Evaluation of NURBS surfaces: an overview based on runtime efficiency

H. Sánchez
University of Extremadura
Centro Universitario de Mérida
Santa Teresa de Jornet, 38
06800 Mérida - Spain
sasah@unex.es

A. Moreno, D. Oyarzun
VICOMTech
Pº. Mikeletegi, 57
20009 San Sebastián - Spain
amoreno@vicomtech.es

A. García-Alonso
University of Basque Country
Computer Science School
Pº. Manuel de Lardizabal, 1
20009 San Sebastián - Spain
agalonso@si.ehu.es

ABSTRACT

Dynamic tessellation of (trimmed) NURBS surfaces is an important topic for interactive rendering of complex models. Independently of the decomposition method used to approximate surfaces by polygons, an evaluation of the surface at many parametric pairs is required to generate meshes vertices. In this paper we give an overview of the most important evaluation algorithms and compare them regarding their runtime efficiency.

Three different representations and five algorithms for the evaluation of NURBS surfaces are compared: direct evaluation of NURBS surfaces, evaluation of their Bezier patches and evaluation of them in power basis representation. We add to the comparison two algorithms, based on the approximated computation of normal vectors. The deviation among the exact and the approximated normal vectors is measured. The obtained results are checked with partial comparisons that appear in the bibliography.

Keywords

NURBS rendering, NURBS evaluation, Bezier evaluation, power basis evaluation, normal vectors

1. INTRODUCTION

In many applications of CAD/CAM, virtual reality, animation and scientific visualization, object models are described by NURBS surfaces. This representation allows defining exactly both algebraic geometries and abstract surfaces standardizing the representation by a single mathematical equation. Moreover, thanks to their growing use in CAD/CAM, NURBS surfaces appear in the main neutral file formats for interchanging geometric data, like IGES and STEP. Farin [Far01a] and Pielg and Tiller [Pie97a] have studied NURBS surfaces in depth.

Because of its importance in computer graphics applications, the rendering of NURBS surfaces has been researched intensively in last three decades.

WSCG SHORT Communication papers proceedings
WSCG'2004, February 2-6, 2004, Plzen, Czech Republic.

Different approaches have been elaborated for visualization. In photo-realistic rendering algorithms, the color intensity value of each screen pixel is generated directly from the parametric surface description of the model. These algorithms can be classified in based on: scan-line [Bli78a] [Whit78a] [Lan80a], ray tracing [Whi80a] [Kaj82a] [Qin97a] [Mar00a] and iso-curve sequences [Roc87a] [Cha89a] [Elb96a]. In hardware rendering [Bed91a] [Gop97a] [Boo00a], the graphical hardware is extended with more complex geometric primitives than the polygon. This hardware has VLSI architectures that make possible the computation of points and normal vectors of a surface. In approximation based rendering algorithms, more simple primitives like polygons [Her87a] [Roc89a] [Pie98a] [Kum96a] or points [Chh03a] approximate surfaces. These algorithms benefit from the graphical hardware features to generate more quickly the image. Moreover, the approximation of surfaces is done outside the graphical API. This fact allows selecting the optimal level of detail in runtime. Therefore, the application can manage the quality of the image to generate in runtime as the function of the wanted interactivity degree or the available graphical hardware features in a local computer.

1.1 Main Contributions

The first contribution of this work is an overview of NURBS surfaces evaluation methods based on runtime efficiency. Other authors have done partial comparisons (only comparisons among two or three methods). The second contribution is a comparison (based on computational cost and image quality) among four approximated normal vectors calculation with different weighting methods. The overview includes two evaluation methods that use approximated normal vectors calculation (weighted by area).

1.2 Paper Structure

The rest of the paper is organized as follows: In section 2 a background to understand better the need of our work is done. In section 3 we describe and analyze the approximated normal vectors calculation methods. Section 4 covers the overview results and section 5 concludes and describes future work.

