Normal Umbrella: A new primitive for triangulating parametric surfaces

Marco Attene

Rapporto Tecnico N. 13/2001
Genova, Ottobre 2001

Abstract

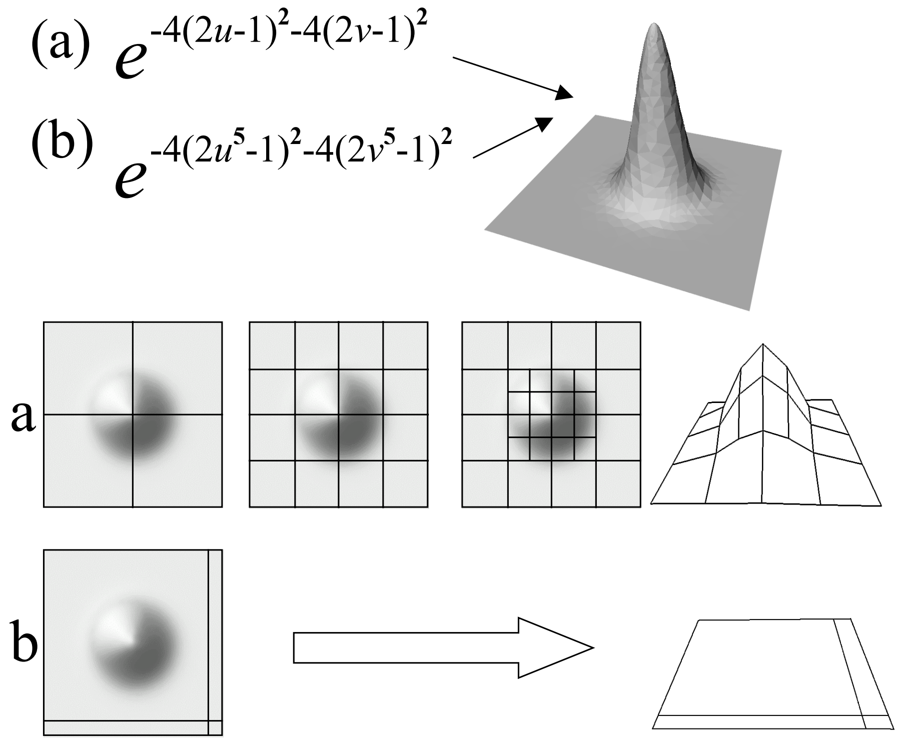
Typical methods for the triangulation of parametric surfaces use a sampling of the parameter space, and the wrong choice of parameterization can spoil a triangulation or even cause the algorithm to fail. We present a new method that uses a local tessellation primitive for almost-uniformly sampling and triangulating a surface, so that its parameterization becomes irrelevant. If sampling density or triangle shape has to be adaptive, the uniform mesh can be used either as an initial coarse mesh for a refinement process, or as a fine mesh to be reduced.

Introduction

The parametric form for the description of surfaces is widely used in geometric modeling. This moved the engineering community towards the study of techniques for the discretization, or tessellation, of such surfaces for visualization or for analysis purposes. Nowadays, the most used approaches are based on advancing front [1][2] or Delaunay triangulation [3][4][5], and there are *ad hoc* methods for specific surface classes [6][7][8]. It is worth to say that, while in cases such as surface reconstruction [9][10] the vertices are given, here the *tessellator* has to sample the surface and construct a mesh connecting the samples. Most of existing methods do both the steps in parameter space: first, they create a triangulation of the surface's 2D domain, then the final tessellation is obtained by mapping the vertices to 3D space, without changing the connectivity. A uniform sampling of the parameter space could be quite inappropriate, so adaptive methods [11][1] are preferred where an initial coarse mesh is iteratively refined depending on some error metrics. Adaptive methods, however, need a starting mesh and, if the surface is *well-behaved*, this can be achieved by a coarse uniform sampling of the parameter domain, otherwise, some problems may arise, as described in the next section. This paper proposes a method for sampling and triangulating the surface independently of the parameterization [12], so that bad mapping properties do not spoil the final triangulation.

