Adaptive Tessellation of NURBS Surfaces

F.J. Espino[1] M. Bóo[1] M. Amor[2] J.D. Bruguera[1]

[1]Department of Electronic and Computer Eng., University of Santiago de Compostela,
15782 Santiago de Compostela. Spain.
E-mail: {javi, mboo, bruguera}@dec.usc.es

[2]Department of Electronics and Systems, University of A Coruña,
15071 A Coruña. Spain.
E-mail: margamor@udc.es

ABSTRACT

NURBS surfaces are widely used in computer graphics, due to their great accuracy of design and reduced amount of data needed for representation. For real-time visualization, tessellation algorithms are needed, as they make use of the actual graphics hardware through the conversion of surfaces to triangle meshes. The existent algorithms for tessellation are only partially adaptive because the tessellation inside each part of the surface is uniform (non adaptive). We propose an algorithm that generates a mesh of triangles from a NURBS representation of the scene in a fully adaptive way, that is, the resolution is locally selected in such a way that the number of triangles to be processed is minimized without reducing the quality of the final image.

Keywords: Surface rendering, NURBS, Bézier, tessellation, subdivision

1 INTRODUCTION

NURBS surfaces [Foley96, Piegl97] (Non Uniform - Rational B-Splines) are used in several applications (CAD/CAM, virtual reality, animation, ...). Modelling complex geometries with NURBS permits high quality results with low storage requirements. Several methods have been proposed for NURBS rendering [Foley96]: ray tracing, pixel level subdivision, scan-line algorithms, ... However, due to the great advances in triangle-rendering hardware, the strategy mostly employed is the surface tessellation [Rockw89, Kumar96, Moret01, YingL02], that is, create a triangle mesh that approximates the original surface.

There are different methods [Rockw89, Kumar96] and hardware proposals [Moret01] for NURBS tessellation. All of them generate a triangle mesh for each surface section (patch) of the object, so the number of triangles per patch depends on its features (flatness, view point, ...). This is what we denote inter-patch adaptive tessellation. However, the mesh generated for each patch is regularly subdivided. This way, flat areas are fully subdivided and most of the triangles generated do not increase the quality of the image but the computational load.

In this paper we present an algorithm for NURBS surface tessellation. This algorithm generates an initial coarse mesh [Rockw89, Kumar96] and subdivide it adaptively using local tests [Dogge00, Amor02]. This way, we reduce the number of triagles needed for obtaining a tessellation of same quality by making the tessellation totally adaptive, that is, in addition to the inter-patch adaptive tessellation, the resolution of the mesh in the patch is not constant. Thus, we can generate high quality meshes with fewer triangles.

Journal of WSCG, Vol.11, No.1., ISSN 1213-6972
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

This paper is organized as follows. In Section 2, the mathematical formulation of NURBS and Bézier surfaces (Bézier surfaces are mathematically simpler than NURBS surfaces, so that the algorithms for processing them are also simpler) is given. Section 3 summarizes previous tessellation methods presented in the bibliography. Section 4 defines the proposed method for surface tessellation and its different steps. In section 5 we analyze the results obtained with our proposal. Finally, in section 6 we summarize the conclusions.

2 PARAMETRIC SURFACES

Parametric surfaces, like NURBS and Bézier surfaces [Piegl97], are geometric objects whose points in 3D space are calculated through the use of a small set of points (control points) and functions of real parameters u and v (blending functions). In this section, the equations needed to calculate points on the surface and normal vectors are given.

NURBS surfaces are parametric surfaces evaluated through equation [Piegl97]:

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)N_{j,q}(v)w_{i,j}} \quad (1)$$

where $\mathbf{P}_{i,j}$ is an array of control points, $w_{i,j}$ are the weights corresponding to each point, and $N_{i,p}(u)$ and $N_{j,q}(v)$ are the (p,q)-degree polynomials (u and v real values are defined by the knot vectors $U = \{0,..,0,u_{p+1},...,u_{r-p-1},1,...,1\}$ and $V = \{0,...,0,u_{q+1},...,u_{s-q-1},1,...,1\}$).