2. BACKGROUND

2.1 Dynamic Tessellation

In dynamic tessellation algorithms, meshes are generated in the interactive stage of the rendering pipeline. Many systems tessellate NURBS surfaces as a pre-process. However, there are two reasons for using dynamic tessellation. In first place, storing NURBS surfaces instead of storing polygonal representations saves a large percentage of available memory. This is especially interesting when using large models where many surfaces are not visible and their incorporation to the potentially visible surfaces set will be gradual. In second place, an appropriate tessellation can be obtained in runtime as the function of the actual viewing conditions. So, this kind of algorithms is very advisable for interactive visualization. On the other hand, as dynamic tessellations must be built as fast as possible, they often are of lower quality than those generated as a pre-process.

The number of polygons generated by a dynamic tessellation is the function of the relative position of the surface respect to the point of view, i.e., its projection onto the screen. However, tessellations can be generated to meet memory requirements or a continuous frame rate.

In dynamic tessellation, uniform decomposition is generally used because it is less time consuming than adaptive decomposition. Some works use adaptive algorithms for interactive visualization. Guthe et al. [Gut02] computes a fine tessellation in the pre-processing stage. In runtime, split and collapse vertex operations are done to adjust view dependently the quality of the meshes. Chhugani and Kumar [Chh01a] pre-compute the set of surface vertices that reduces the distance of the generated mesh to the surface. In runtime, their algorithm updates the tessellation adding or removing vertices of a Delaunay incremental triangulation that represents the tessellation. Uniform decomposition tessellates the surface using a regular grid defined in the parametric domain. This does not guarantee that the resulting tessellation will be uniform. Anyway, it is possible to determine a parametric grid size that may produce polygons that meet certain restrictions. For instance, polygons that projected onto the screen will be within specific size bounds.

In uniform decomposition based algorithms, the mesh generation is composed of three operations:

- determine the tessellation density or step sizes (distance between two consecutive parametric points in a parametric direction),
- evaluate points and normal vectors of the surface for the grid of parametric points,
- generate the sequence of polygons (normally triangles) that approximate the surface.

As all the operations are done in the interactive loop, it is necessary to set bounds to their computational cost. This paper researches solutions for the second operation. Although several methods have been published dealing with point and normal vectors evaluation, this topic has few times been studied from the point of view of interactive visualization using dynamic tessellations [Roc89a] [Kum96a].

2.2 Generation of mesh vertices

Obtaining a mesh vertex of the tessellation involves evaluating the NURBS surface in order to generate a three-dimensional point and its associated normal vector. Given a parametric coordinate (u,v), evaluation of a surface point means to obtain its own three-dimensional point $S(u,v)$ in the surface. The exact normal vector of a tensor product surface is calculated by the cross product of two tangent vectors obtained as partial derivatives.

NURBS surfaces are evaluated using three different representations. Each representation has its own set of point and normal vectors evaluation algorithms. In the first representation (B-spline), points and normal vectors are generated evaluating directly the NURBS mathematical description. The second is the Bezier representation. Each NURBS surface is decomposed in a rectangular matrix of Bezier patches whose union is geometrically the same as the original surface. Points and normal vectors are generated using Bezier surface evaluation methods. The third is the power basis representation. Here, Bezier patches are converted into the power basis. It is well known that the conversion from Bezier to the power basis is numerically unstable when polynomial degree is high. Farouki [Far91a] makes a comparative among Bezier

and power basis representations and analyzes the robustness and errors involved in this transformation. However, that work does not provide performance comparison data.

In our performance study we do not consider the computational cost required to convert from the NURBS representation to the Bezier and power basis ones. Although those representations require more memory than the original NURBS representation, the storage consumed is much lower than that used for polygonal representations. So, our tests consider only the time required for the point and normal vectors evaluation.

In the bibliography, few works compare Bezier representation algorithms: Carriére [Car95a] compares the deCasteljau algorithm and its modified version. The same happens with B-spline representation algorithms: Luken and Cheng [Luk96a] compares the two-stages Cox-de-Boor algorithm with the knot insertion method.