Previous Work

As far as visualization is concerned, the first methods were based on ray tracing techniques [13][14]; here the main problem is efficiently calculating the first point of intersection of a straight half-line with the surface. In [15] a polygonization approach is presented by which this intersection point can be computed quite quickly. The tessellation of parametric surfaces, however, has been mainly developed for direct visualization [11] and for FEM applications [3]. As described in [16][17], scientific literature proposes approaches for generating either isotropic meshes, in which the element size and shape is roughly constant, or anisotropic ones, in which the sampling density varies depending on the surface curvature and triangles are stretched along principal curvature directions. The methods proposed for isotropic mesh generation are generally based on iterative repositioning of an initially random vertex set [18], or on physical principles of attraction/repulsion of the vertices until an equilibrium state is reached [3][19]. Unfortunately, due to their converging nature, these approaches lack a formal complexity estimate and, even more important, the number of samples must be fixed in advance, instead of depending on the surface area. An exception in this class of algorithms is the *marching method* presented in [20]; here the evaluation of lengths on the tangent plane can introduce significant errors in regions of high curvature, moreover the necessity of *numerical implicitization* for treating parametric surfaces can introduce a further error. Conversely, the creation of anisotropic meshes [21][11] is mainly based on adaptive schemes; in these cases the main drawback is the dependency on the surface definition. An example of such a problem is shown in figure 1: the same surface is defined in two different ways on the same 2D domain, and the adaptation criterion splits an edge in its middle 2D point if the image of this point is too far from the image of the edge, in the Euclidean 3D space. While in the first row (a) the initial coarse mesh has to be subdivided, it has not to be in the second case (b). On the other hand, if the initial coarse mesh were fine enough for *catching* the surface details, a huge number of redundant vertices would have to be created.



(a) $e^{-4(2u-1)^2-4(2v-1)^2}$
(b) $e^{-4(2u^5-1)^2-4(2v^5-1)^2}$
a
b

Figure 1: *The initial coarse mesh could not detect some surface details, preventing edge splits.*

In the following sections we discuss a novel approach that does not suffer these drawbacks. The remainder of the paper is organized as follows: some remarks on definitions and notation are presented, then the *normal umbrella* and its use as a tessellation primitive is explained and, finally, we give a brief description of the necessary extensions for the creation of closed and adaptive triangulations.

Parametric Surfaces

The image of a position vector-valued function $f : A \subseteq \Re^2 \rightarrow \Re^3$, ***f****(u,v)* = *<x(u,v), y(u,v), z(u,v)>* is called a *parametric surface*. The subset of $\Re^2$ from which the two parameters *u* and *v* take their values, usually the square [0,1]×[0,1], is called the *parameter domain*. The function ***f*** is called a parameterization, or mapping, and it serves only as a representation of the surface we are interested in. The same surface can be represented by several mappings, as shown for example in figure 2, and the application of a given tessellator can give different results depending only on the way the surface has been parameterized.

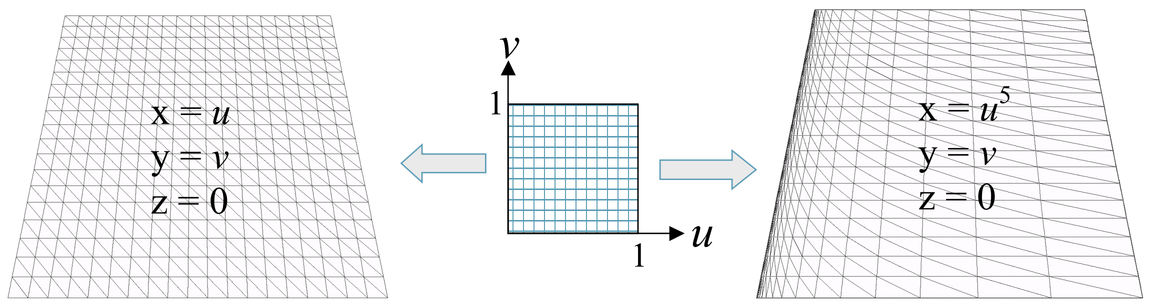


Figure 2: *The same plane can be defined in several different ways on the same domain.*

Since the tessellator should approximate the surface, and not its representation, such behavior is not a good characteristic. Worse than that, the *wrong* choice of parameterization for a surface could lead to quite bad results, as shown in figure 3.

