in Figure 4 includes patches with strong curvature variations along different directions, thus it is an example where the directional algorithm can achieve good results. Patches have been drawn using different colors to better appreciate the results. It can be seen that the triangulation is adaptive not only to curvature magnitude but also to curvature directions. Triangle size depends on the region of the patches and edges tend to be aligned according to minimal curvature directions.

Figure 5 shows an object triangulated using the isotropic and the directional algorithm. The input solid is the result of a boolean operation between a block, a 5×6 order patch and a cylindrical-like part. Both output triangulations are composed of about 300 triangles, therefore a smaller tolerance value had to be used for the directional algorithm. The more noticeable difference that can be observed is the way the cylindrical part is triangulated: while

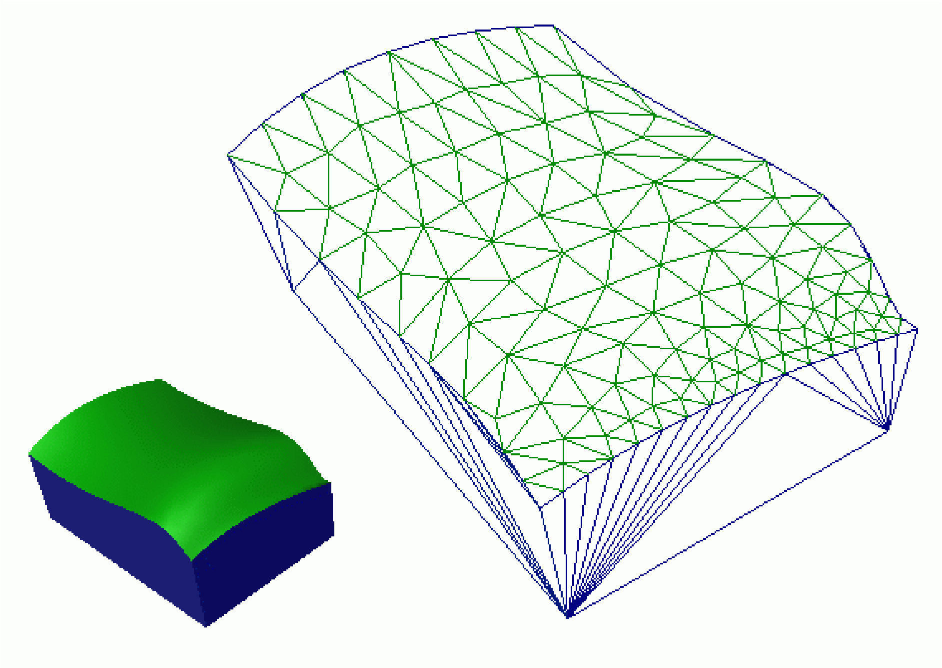

Figure 3: A triangulation of an object obtained using the isotropic algorithm.

the isotropic algorithm produced many quite well-shaped triangles, the directional version was able to approximate this surface using few long triangles oriented according to principal curvature directions. This example illustrates the behavior of the two algorithms: since the isotropic curvature bounds are less accurate than the directional ones, the isotropic algorithm needs more triangles to approximate a surface, specially if the curvature highly depends on the direction.

Although the shape of the triangles produced by the isotropic algorithm can be of a better quality, the triangulation obtained by the directional version is a better approximation of the original surface in a different sense. In fact, if tangency error is measured, the triangle normals output by the directional algorithm will be closer to the surface than the ones produced by the isotropic version. Tangency error (or normal error) is defined as the maximum error between the normal vector to the triangle and the normal vectors to the surface at corresponding parametric points. This is illustrated in Figure 6.

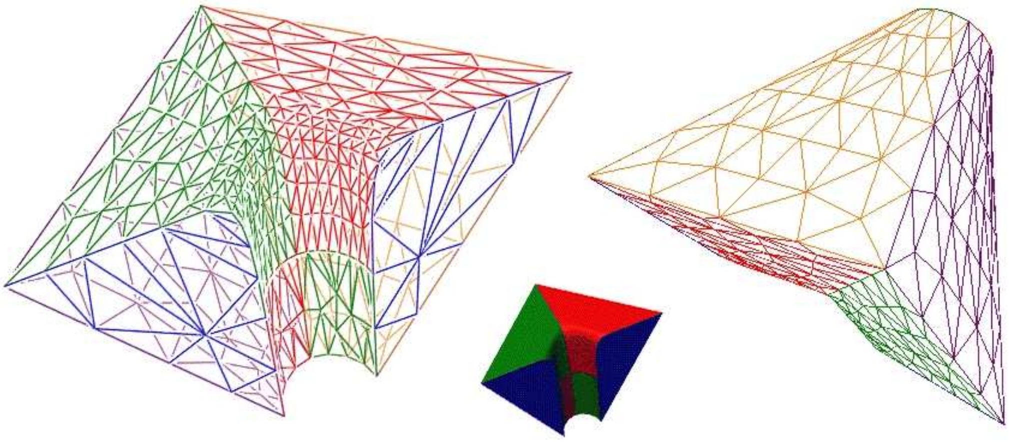

Figure 4: Two views of an object triangulated using the directional algorithm. Each patch is drawn using a different color.