Even less works compare algorithms that belong to different representations: Sederberg [Sed95a] compares the deCasteljau algorithm (Bezier representation) with the Horner's scheme based algorithm (power basis representation).

Moreover, a global comparison (considering all representations) has not been published. In this work, we evaluate in the same conditions the best five algorithms proposed in the bibliography for evaluating a surface in B-spline, Bezier and power basis representation:

- For direct NURBS evaluation we have tested two-stages Cox de Boor [Luk93a] algorithm and optimized direct evaluation [Pie97a]. The first method avoids repeated calculations of the same components or intermediate values (inverses of differences of u and v knots) by calculating and storing them in both directions first, and then evaluation the tensor products at the parametric pairs. The last method uses coherence between mesh vertices to speed up the evaluation of vertices that lies in the same parametric span. We do not include the knot insertion method in the comparison because Luken and Cheng [Luk96a] demonstrate in their work that knot insertion is less efficient than two-stages Cox de Boor.
- Substituting the NURBS by a Bezier representation we have tested two algorithms: deCasteljau [deC86a] and modified deCasteljau [Man95a]. We have improved deCasteljau algorithm using the pre-calculation of the weights affecting intermediate points of the recursion, as proposed by Carrière [Car95a].
- For polynomial representation we have tested Horner's scheme [Sed95a]. In this method, coefficients that multiply the powers of polynomials are previously pre-calculated and stored.

In our work we have also considered the following fact. When geometric information is used only for image generation, it is not necessary to calculate the exact normal vectors for surface points: it is enough to obtain a good approximation to them.

We have made rich the comparison adding our original results obtained when normal vectors are calculated approximately by interpolation. This methodology is applied to the optimized direct evaluation and the Horner's scheme.

In the next Section we study the problems and advantages of computing approximations to normal vectors.

3. CALCULATION OF APPROXIMATED NORMAL VECTORS

It is well known how descriptions of polyhedra that only have vertex coordinates as geometry data can be processed in order to obtain a normal vector for each vertex. This procedure is carried out routinely as preprocess. It is performed when polygonal descriptions that lack normal vectors are read from files. It is well known that it helps to save memory in files and the vectors approximated in this way are usually acceptable for shading purposes.

We are using this technique for speeding the computation of meshes from NURBS descriptions. For this reason we have studied two problems. In first place we have revised the different methods that can be used to approximate normal vectors. Our goal was to compare their computational cost and the goodness of the approximations they provide. In second place we wanted to find a tessellation method that could be applied to NURBS surfaces ignoring data from their neighbor surfaces. As we will see, this goal obliged us to extend the algorithm that approximates normal vectors for the vertices of polyhedra. We insert a final subsection that discusses one finding we were not looking for: when using approximated normal vectors it is not required to check for numerical errors that sometimes appear when computing normal vectors exactly.

The surface normal at a mesh vertex can be approximated averaging normal vectors of its adjacent facets (using uniform decomposition, four triangles are required to evaluate the normal vector of a interior vertex). Facet normal vectors can be weighted as the function of their geometrical or

topological properties. Thürmer and Würthrich [Thü98a] use the angle of each facet that has a bearing on the vertex as the normal weight. Max [Max99a] derives weights using the facet area and assigns larger weights to smaller facets. If facet normal vectors are not weighted, we say that normal vectors are uniformly weighted. We add a new method of weighting that improves the previously described methods. In this method we assign smaller weights to smaller facets.

We have compared these four methods that approximate the normal vector at a mesh vertex.

$$Uniformly\ weighted : N = \frac{\sum_{i=1}^{4} N_i}{\left|\sum_{i=1}^{4} N_i\right|}$$

$$Weighted : N = \frac{\sum_{i=1}^{4} \alpha_i N_i}{\left|\sum_{i=1}^{4} \alpha_i N_i\right|}$$

where α_i is the associated weight to the facet i.

As the results in the following subsection show, approximated normal vectors do not differ much using one method or another. However, area weighted normal vectors are computed faster. This was not a surprise, because we realized the following fact when we were preparing the different algorithms: weighting normal vectors with area, instead of adding operations to the algorithm, reduces the need for making them. The other methods compute the normal using triangle edges and then normalize its module before adding them.