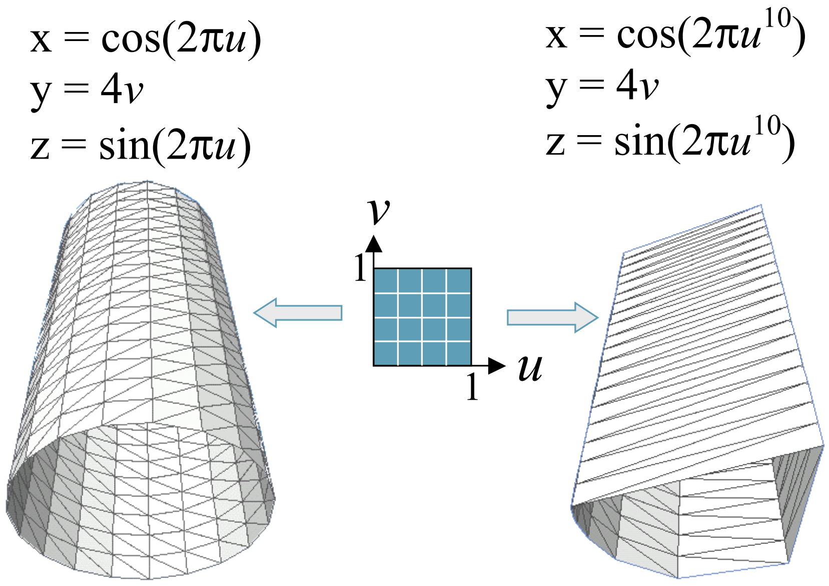


Figure 3: *The wrong choice of the parameterization could spoil the tessellation of the cylinder.*

Uniformly sampling a surface

The problem of generating an isotropic mesh independently of the parameterization can be approached from different points of view:

1. How to find a sampling that is as uniform as possible on the surface rather than on its domain ?
2. Is it possible to re-parameterize the surface so that a uniform grid in parameter space maps uniformly on the surface ?

Let us consider the second question in the case of plane curves. For a regularly parameterized curve ***c(t)*** = *<x(t), y(t)>*, the arc length between two points can be computed using the integration of the curve tangent vector. Starting from this, it is not difficult to derive a parametrization of the same curve based on the *curvilinear abscissa*. The represented curve is the same, but for the new mapping a uniform sampling of the parameter space maps uniformly on the curve. In [22] the problem has been approached in order to find a piecewise linear approximation of a plane curve in which each segment forms the same angle with the next one. In the same paper they attempted to define a reasonable extension for surfaces, but the author himself could not prove whether a solution exists or not. In the case we want to create an isotropic tessellation, it is not difficult to find a counterexample, as shown in figure 4.

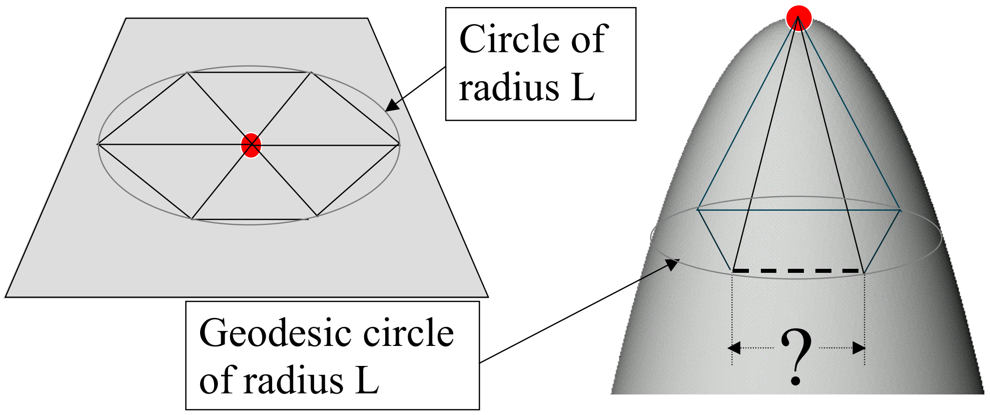


Figure 4: *An exactly uniform mesh does not always exist on curved surfaces.*

While in the plane it is always possible to create a closed fan of equilateral triangles around each point, that is, a triangulation in which all the edges have the same length, this can not always be achieved for generic 2-manifolds.