An important problem we found when triangulating objects is that the way that the surfaces are parameterized can have a drastic effect on the resulting triangulations. Since the curvature bounds are computed on parametric space and the edge refinement is also performed in this space, the size of the triangles in image space depends on the parameterization. Therefore, a good parameterization of a patch is an isometrical one, that is, a parameterization that maps equilateral triangles in $\mathbb{D}$ to equilateral triangles in $\mathbb{R}^3$. The optimal case will be an arc-length parameterization of the surfaces. To solve this problem, an optional preprocessing step can be performed, consisting on a linear reparameterization of the patches. Although a linear transformation does not solve all the cases, this option was chosen because a general reparameterization could prohibitively raise the degree of the patches.

Nevertheless, we noticed that the directional algorithm is less affected by the way patches are parameterized than the isotropic. This can be easily explained because directional bounds include more information about the metric of the surface than the isotropic bounds. For example, the directional algorithm will approximate tensor-product surfaces produced by sweeping a curve along a vector regardless of its parameterization on the vector direction. This is a very common type of patches in CAD systems, but one can not expect that surfaces will be parameterized in a different way after a non-uniform scaling operation.

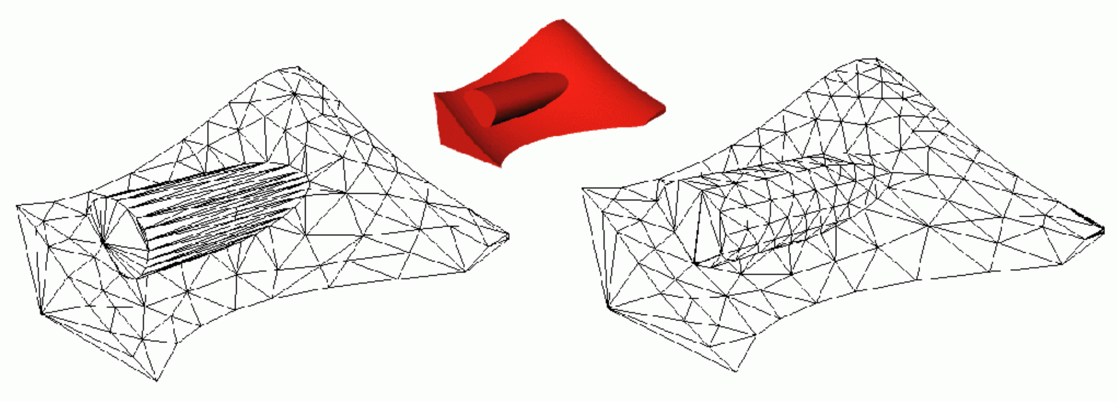

Figure 5: Two approximations of the same object with similar number of triangles. The object on the left was obtained using the directional algorithm. On the right, the output of the isotropic algorithm.

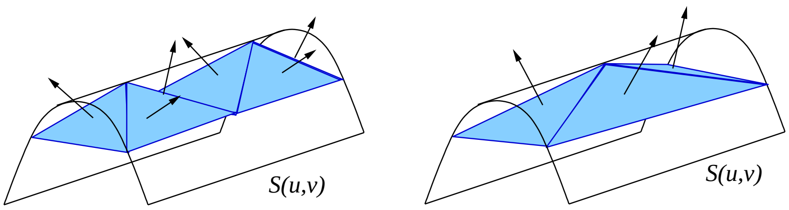

Figure 6: Tangency error between a piece of a surface and two approximations using four and two triangles. Left case corresponds to the isotropic algorithm, right figure corresponds to the directional. Although maximum approximation error is the same in both cases, tangency error is more accurate on the right case.

7 Conclusions and open problems

We have presented two algorithms for approximating the surface of a closed object by a mesh of triangles with a given tolerance. Both algorithms use a refining edges method based on a constrained Delaunay triangulation and are adaptive to the curvature of the patches. The first algorithm, the isotropic algorithm, uses bounds for the surface curvature that limit the maximum length of the edges to be created along any parametric direction. The second algorithm, the directional algorithm, is based on tighter curvature bounds that depend on the parametric direction, limiting the edge length according to its alignment.