If this operation is not performed, what we have is a weighted area normal for the polygon. These savings are not important when dealing with non-deformable polyhedra whose normal vectors are computed as preprocess, but they provide some help when trying to compute meshes in the fastest possible way within a simulation.

Now we will discuss another problem. When a vertex is on the edge of the surface, all these methods need to know the tessellation of the neighbor surface. In this case, when the normal is computed using only facets from the surface that contain the vertex, in the final shaded image surface edges appear clearly. This is due to a lack of continuity in the shading between contiguous surfaces caused by using different normal vectors for common vertices of neighbor surfaces. Obviously, if neighbor surfaces have continuity C^1, and normal vectors are exactly computed, this problem does not appear.

We have devised a method to keep the independence among surfaces while using approximated normal vectors. Figure 1.a shows the straightforward way of computing the normal at a vertex that is shared by two adjacent surfaces and Figure 1.b shows the extrapolation that we make when computing the normal without using adjacent surface data. The figures simplify the problem using a 2D analogy. This is nearly real because the vertices that appear in the figure are on the same iso-parametric curve.

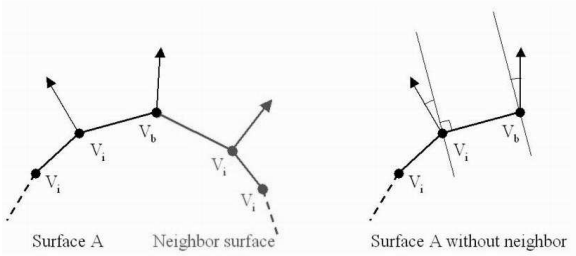

Figure 1: a) Two surfaces, b) Extrapolation of the normal in the last interior vertex.

The process used for computing the normal in a boundary vertex is the following one. Following an iso-parametric curve, once the normal vector for the last interior vertex is computed, we compute another vector. It has the following characteristics: it is parallel to a plane defined by the normal in the last interior vertex and the segment that joins that last interior vertex with the boundary vertex; it is also perpendicular to that segment. Then we use this vector as the symmetry axis to obtain the symmetric vector to the normal in the last interior vertex. So, the normal in the boundary vertex, is "pulled out" extrapolating the last normal computed along its own iso-parametric curve.

Next subsection presents the computational cost and precision in the tests that have been performed.

3.1 Computational cost and results quality

In Table 1, two examples of the goodness of the approximated normal vectors in an edge shared by two adjacent surfaces are showed. For this, the average deviation between approximated normal vectors obtained in each adjacent surface edge is measured. The test is made for two C^1 continuity adjacent surfaces and two C^2 continuity adjacent surfaces and with different resolutions of the tessellation. As it can be seen in the table the deviation is nearly null.

	Model			
	C^1 adjacent surfs.		C^2 adjacent surfs.	
Number of Vertices	Average	Max.	Average	Max.
40K	0.76º	1.56º	0.01º	0.02º
32K	0.81º	1.78º	0.01º	0.02º
21K	1.08º	2.02º	0.02º	0.03º
9K	1.37º	2.53º	0.06º	0.07º
1.6K	2.15º	3.97º	0.09º	0.09º

Table 1: Average and maximum deviation (in degrees) between approximated normal vectors in and edge shared by two adjacent surfaces.

In Table 2, the computational cost in the generation of the approximated normal vectors with each weighting method is showed. In the four methods, the extrapolation of the normal vectors in the boundary vertices is included. Weighted by area is the most efficient method.

Model (Vertices)	Uniform	Area	Inverse Area	Angle
Goblet (40K)	0.55	0.46	1.1	1.22
Torus (40K)	0.55	0.49	0.86	0.66
Utah Teapot (5K)	0.06	0.05	0.09	0.1

Table 2: Time comparative (in seconds) among four approximated normal vectors calculation with different weighting methods.