The Normal Umbrella

In the previous section it has been shown that, for a generic surface, an approximating mesh that is perfectly uniform may not exist. It must be considered that for most applications a good approximation is absolutely enough, for this reason we are going to define a local quasi-isotropic approximation called the *Normal Umbrella, or NU*.
Given a point p_p in the parameter domain whose image in the surface is p, let $H = \{p_1, p_2, .., p_6\}$ be a regular hexagon centered on p and lying on the tangent plane at p, Tp(p). For each p_i, draw a curve of arc length r on the surface from p to a point q_i so that its projection on Tp(p) is a straight line parallel to $\boldsymbol{v}_i = p_i - p$. Connect each end-point, q_i, with its neighbor, q_{i+1}. This process defines a 6th degree normal umbrella of radius r and center p which will be triangulated by the star $\{(p,q_1,q_2), (p,q_2,q_3), ..,(p,q_6,q_1)\}$. The result of this process is shown in object and parameter space in figure 5.

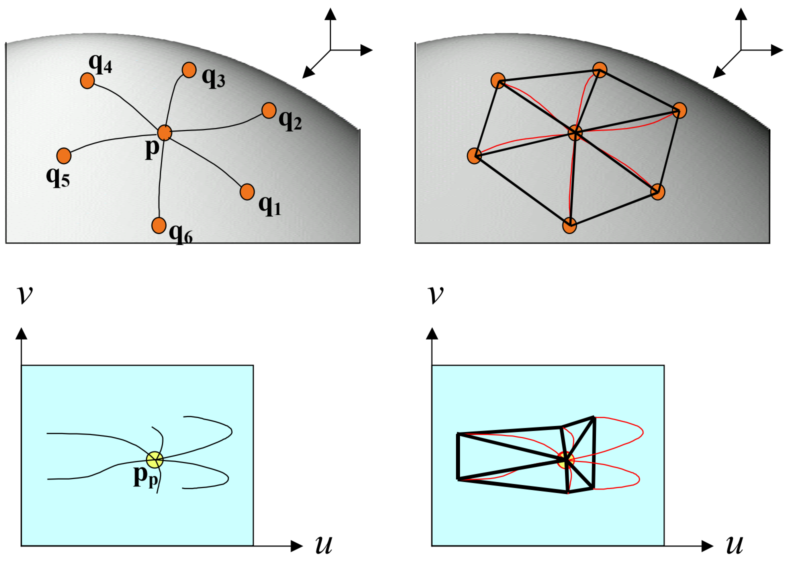


Figure 5: *A 6th degree normal umbrella and its counterimage in parameter space.*

Notice that the described process defines exactly a regular hexagon on the plane, but as the surface curvature increases the edges connecting the end points q_i - q_{i+1} become shorter. This behavior suggests that the number of paths to be tracked, n, should depend on the local curvature, specifically, high positive Gaussian curvature implies less than six paths, while high negative curvature implies $n>6$. Actually the curvature can change along the path, so we need a more accurate evaluation: consider all paths emanating from the point p whose projection on the tangent plane is a straight line and whose arc length is r. The end-points of these paths constitute a closed surface line[1] that we call a *Normal Umbrella- Front (NU-front)*. If the length of this line is L, then the optimal number of triangles n around the point p is the closest integer to L/r. In the case of a plane, for example, the NU-front is the circle of radius r, so its length L is $2\pi r$ and n = round(2π) = 6. The length of the NU-front of a paraboloid's tip is $\leq 2\pi r$, so $n \leq 6$ while in the case of a saddle point $L \geq 2\pi r$, so $n \geq 6$.

Computing the Normal Umbrella

The main difficulty in computing the normal umbrella comes from the fact that the definition is based on tracking surface paths and, since we can only play with parameters, tracking such paths is not a trivial operation. Fortunately, differential geometry [23] provides all the tools we need.

Normal Sections

The paths to be tracked for building the normal umbrella are nothing but parts of normal sections, which are defined for each point as the intersection of the surface and a normal plane.
Even if a normal section can be made of several connected components, we want to preserve the topology [24], so we are only interested in the one containing the generating point. By the means of differential calculus, it is not difficult to prove that the following system of differential equations represents the component we are interested in:

$$\begin{cases} u' = (n \times d) \cdot f_v(u,v) \\ v' = (d \times n) \cdot f_u(u,v) \\ u(0) = u_0 \\ v(0) = v_0 \end{cases}$$

where n is the normal of the generating point $f(u_0,v_0)$, d is a tangent unit vector belonging to the intersecting normal plane, and f_u and f_v are the two derivatives of the mapping function.
Now that we know how to track a path we need to calculate its initial direction d. First of all consider that the normal umbrella of a given radius at a point is not unique, in fact, there exist infinite regular hexagons[2] on the tangent plane; since we have no particular preference, the direction of the first path is arbitrary, while the others have to be computed according to this first one. Once a path has been tracked, its projection on the tangent plane is a straight segment, and the projection of each path must produce an angle of exactly 60 degrees with the previous one, so we know how to set up the differential equations.