Almost all NURBS rendering and tessellation algorithms perform a NURBS to Bézier conversion, because the Bézier formulation complexity is lower, then the associated algorithms are simpler. Bézier surfaces are described through equation:

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{p}\sum_{j=0}^{q} B_{i,p}(u)B_{j,q}(v)w_{i,j}\mathbf{P}_{i,j}}{\sum_{i=0}^{p}\sum_{j=0}^{q} B_{i,p}(u)B_{j,q}(v)w_{i,j}} \quad (2)$$

where $B_{i,p}(u)$ and $B_{j,q}(v)$ are the (p,q)-degree Bernstein polynomials with $0 \leq u,v \leq 1$. In the conversion from NURBS to Bézier representation, a NURBS surface is split into a set of Bézier surfaces. From now on we will call patch to each Bézier surface generated.

In addition to surface points, normal vectors to the surface are also needed. These normal vectors are used in different algorithms such as shading operations [Foley96]. To evaluate the normal vector to the surface in a point of parametric coordinates (u_0, v_0):

$$\mathbf{N}_S(u_0,v_0) = \frac{\left(\frac{\partial \mathbf{S}(u,v)}{\partial u}\right)_{u_0,v_0} \times \left(\frac{\partial \mathbf{S}(u,v)}{\partial v}\right)_{u_0,v_0}}{\left|\left(\frac{\partial \mathbf{S}(u,v)}{\partial u}\right)_{u_0,v_0} \times \left(\frac{\partial \mathbf{S}(u,v)}{\partial v}\right)_{u_0,v_0}\right|} \quad (3)$$

where $\frac{\partial \mathbf{S}(u,v)}{\partial u}$ is the tangent vector to the surface in the u direction.

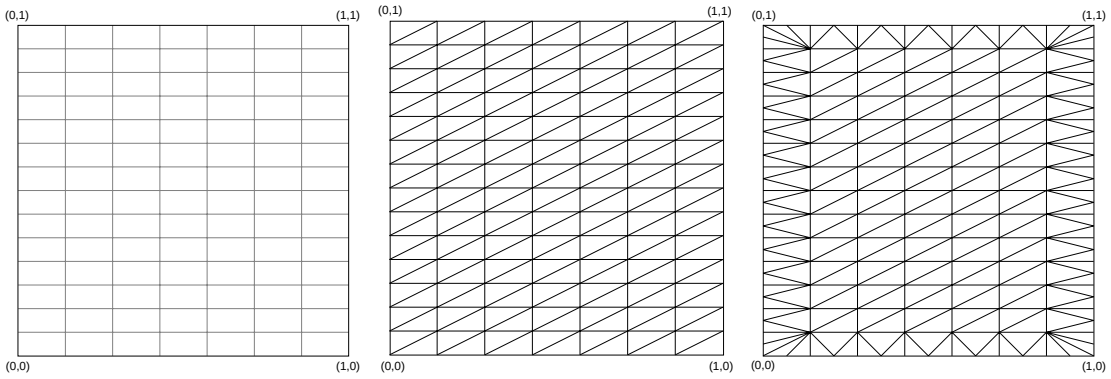
3 UNIFORM AND INTER-PATCH ADAPTIVE NURBS TESSELLATION

Tessellation is known as one of the main methods for parametric surface rendering. In this section we summarize the main procedure used for Bézier surface tessellation, that we will use as the starting point in our algorithm. Tessellation performed by this method can be either uniform (the same mesh resolution for all the figure) or *inter-patch* adaptive (each patch is tessellated with its own resolution, but patches are not tessellated adaptively in their local domain).

Almost all the Bézier surface tessellation algorithms [Rockw89, Kumar96] tessellates the surfaces in the parametric space, and perform the following steps:

- Partition of parametric domain (u,v) in $n_u \times n_v$ cells of size $\frac{1}{n_u} \times \frac{1}{n_v}$ (see Figure 1(a)). The parameters n_u and n_v may be the same for all patches in a uniform tessellation, or view-dependent and surface dependent in an inter-patch adaptive tessellation.