In Tables 3 and 4, the averaged angular deviation between the exact and approximated normal vectors is showed for models with different geometric complexity.

Number of Vertices	Uniform	Area	Inverse of Area	Angle
40K	0.01º	0.01º	0.01º	0.01º
32K	0.01º	0.01º	0.01º	0.01º
21K	0.01º	0.01º	0.01º	0.01º
9K	0.02º	0.03º	0.02º	0.01º
1.6K	0.11º	0.20º	0.20º	0.03º

Table 3: Average angular deviations (in degrees) among approximated and exact normals in Torus model (see Figure 2).

Number of Vertices	Uniform	Area	Inverse of Area	Angle
40K	0.92º	0.91º	0.92º	0.93º
32K	1.02º	1.02º	1.03º	1.03º
21K	1.29º	1.28º	1.30º	1.31º
9K	2.03º	1.94º	2.04º	2.15º
1.6K	5.07º	4.95º	5.14º	5.46º

Table 4: Average angular deviations (in degrees) among approximated and exact normals in Goblet model (see Figure 3).

In Figures 2, and 3, the comparison among the generated images with exact normal vectors computation and weighted by area approximated method is shown.

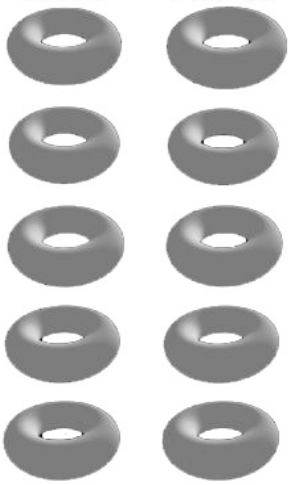

Figure 2: a) Exact Normals. b) Approximated Normals (weighted by Area). Resolutions of tessellations are from top to bottom: 40K, 32K, 21K, 9K and 1.6K vertices.

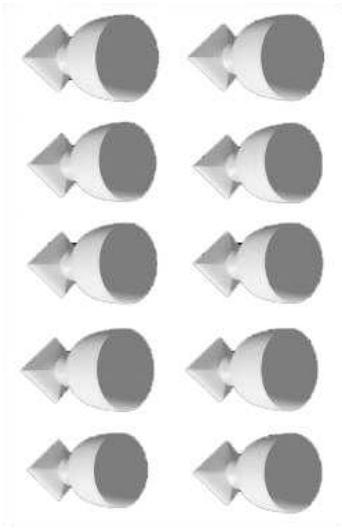

Figure 3: a) Exact Normals. b) Approximated Normals (weighted by Area). Resolutions of tessellations are from top to bottom: 40K, 32K, 21K, 9K and 1.6K vertices.

4. OVERVIEW RESULTS

The computational cost of the evaluation methods has been obtained testing the following models illustrated in Figure 4 (the run time efficiency of evaluation algorithms is uncoupled from the trimming process; for this reason we have made tests with non-trimmed surfaces):

- Sup15x15 is a NURBS surface of bi-degree 15, which is decomposed in 64 Bezier patches.

- Sup10x10 is a NURBS surface of bi-degree 10, which is decomposed in 25 Bezier patches.
- Sup5x5 is a NURBS surface of bi-degree 5, which is decomposed in 17 Bezier patches.
- Partial Lion is composed of 28 bi-cubic NURBS surfaces. Its Bezier decomposition leads to 456 patches.
- Partial Ride is composed of 37 bi-cubic NURBS surfaces. Its Bezier decomposition leads to 170 patches.
- Utah teapot is composed of 28 bi-cubic NURBS surfaces that are represented by the same number of Bezier patches
- Goblet is a NURBS surface of degree 2x3. Its Bezier decomposition leads to 72 patches.

To compare the computational cost of vertex evaluation methods, these models have been uniformly tessellated into a large number of vertices.

In Table 5, computational times obtained with each evaluation method and applied over each model is shown. These tests have been made with a 600MHz AMD K7 with 386 MB memory.