The NU as a tessellation primitive

Now that the normal umbrella has been defined, let us see how to use it as a tessellation primitive. Given a parametric surface and a desired edge length, let us choose some point of the parameter domain and build an appropriate normal umbrella around its image. Now pick a vertex, v, of the boundary of the NU and complete its fan, treating the triangles meeting at v as if they were already part of a normal umbrella around v. Repeating the latter step until all the parameter domain is spanned and avoiding overlaps, gives a rough solution of our problem. Now let us see how to perform these steps in detail.

[1] Except for special cases generating singularities or self-intersecting paths.

[2] For simplicity we restrict the explanation to the 6th degree NU, but it holds for every degree ≥ 3.

Completion of a partial approximate NU

Except for the starting NU, each step must complete the fan around a vertex of the current boundary in the following way (with reference to figure 6.):

1. Compute the part of NU-front between the two boundary edges.
2. Divide it in equal parts so that the distance between the end-points of each part is as close as possible to the desired edge length.
3. Consider only the paths generating the selected end-points.
4. Connect these end-points by edges.

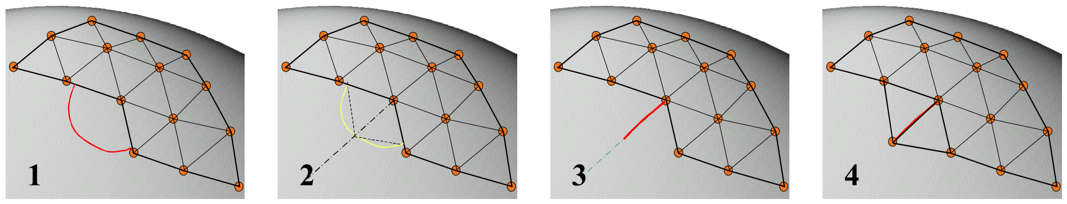


Figure 6: *The four steps for completing a partial approximate NU around a boundary vertex.*

The tiling algorithm

Let us consider a few more details in order to design a working algorithm. First of all, at each step a vertex must be processed and it must be chosen among the ones of the current boundary; it is not difficult to see that the order of the processing influences the final result; in particular, the insertion of some *short* edges (that is, thin triangles) may be required in some cases. By experiments we found out that choosing the vertex with the smallest angle to be triangulated drastically reduces such cases. So we sort the boundary vertices in a queue and, at each step, we pop the first vertex from it. A similar method has been used by Hartmann in [20]; here the algorithm performs a marching method on implicit surfaces and it uses the same sorting technique for choosing the boundary vertex to be triangulated. Putting everything together, the following is a sketch of the algorithm:

1. INPUT: Surface definition (mapping function), parameter domain (bounds or trimming curves) and desired edge length;
2. Choose a random point of the parameter domain and build the starting NU;
3. Build the queue containing the boundary vertices;
4. Pop the first vertex from the queue, complete its approximate NU and update the queue;
5. While the queue is not empty go to 4.

As we said in the section defining the NU, the number of paths to be tracked around the first vertex (that is, the degree of the NU) depends on the length of the NU-front and the radius. In our implementation we compute this length by linearly interpolating the end-points of a fixed number of normal sections around the vertex. Of course, the higher this number, the more precise is the result. The adaptive step-size Runge-Kutta method [25] is used to track the normal sections.

Intersection checks
In cases where the curvature is particularly high it may happen that the iterative expansion of the mesh makes the boundary intersect itself, so, before inserting a new edge, we must be sure that it does not intersect other parts of the boundary.

Complexity

We give the complexity as a function of the area of the surface to be triangulated. Let A be this area; the number of elements of the boundary is $O(A^{1/2})$. The number of triangles is linearly proportional to A, and the computation of each one of them requires $A^{1/2}$ intersection checks. The size of the queue to update is $A^{1/2}$, so the global complexity is $O(A^{1.5})$.