Results show that the directional algorithm produces a mesh composed of less triangles than the isotropic, but in the second case the triangles can be better shaped. The triangulation output by the directional method in some way is also better because it approximates better the tangency error.

Since the triangulations work in parametric space, the output can vary depending on the parameterization of the original patches. This has less effect in the directional case, thanks to that elliptical bounds include more geometrical information about the surfaces. Nevertheless, a future research line would be to develop a method to correct this better than the linear reparameterization preprocess. An algorithm for reparameterizing the original surfaces in terms of arc-length would minimize this effect.

Notice that although computing the curvature bounds is a rather complex step that can consume quite a lot of time, if several approximations of the same object are to be produced this analysis would have to be performed only once, because it is a preprocessing step that can be parameterized on the tolerance error, ε. Therefore, producing a multiresolution set of a solid requires analyzing only once the curvature of the patches.

Another unpleasant effect of the developed methods, specially if FEM analysis is to be applied to the mesh, is that in planar (or almost planar) patches may be approximated by long skinny triangles if neighbor patches have strong curvature (see for example vertical faces of solid in Figure 3). This is consequence of giving priority to the number of points. To correct this, a post-processing could be performed; for example an adaption of the algorithm in [Rup92] would be a good option.

Finally, another future development would be to design a global algorithm able to manage sets of G^1 or G^2-continuous patches instead of triangulating patches independently. In this way, the final mesh could contain triangles with vertices on neighbor patches, that is, points would not had to be placed on trimming curves that correspond to edges between smooth enough patches.

References

[BV95] 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.

[Che93] L.P. Chew. Guaranteed quality mesh generation for curved surfaces. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 274–280, 1993.

[Cui98] J.C. Cuillère. An adaptive method for the automatic triangulation of 3D parametric surfaces. *Computer Aided Design*, 30(2):139–149, 1998.

[DM90] Dolenc and Makela. Optimized triangulation of parametric surfaces. *Mathematics of Surfaces IV*, 1990.

[HB87] B. Von Herzen and A. Barr. Accurate triangulations of deformed, intersecting surfaces. *Computer Graphics*, 21(4):103–110, 1987.

[KM95] S. Kumar and D. Manocha. Efficient rendering of trimmed NURBS surfaces. *Computer Aided Design*, 27(7):509–521, 1995.

[KS95] R. Klein and W. Straßer. Large mesh generation from boundary models with parametric face representation. In *Proceedings of Solid Modeling'95*, pages 431–440, 1995.

[MG98] D. Marcheix and S. Gueorguieva. Nibble meshing: incremental triangulation of non-manifold solid boundary. *Computers & Graphics*, 22(2-3):181–188, 1998.

[PR95] L.A. Piegl and A.M. Richard. Tesselating trimmed NURBS surfaces. *Computer Aided Design*, 27(1):16–26, 1995.

[PT98] L.A. Piegl and W. Tillert. Geometry-based triangualtion of trimmed NURBS surfaces. *Computer Aided Design*, 30(1):11–18, 1998.

[RHD89] A. Rockwood, K. Heaton, and T. Davis. Real-time rendering of trimmed surfaces. *Computer Graphics*, 23(3):107–116, 1989.

[Rup92] J. Ruppert. A new and simple algorithm for 2-dimensional mesh generation. Technical report, Computer Science Division, Univ. of California at Berkeley, 1992.

[SG95] K. Shimada and D.C. Gossard. Bubble mesh: Automated triangular meshing of non-manifold geometry by sphere packing. In *Proceeds of Solid Modeling'95*, pages 409–419, 1995.

[SG98] 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.

[SH92] X. Sheng and B.E. Hirsh. Triangulation of trimmed surfaces in parametric space. *Computer Aided Design*, 24(8):437–444, August 1992.

[Vig97] M. Vigo. An improved incremental algorithm for constructing restricted Delaunay triangulations. *Comput. & Graphics*, 21(2):215–223, 1997.

[Vig98] M. Vigo. *Aproximació facetada de superfícies paramètriques retallades*. PhD thesis, Universitat Politècnia de Catalunya, Nov. 1998.

[VN00] M. Vigo and N.Pla. Computing directional constrained Delaunay triangulations. *Comput. & Graphics*, 24(2), 2000.

[VPB97] 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.

[VPB99] M. Vigo, N. Pla, and P. Brunet. Directional adaptive surface triangulation. *Computer Aided Geometric Design*, 16:107–126, 1999.