In Bezier representation, one of Carrière conclusions has been corroborated: deCasteljau is more efficient than modified deCasteljau. However, according to Carrière, this fact only happens until bi-degree six surfaces. This conclusion is corroborated with the evaluation times taken by Sup.10x10 and Sup.15x15 (CAD models sometimes have patches of degree 15x15 or even higher). For larger bi-degrees, modified deCasteljau is more efficient than deCasteljau. Moreover, while the initial algorithm is used with bi-degree surfaces, the second one can be effectively used in surfaces with different degree in each parametric direction.

In NURBS representation, original contributed results by this work show that the optimized direct evaluation is more efficient than two stages Cox de Boor. In second place, for optimized direct evaluation computing normal vectors takes about 30-40% of the computing effort. This result led us to test approximated normal methods and, as shown, achieved results confirmed our expectations: it is faster to substitute the exact computation of normal vectors by their approximation.

Comparing NURBS and Bezier representations, optimized direct evaluation with approximated normal vectors computation is more efficient than all Bezier representation based evaluation methods.

Our results confirm the results provided by Sederberg: Horner's scheme is more efficient than deCasteljau and modified deCasteljau algorithms. Even more, we show that the version of this algorithm that uses approximated normal vectors is more efficient than the version that evaluates them exactly. However, comparing with NURBS representation, it is not faster than optimized direct evaluation with approximated normal vectors.

5. CONCLUSIONS

The following points summarize the presented work:

- An extension of the classical method of approximating normal vectors to solve edge discontinuities is provided.
- Five evaluation methods have been implemented and analyzed. Two versions that use approximated normal vectors calculation are added to the comparative.
- The computational efficiency of the methods has been compared. Some obtained results confirm results from the bibliography. Our original results enrich the comparative.
- The approximated method is more efficient than the exact one.

Analyzed methods allow generating the surface tessellation evaluating the surface for each parametric coordinate. The comparison can be enriched adding recursive summation based evaluation methods like forward differencing [Bar95a] or Silbermann's efficient implementation [Sil90a].

6. ACKNOWLEDGMENTS

This work has been carried out with the help of a Researching Personal Training Grant from Basque Government to Héctor Sánchez and CICYT grant TIC99-0252. Thanks to Alpha 1 System (Utah teapot and Goblet) and Kenny Hoff III (Lion and Ride) for the models. Héctor Sánchez also wish to thank Prof. Subodh Kumar and Jatin Chhugani for the time spent discussing fast tessellation algorithms during his stay at Johns Hopkins University. We also thank Michael Guthe from Institute of Computer Science – University of Bonn for his suggestions.

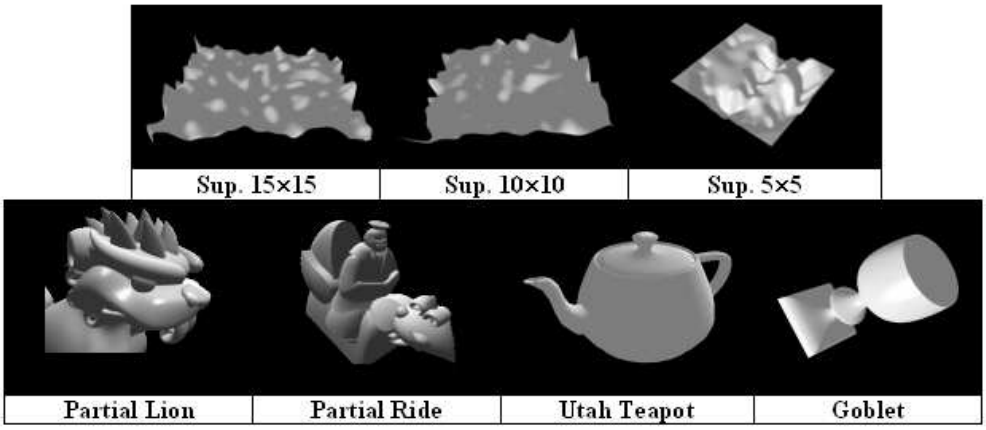

Figure 4: Tested models.