Closed surfaces

The basic algorithm, as it has been described, can only manage open surfaces with the same topology as their parameter domains. In order to build the tessellation of surfaces with different topology, we look for paths of (u,v) points that map on a unique 3D image (cutting line). For example, in the typical definition of the sphere by parallels and meridians, the two lines (0,0)-(0,1) and (1.0)-(1,1) of the parameter domain map on the same meridian (figure 7.).

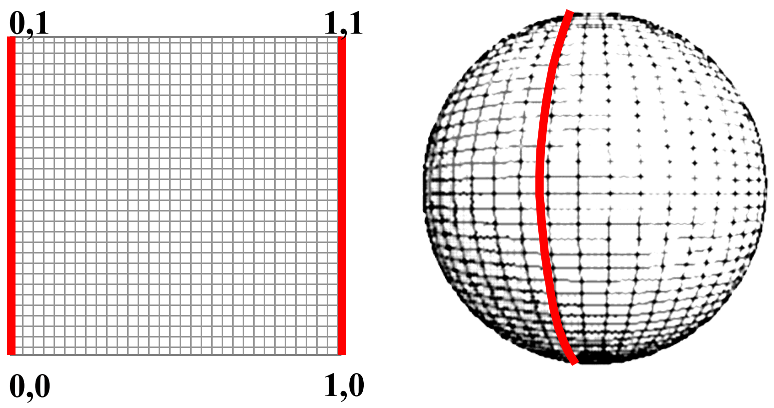


Figure 7: *Two edges of the parameter square map on the same line on the sphere.*

Our method detects overlapping boundary edges and, by splitting each one of them by one or both the end-points of the overlapping edge, establishes the new connectivity, as shown in figure 8.

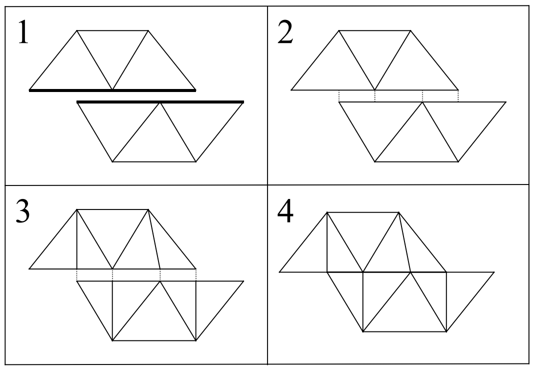


Figure 8: *Joining two parts of the boundary. In 1-3 the gap has been enhanced to show the lacking topological connection.*

By this simple extension the algorithm can tessellate every regularly parameterized C^1 2-manifold. In figure 9 the triangulation of a torus is shown.