- Triangle generation. For a uniform tessellation, each cell is decomposed in two triangles, as shown in Figure 1(b). If the tessellation is inter-patch adaptive (i.e., n_u and n_v are different for each Bézier patch), an additional step is required in order to avoid cracks in the shared borders of the patches. In this step, the four borders of



(a) Parametric domain partition

(b) Uniform tessellation

(c) Inter-patch adaptive tessellation

Figure 1: Tessellation in parametric space

the patch are considered as four individual Bézier curves, and are partitioned independently. This way, the inner cells of the patch are transformed into triangles as in the uniform tessellation case, while, in the border of the patch, coving triangles are generated following the scheme of Figure 1(c). The individual partitioning for each bounding curve makes sure that shared borders are tessellated with the same number of points, avoiding cracks and surface defects in the final tessellation.

- The tessellation has been performed in the parametric domain, so the final step is to evaluate the Equations (2) and (3) in order to obtain the Euclidean space points and normal vectors, respectively. Either direct evaluation or alternative methods like forward-differences can be used [Foley96, Moret01].

4 FULLY ADAPTIVE NURBS TESSELLATION

The method we propose performs tessellation of Bézier patches with variable resolution inside the patch. This permits the use of meshes with a lower number of triangles for the same visualization quality. This method performs tessellation in two steps: first, an initial coarse tessellation is produced (section 3); in a second step, the initial mesh is processed in order to subdivide those regions that do not approximate the original image.

The algorithm proceeds as shown in Figure 2. As

```
Calculate_initial_mesh;
List=initial_mesh;
while(List!=empty) {
    Extract_triangle(List);
    Test(edge1,edge2,edge3);
    if(test=TRUE)
        render_triangle;
    else {
        subdivide_triangle;
        List←new_triangles;
    }
}
```

Figure 2: Fully adaptive tessellation algorithm

an initial step, a triangle list is created in order to store the triangles of the initial tessellation. Each triangle is defined by three vertices, and each vertex by eight coordinates: two parametric coordinates (u, v), three space coordinates (x, y, z) and three coordinates of the normal vector to the surface (n_x, n_y, n_z). Once the initial mesh is stored and a loop is started, in each iteration, a triangle from the list is processed; a test is performed for each triangle edge and, in case one or more edges do not pass the test (i.e., the quality of the tessellation is not good enough in the analyzed region), the triangle is subdivided and the resulting triangles are stored in the list for future processing. If the test is passed by the three edges, the triangle is sent to be rendered. This loop iterates until the list is empty, meaning that the whole patch is rendered.

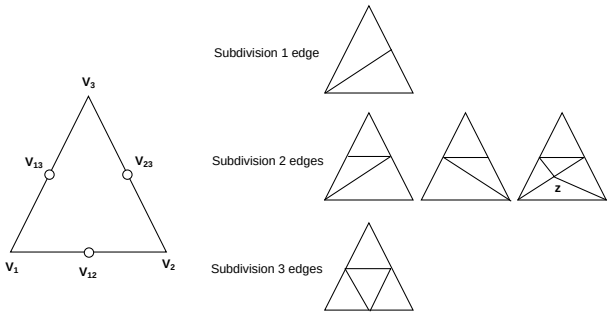


Figure 3: Triangle subdivision

In the next section, the different steps of the algorithm are explained and the different tests that we use to subdivide the patch are enumerated and explained.

4.1 Initial Tessellation

In order to obtain an initial mesh, a partition in parametric domain is performed (section 3). This partition may be uniform (the same for every patch) or inter-patch adaptive (different for each patch).

If the initial mesh is uniform, the parametric coordinates are the same for every patch, so tessellation is immediate. For instance, if the initial mesh is made of two triangles, the mesh would be defined by points {{*(0,0), (1,0), (1,1)*}, {*(0,0), (1,1), (0,1)*}}. The rest of the coordinates *(x, y, z, n_x, n_y, n_z)* are computed through Equations (2) and (3).