Model	Number of vertices	NURBS representation				Bezier representation		Power basis representation	
		Optimized direct evaluation			*Two-stages Cox de Boor*	deCasteljau	Modified deCasteljau	Horner's scheme	
		Exact normals	Without normals	Approximated normals (w. area)				Exact Normals	Approximated normals (w. area)
Sup.15x15	5K	1.9	1.07	1.12	3.04	14.95	11.98	3.17	2.98
Sup.10x10	9K	1.9	1.07	1.16	2.78	10.7	9.12	3.37	3.18
Sup.5x5	10K	1.04	0.6	0.82	2.03	3.15	3.18	2.06	1.86
Partial Lion	4K	0.33	0.2	0.25	0.39	0.37	0.38	0.33	0.3
Partial Ride	3,7K	0.3	0.21	0.24	0.36	0.3	0.31	0.27	0.25
Utah Teapot	5,5K	0.46	0.33	0.38	0.56	0.53	0.55	0.44	0.4
Goblet	40K	2.80	1.72	2.18	3.58	----	3.00	2.55	2.31

Table 5: Computational times (in seconds) for each surface evaluation method.

7. REFERENCES

[Bar95a] R. H. Bartels, J. C. Beatty and B. A. Barsky. "An Introduction to Splines for Use in Computer Graphics and Geometric Modelling" Morgan Kaufmann. September 1995.

[Bed91a] R. Bedichek, C. Ebeling, G. Winkenbach and A. D. DeRose "Rapid low-cost display of spline surfaces", Advanced Research in VLSI: Proceedings of the 1991 UCSC Conference, MIT Press, Cambridge MA, pp. 340-355, March 1991.

[Bli78a] J.F. Blinn. "A scanline algorithm for computer display of curves surfaces". Proceedings of the 5th annual conference on Computer graphics and interactive techniques, 1978.

[Boo00a] M. Boó, J. D. Bruguera and E. L. Zapata "Parallel architecture for the computation of NURBS surfaces". SPIE (The International Society for Optical Engineering) Media Processors 2000, pp. 37-48, 2000.

[Car95a] J. Carrière, "Evaluating Tensor Product and Triangular Bezier Surfaces" Research Report CS-95-22, Computer Science Department, University of Waterloo, Ontario, May 1995.

[Cha89a] S. Chang, M. Shantz and R. Rocchetti. "Rendering Cubic Curves and Surfaces with Integer Adaptive Forward Differencing" Computer Graphics (SIGGRAPH), vol. 23, nº 3, pp. 157-166, Jul. 1989.

[Chh01a] J. Chhugani and S. Kumar "View-Dependent Adaptive Tessellation of Parametric Surfaces", ACM Symposium on 3D Interactive Graphics, pp 59-62, 2001.

[Chh03a] J. Chhugani and S. Kumar. "Budget Based Sampling of Parametric Surfaces" To appear in ACM 3D Interactive Graphics, 2003.

[deC86a] P. de Casteljau, "Shape Mathematics and CAD" London: Kogan Page, 1986.

[Elb96a] G. Elber and E. Cohen, "Adaptive isocurve-based rendering for freeform surfaces",

ACM Transactions on Graphics, vol. 15, nº 3, pp. 249-263, 1996.

[Far01a] G. Farin. “Curves and Surfaces for CAGD”, Morgan-Kaufmann. 5th edition, October 2001.

[Far91a] R. Farouki. “On the stability of transformations between power and Bernstein polynomial forms”, Computer Aided Geometric Design, vol. 8, n. 1, pp. 29-36, 1991.

[Gop97a] M. Gopi and S. Manohar, “A Unified Architecture for the Computation of B-Spline Curves and Surfaces”, IEEE Transactions on Parallel and Distributed Systems, vol. 8, nº. 12, pp. 1275-1287, 1997.