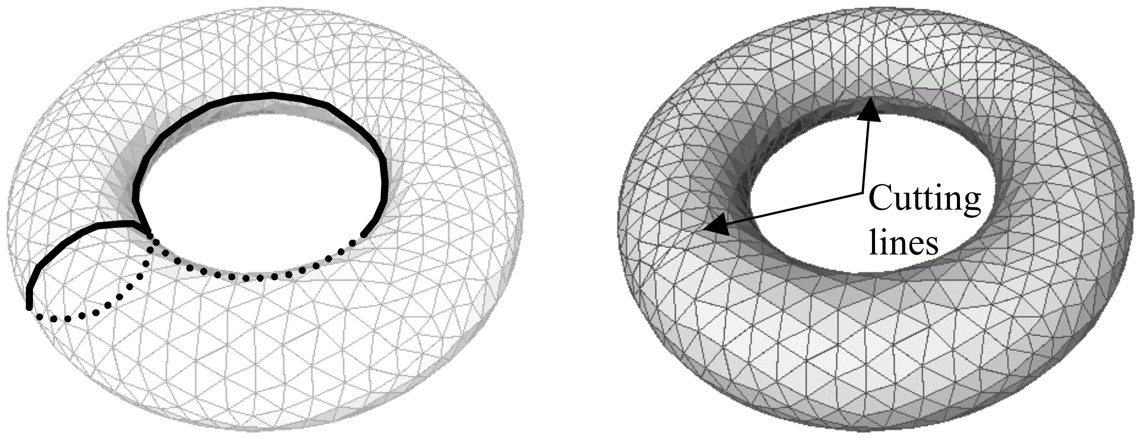


Figure 9: *Triangulation of a torus with corrected connectivity.*

Adaptivity

Now that we can almost uniformly tessellate a surface whatever its parameterization, we can develop a scheme to refine the initial coarse triangulation iteratively. The user chooses the expected edge-length, representing the dimension of the smallest feature that must be detected, then he sets the maximum acceptable distance, ε, of an edge from the surface. Once the initial mesh has been constructed, each edge is split by its farthest point from the surface if this point is farther than ε. In figure 10 the triangulation of a volcano is shown [26]; the upper left mesh is the initial one (no adaptation has been used), and it counts 354 triangles (nt). The other three are adaptive meshes obtained by refining the first one with different tolerances. Notice how the density and shape of the triangles follow the curvature and its principal directions. This *stretching* can be seen even better in figure 11; here the triangles are elongated in the direction of the cylinder axis.

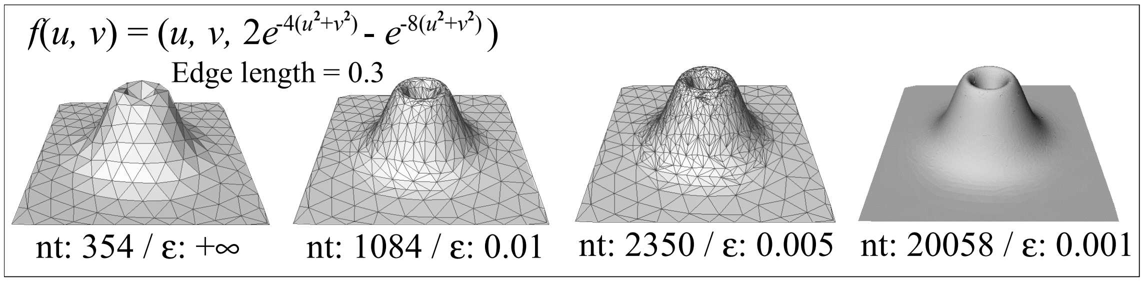


Figure 10: *Adaptive triangulation of a volcano with different tolerances.*

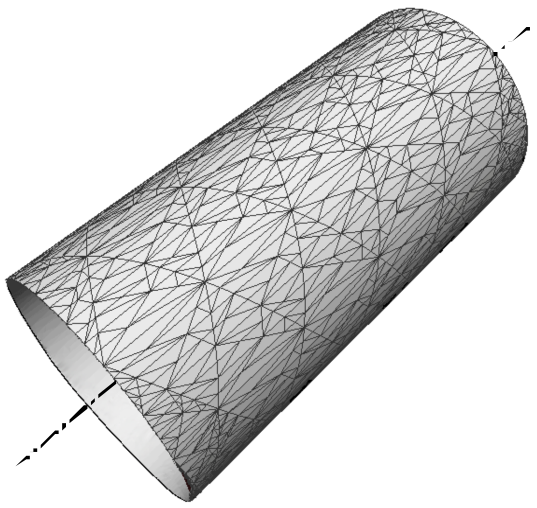


Figure 11: *Adaptive triangulation of a cylinder. Triangles are stretched along the direction of the axis.*

Conclusions

In this paper we presented a method for overcoming the limits of existing algorithms for the polygonization of parametric surfaces. The normal umbrella allows us to build a triangulation depending only on the surface's intrinsic characteristics, avoiding the problems of *bad* parameterization. The algorithm works on all regularly parameterized surfaces of class C^1, provided that the surface is a 2-manifold. The method for *sewing-up* the boundary of closed surfaces can be used to join several parametric patches, as usual in CAD applications. If the initial triangulation is fine enough, the shown adaptation criterion allows the construction of bounded error meshes. We can foresee that one of the first extensions of this method will be the introduction of a simplification technique that, maintains the selected error threshold, while decreasing the polygon count.

Acknowledgements

This work has been partially supported by the EC Project Nr. F0069 "THETIS". Thanks are due to all the members of the Computer Graphics Group of the IMA and the ones of the Computer Graphics and Vision Lab of the University of Otago, for their encouragement in writing the paper and their helpful advice.

References

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

[2] Joseph R. Tristano, Steven J. Owen and Scott A. Canann, *Advancing Front Surface Mesh Generation in Parametric Space Using a Riemannian Surface Definition*. Proc. of 7th Intl. Meshing Roundtable, 1998.