If the initial tessellation is inter-patch adaptive (as is the one shown in Figure 1(c)), the number of triangles of each patch depends on the view point and the surface shape. As in the uniform case, the space coordinates and normal vectors are calculated through Equations (2) and (3).

4.2 Fully Adaptive Patch Subdivision

Taking the initial mesh as a starting point, the triangle edges are examined using tests and a decision about the insertion of a new vertex in the edge is made. If one or more edges are subdivided, this will result in two or more triangles that will be analyzed again until all tests are passed. The testing of each edge is done in two steps: the calculation of the new vertex, and the test itself, which will decide whether the new vertex is going to be inserted.

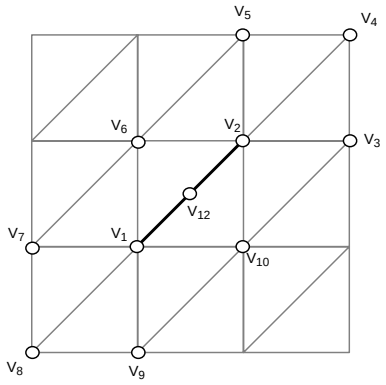


Figure 4: First order neighbors of edge $V_1 - V_2$

The parametric coordinates of the new vertex are calculated using the vertices of the edge. The new point is the middle point in parametric space ($V_{12} = \frac{(u_1,v_1)+(u_2,v_2)}{2}$). The rest of the coordinates are calculated through the evaluation of the Bézier equations (Equations (2) and (3)).

Each triangle has to be tested in order to decide on the insertion of the new vertices. These tests use the edge information and the neighbors vertices information to calculate the edge-surface distance. The tests give a boolean result for each edge. Depending on these three values, the triangle is subdivided using the scheme shown in Figure 3. In this figure the subdivision schemes for one, two and three inserted vertices are shown. Note that the subdivision of two edges results in three possibilities. The subdivision scheme producing an additional vertex z was not considered in this paper since it produces the same quality with a higher computational cost than the other schemes.

Several tests we have employed have been used in adaptive subdivision of meshes based on displacement maps [Dogge00, Amor02]; however, extension to Bézier surfaces has not been done. Next, these tests and several new tests are presented. The first group of tests is made of tests that use the normal vectors of the edge vertices and neighbors, while the second group use the mesh information for measuring the flatness of the mesh in the region. Both tests measure whether the mesh is smooth enough in the region around the analyzed edge or not.

4.3 Normal Deviation

Normal comparison tests calculate the normal vector deviation between the analyzed points and

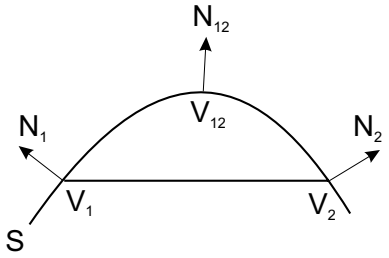


Figure 5: One edge test

its neighbors. The purpose of using these tests is to ensure that normal vectors of adjacent vertices are not too different, avoiding a bad shading in the rendered mesh.

There are several options for performing the comparison. Next, the proposed tests are summarized, beginning with tests that only use local edge information (vertices V_1, V_{12}, and V_2 in the Figure 4) and generalizing these tests for using information of all the first order neighbors (vertices V_1 to V_{10} in the Figure 4).

4.3.1 One-edge normal test

The easiest way for normal comparison is using the local edge information (i.e., vertices V_1, V_{12} and V_2 shown in Figure 5) and comparing normal vectors (N_1, N_2 and N_{12}) using coordinates comparison, according to:

$$Test1 = (|N_1 - N_{12}| > t_1)\ OR\ (|N_2 - N_{12}| > t_1) \quad (4)$$

where t_1 is a threshold. If one coordinate of vectors $(N_1 - N_{12})$ or $(N_2 - N_{12})$ is larger than the threshold, the test is not passed, and the edge $V_1 - V_2$ is subdivided inserting the point V_{12}.