[Gut02] M. Guthe, J. Meseth and R. Klein “Fast and Memory Efficient View-Dependent Trimmed NURBS Rendering”, Proceedings of Pacific Graphics 2002, IEEE Computer Society, pp. 204-213, October 2002.

[Her87a] B. von Herzen and A.R. Barr “Accurate Triangulations of deformed, intersecting surfaces”. Computer Graphics (SIGGRAPH ’87 Proceedings) vol. 21, pp. 103-110, July 1987.

[Kaj82a] J.T. Kajiya, “Ray Tracing Parametric Patches”, Computer Graphics, vol. 16, pp. 245-254, July 1982.

[Kum96a] S. Kumar “Interactive Display of Parametric Spline Surfaces”, PhD Thesis, University of North Carolina, 1996.

[Lan80a] J.M. Lane, L.C. Carpenter, J.T. Whitted and J.F. Blinn. “Scan line methods for displaying parametrically defined surfaces” Communications of ACM, vol. 23, nº. 1, pp. 23-34, 1980.

[Luk93a] W.L. Luken and F. Cheng “Rendering trimmed NURBS surfaces” Technical Report 18669(81711), IBM Research Division, 1993.

[Luk96a] W. L. Luken and F. Cheng “Comparison of Surface and Derivative Evaluation Methods for the Rendering of NURB Surfaces”, ACM Transactions on Graphics, vol. 15, nº. 2, pp. 153-178, 1996.

[Man95a] S. Mann and T. DeRose, “Computing Values and Derivatives of Bezier and B-spline Tensor Products” Computer Aided Geometric Design, vol, 12, nº 1, February 1995.

[Mar00a] W. Martin, E. Cohen, R. Fish and P. Shirley “Practical Ray Tracing of Trimmed NURBS Surfaces", Journal of Graphics Tools vol. 5, nº. 1, pp. 27-52, 2000.

[Max99a] N. Max “Weights for computing vertex normals from facet normals” Journal of Graphics Tools, vol. 4, nº. 2, pp.1-6, 1999.

[Pie97a] L. Piegl and W. Tiller. “The NURBS Book. Monographs in Visual Communication. Springer”, 2ª edición, 1997.

[Pie98a] L. Piegl and W. Tiller, “Geometry-based triangulation of trimmed NURBS surfaces” Computer Aided Design, vol. 30, nº 1, pp. 11-18, 1998.

[Qin97a] K. Qin, M. Gong, Y. Guan and W. Wang, “A New Method for Speeding Up Ray Tracing NURBS Surfaces”, Computers & Graphics, vol. 21, nº. 5, pp. 577-586, September-October, 1997.

[Roc87a] A. Rockwood, “A Generalized Scanning Technique for Display of Parametrically Defined Surfaces” IEEE Computer Graphics and its Applications, vol. 7, nº. 8, pp. 15-26, August 1987.

[Roc89a] A. Rockwood, K. Heaton and T. Davis “Realtime rendering of trimmed surfaces”, ACM Computer Graphics (SIGGRAPH Proceedings), vol. 23, nº. 3, 1989.

[Sed95a] T.W. Sederberg. “Point and Tangent Computation of Tensor Product Rational Bezier Surfaces”, Computer Aided Geometric Design, vol. 12, pp. 103-106, 1995.

[Sil90a] M.J. Silberman. “High Speed Implementation of Nonuniform Rational B-splines (NURBS)”, SPIE (The International Society for Optical Engineering) Curves and Surfaces in Computer Vision and Graphics, vol. 1251, pp. 338-345, 1990.

[Thü98a] G. Thürmer and C. A. Wüthrich “Computing vertex normals from polygonal facets” Journal of Graphics Tools, vol. 3, nº. 1, pp. 43-46, 1998.

[Whit78a] J.T. Whitted “A scanline algorithm for displaying parametrically defined surfaces” ACM Computer Graphics, vol. 12, nº. 3, pp 8-13, 1978.

[Whi80a] J.T. Whitted “An Improved Illumination Model for 3D Shaded Display” C.A.C.M. vol. 23, nº. 6, pp. 43-349, June 1980.