[3] Kenji Shimada and David C. Gossard, *Automatic triangular mesh generation of trimmed parametric surfaces for finite element analysis*. Computer Aided Geometric Design, Vol 15, pp.199-222, 1998.

[4] X Sheng and B Hirsch, *Triangulation of trimmed surfaces in parametric space*. Computer Aided Design, 24(8): pp.437-444, 1992.

[5] Hao Chen and Jonathan Bishop, *Delaunay Triangulation for Curved Surfaces*. Proc. of 6th Intl. Meshing Roundtable, pp. 115-127, 1997.

[6] Chandrajit L. Bajaj and Andrew Royappa, *Triangulation and Display of Rational Parametric Surfaces*. Proc. of Visualization'94, pp.69-76, 1994.

[7] Subodh Kumar, *Interactive Rendering of Parametric Spline Surfaces*. PhD thesis, Department of Computer Science, University of N. Carolina at Chapel Hill, 1996.

[8] L.A. Piegl and A.M. Richard, *Tessellating trimmed NURBS surfaces*. Computer Aided Design, 30(1): pp.11-18, 1998.

[9] Marco Attene and Michela Spagnuolo, *Automatic Surface Reconstruction from Point Sets in Space*. Computer Graphics Forum, Proc. of EUROGRAPHICS 2000, 19(3): pp.457-465, 2000.

[10] Nina Amenta, Sunghee Choi and Ravi Kolluri, *The power crust*. To appear in the sixth ACM Symposium on Solid Modeling and Applications 2001.

[11] Luiz Velho and Luiz H. De Figueiredo, *Optimal adaptive polygonal approximation of parametric surfaces*. Proc. of SIBGRAPI'96, pp.127-133, 1996.

[12] Marco Attene and Geoff Wyvill, Tech. Sketch, *Mapping independent triangulation of parametric surfaces*. SIGGRAPH 2001 Conference Abstracts and Applications, pp. 224, 2001.

[13] Kajiya JT, *Ray tracing parametric surfaces*. Computer Graphics, Proc. SIGGRAPH'82, 16(3):245-254, 1982.

[14] Joy KI and Bhetanabholta MN, *Ray tracing parametric surfaces patches utilizing numerical techniques and ray coherence.* Computer Graphics, Proc. SIGGRAPH'86, 20(4):279-285, 1986.

[15] Vasilis Vlassopoulos, *Adaptive polygonization of parametric surfaces*. The Visual Computer, Vol. 6, pp.291-298, 1990.

[16] Marc Vigo, Nùria Pla and Pere Brunet, *Curvature Adaptive Triangulations of Surfaces*. Tech. Rep. LSI-00-16-R, Universitat Politècnica de Catalunya, 2000.

[17] Marshall Bern and David Eppstein, *Mesh Generation and Optimal Triangulation*. Computing in Euclidean Geometry, 2nd Ed., pp.47-123, 1995.

[18] S Z Li, *Adaptive sampling and mesh generation*. Computer Aided Design, 27(3): pp.235-240, 1995.

[19] Frank J. Bossen and Paul S. Heckbert, *A Pliant Method for Anisotropic Mesh Generation*. Proc. of 5th Intl. Meshing Roundtable, pp.63-74, 1996.

[20] Erich Hartmann, *A marching method for the triangulation of surfaces*. The Visual Computer, 14(3): pp.95-108, 1998.

[21] Kenji Shimada, *Anisotropic Triangular Meshing of Parametric Surfaces via Close Packing of Ellipsoidal Bubbles*. Proc. of 6th Intl. Meshing Roundtable, pp. 375-390, 1997.

[22] M Kosters, *Curvature-dependent parameterization of curves and surfaces*. Computer Aided Design, 23(8): pp.569-578, 1991.

[23] Martin M. Lipschutz, *Shaum's Outline of Differential Geometry*, Ph.D., Hahnemann Medical College, McGraw-Hill, 1969.

[24] Marco Attene, Silvia Biasotti and Michela Spagnuolo, *Re-Meshing Techniques for topological analysis*. Shape Modeling and Applications, pp.142-151, 2001.

[25] William H. Press, *Numerical Recipes in C: The art of scientific computing*. Cambridge ; New York : Cambridge University Press, 1992.

[26] W. Seibold and G. Wyvill, *Near-Optimal Adaptive Polygonization*, Proc. CGI'99, pp. 206-213, IEEE Computer Soc 1999.