4.3.2 One-edge complementary test

A modification of this test is developed in [Amor02]. This modification makes an additional comparison between the normal vectors of the edge vertices:

$$Test2 = (|N1 - N2| > t_2) \quad (5)$$

where t_2 is a threshold different to t_1. This way, the modified test is the combination of $Test1$ and $Test2$:

$$Test12 = \ Test1\ OR\ Test2 \quad (6)$$

i.e., if one of the tests ($Test1$ or $Test2$) is greater than its own threshold, the new vertex is inserted.

4.3.3 Two triangle test

This test uses local edge information and the two vertices of the triangles sharing the edge (V_6 and V_{10} in Figure 4). The test is:

$$Test3 = (|N_{12} - N'|) > t_3 \quad (7)$$

where $N' = \frac{N_1+N_2+N_6+N_{10}}{4}$

4.3.4 First order neighbors test

The most general case uses all the first order neighbors (vertices V_1 and V_{10} in Figure 4) in the comparison. The test uses the following expression:

$$Test4 = (|N_{12} - N''|) > t_4 \quad (8)$$

where $N'' = \sum_{i=1}^{n} \frac{N_i}{n}$ and n is the number of first order neighbors.

4.4 Flat Tests

The flat tests use the spatial coordinates of the triangle vertices for testing the flatness of the mesh. If the region near an edge is flat, the insertion of a new vertex does not increase the quality of the mesh. In the following subsections, several methods that test the flatness of the mesh in the region near a vertex are presented. From the most straightforward, that use local edge information, to the more complex, that use neighbor information.

4.4.1 Vector deviation flat test

To ensure that the mesh is flat enough, the distance between the new vertex V_{12} (see Figure 5) to the edge of the triangle where the new vertex may be inserted is measured. This test computes the normalized vector $|V_1 - V_2|$ and the dot product of this vector with $|V_{12} - V_1|$ and $|V_{12} - V_2|$. This way, though vertex-edge distance is not directly measured, we calculate the deviation between the vectors that point to the new vertex and the edge vector. The test consist of the following steps:

1. Calculation and normalization of vectors $A = V_1 - V_2$, $B = V_{12} - V_1$ and $C = V_{12} - V_2$.

2. Computation of the unsigned dot products $|B \cdot A|$ and $|C \cdot A|$.

(a) Teacup (b) Single_patch (c) Teapot

Figure 6: Sample meshes

3. Comparison between the dot products and a threshold t_{f_1}. If one of them is smaller than the threshold, the new vertex is inserted. The test can be represented by the equation:

$$flat = \quad (|B \cdot A| < t_{f_1}) \\ OR\ (|C \cdot A| < t_{f_1}) \qquad (9)$$

4.4.2 Local flat test

An alternative way to test the flatness around a vertex is to compare the tangent vector with the normal vector to the analyzed vertex. If the region around the vertex is flat, both vector are perpendicular (dot product is zero). The steps are:

1. Calculation and normalization of vector $U = |V_1 - V_2|$.
2. Calculation of the unsigned dot products of U with normal vectors of the two vertices of the edge ($U \cdot N_1$ and $U \cdot N_2$).
3. Comparison between both dot products and the threshold t_{f_2}. If one of them is greater than the threshold, the edge is subdivided and V_{12} inserted. The test is summarized by the following equation:

$$flat2 = \quad (|U \cdot N_1| > t_{f_2}) \\ OR\ (|U \cdot N_2| > t_{f_2}) \qquad (10)$$

4.4.3 First order neighbors flat-test: The umbrella operator

The test defined in Equation (10) is a particular case of a more general test that uses first order neighbors of the analyzed edge. This test makes use of the umbrella operator [Kobbe98, Amor02]. This operator is defined by:

$$U(V) = \frac{1}{n} \sum_{i=0}^{n-1} V_i - V \qquad (11)$$

where V is the vertex whose flatness is being analyzed and V_i the neighbor points.

The testing of the flatness can be calculated using the dot product of $U(V)$ and the normal vector to the analyzed vertex. The steps of this test are:

1. Calculation and normalization of the umbrella operator over the points $V = V_1$ and $V = V_2$. The operators are U_1 and U_2.
2. Dot product of normalized operators and the normal vectors ($U_1 \cdot N_1$ y $U_2 \cdot N_2$).
3. Comparison between the dot product and the threshold t_{f_3}. If it is greater than the threshold, the new vertex is inserted.

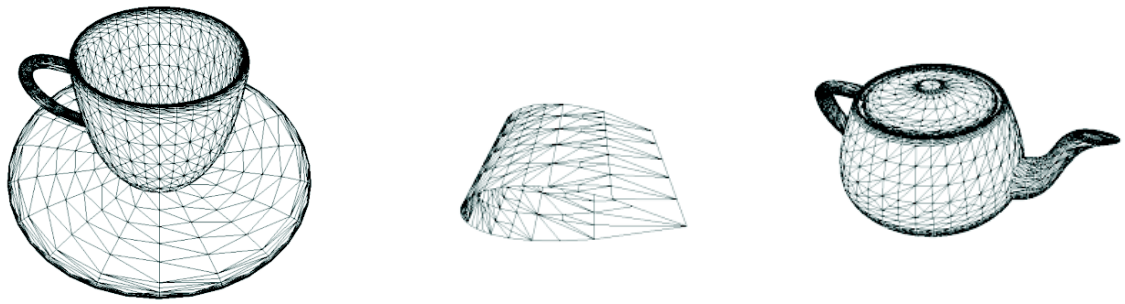
5 RESULTS

We have simulated our algorithm implementing it in C and using a set of sample figures to check the performance of our proposal. From the simulation we measure the number of triangles of the tessellation and an estimation of the resulting mesh quality.

The implementation of the algorithm follows the scheme of Figure 2. First a list is generated with the triangles of the initial mesh, and then, they

Figure	Test1		Test12		Flat1		Flat2	
	Triangles	Error	Triangles	Error	Triangles	Error	Triangles	Error
teacup	2648	0.008172	2464	0.008233	1986	0.009654	2352	0.009533
	3638	0.007525	4402	0.006612	2978	0.008714	7882	0.003332
	12400	0.002838	15458	0.002967	17052	0.001603	27076	0.001536
single_patch	140	0.050421	250	0.045949	150	0.048982	240	0.023283
	320	0.043934	360	0.016169	260	0.022560	600	0.009515
	920	0.008081	1140	0.007554	820	0.005343	1550	0.003017
teapot	2876	0.026105	3312	0.024850	2394	0.026016	2942	0.023627
	5916	0.019524	5360	0.019492	3790	0.022476	9036	0.007757
	18584	0.006190	21244	0.006119	16994	0.005242	33154	0.002527

Table 2: Tessellation results



(a) Resulting Mesh for teacup

(b) Resulting Mesh for single_patch

(c) Resulting Mesh for teapot

Figure 7: Resulting meshes for Flat2 test

Figure	Triangles	Error
teacup	1300	0.0109
single_patch	50	0.1022
teapot	1200	0.0311

Table 1: Initial meshes triangles and error

are processed sequentially, storing in the list the triangles resulting from subdivision, and rendering those that don't need subdivision.

The results associated to the tests based on neighbor information (Equations (7), (8) and umbrella operator) are not included in this analysis due to the quality of the final images obtained is similar to the one obtained with the tests based on local information; but the implementation cost is larger.

We analyze two features: The mesh error and the number of triangles. The mesh error is an estimation of the distance between the real surface (a uniform tessellation with a very high number of triangles per patch) and the resulting mesh. The objective is the generation of the minimum number of triangles without reducing the quality of the final image. The threshold is selected as a tradeoff between number of triangles and error.

In Figure 6 the sample meshes we employed are shown (similar results were obtained with other meshes). The number of triangles and mesh errors of the initial meshes are shown in Table 1. The number of triangles is determined by the number of patches per image (26 patches for Figure 6(a), 1 for Figure 6(b) and 24 for Figure 6(c)) where 50 triangles per patch were considered in the initial uniform tessellation. The objective is reducing the error by means of an adaptive subdivision so that the number of triangles is minimally increased.

The mesh error and number of triangles are shown in Table 2. Specifically, the table shows the number of triangles and the error for each sample mesh and for each test considered. As

expected, if the number of triangles is increased, the error diminishes for all tests under analysis.

The results for the four tests show that the sampled meshes are adaptively subdivided. This way, the number of triangles generated can be chosen by the user in function of the desired quality of the final image. Although the performance of all tests is good, tests based on flatness (Flat1 and Flat2 in the table) are slightly better; that is, for a similar number of triangles the error is smaller.

In Figure 7 the adaptively subdivided meshes (with low number of triangles) employing the Flat2 test are shown. In the three figures, it can be clearly observed that regions of higher curvature are subdivided with more triangles. Then, our method tessellates surfaces in a fully adaptive way; so that the number of triangles depend on the surface curvature.

6 CONCLUSIONS

In this paper we have presented a fully adaptive tessellation method for NURBS. It is well known that the quality of flat regions is not improved by incrementing the number of triangles. Based on this fact, we have proposed an adaptive subdivision method that analyzes the flatness of the meshes and permits to reduce the number of triangles in areas with smooth curvature. That is, the number of triangles depends on the surface curvature.

We have checked tests based on the analysis of normal and tangent vectors to the surface in a neighborhood of the area to be subdivided. Among these tests we have checked tests using only local information (two vertices) and tests using first order neighborhood information (more than two vertices). Similar results are obtained in both cases but the computational load of the local tests is lower.

Among the tests based on local information better results are obtained when the flatness properties are directly analyzed using the tangent and normal vectors.

Our method shows that the NURBS surfaces can be adaptively tessellated in an efficient way using tests that have been previously proposed only for adaptive subdivision of triangle meshes based on displacement maps.

7 ACKNOWLEDGMENTS

This work was supported in part by the Ministry of Science and Technology of Spain under contract MCYT-FEDER TIC2001-3694-C02-01. M. Amor has been supported in part by research funds of the University of A Coruña (Spain).

REFERENCES

[Amor02] M. Amor and M. Bóo. Adaptive Tessellation of Triangle Meshes According to the Displacement Map. Technical report, Universidad de Santiago de Compostela, http://www.ac.usc.es, 2002.

[Dogge00] M. Dogget and J. Hirche. Adaptive View Dependent Tessellation of Displacement Maps. *Siggraph/Eurographics Workshop on Graphics Hardware*, pages 59–66, 144, 2000.

[Foley96] J.D. Foley. *Computer Graphics: Principles and Practice*. Ed. Addison-Wesley, 1996.

[Kobbe98] L. Kobbelt, S. Campagna, J. Vorstz, and H.P. Seidel. Interactive Multi-Resolution Modeling on Arbitrary Meshes. *SIGGRAPH '98 Conference Proceedings*, pages 105–114, 1998.

[Kumar96] S. Kumar, D. Manocha, and A. Lastra. Interactive display of large NURBS models. *IEEE Transactions on Visualization and Computer Graphics*, 2(4):323–336, 1996.

[Moret01] H. Moreton. Watertight Tessellation Using Forward Differencing. *ACM Siggraph/Eurographics Workshop on Graphics Hardware*, pages 25–32, 2001.

[Piegl97] L. Piegl and W. Tiller. *The NURBS Book*. Ed. Springer, 1997.

[Rockw89] A. Rockwood, K. Heaton, and T. Davis. Real-Time Rendering of Trimmed Surfaces. In *ACM Siggraph*, pages 107–117, 1989.

[YingL02] M. YingLiang and T. Hewitt. Adaptive Tessellation for Trimmed NURBS Surface. *Eurographics*, 2002.