GPU-based Appearance Preserving Trimmed NURBS Rendering

Michael Guthe guthe@cs.uni-bonn.de

Ákos Balázs edhellon@cs.uni-bonn.de

Reinhard Klein rk@cs.uni-bonn.de

Universität Bonn
Institute of Computer Science II
Computer Graphics
Römerstraße 164
53117 Bonn, Germany

ABSTRACT

Trimmed NURBS are the standard surface representation used in CAD/CAM systems and accurate visualization of trimmed NURBS models at interactive frame rates is of great interest for industry. To support modification and/or animation of such surfaces, a GPU-based trimming and tessellation algorithm has been developed recently. First, the NURBS is approximated with a bi-cubic hierarchy of Bézier patches on the CPU and then these are tessellated on the GPU. Since this approach only took the geometric error of an approximation into account, the various illumination artifacts introduced by the chosen bi-cubic approximation and the subsequent tessellation were neglected. Although this problem could be solved partially by calculating exact per-pixel normals on the GPU, the shading error introduced due to the bi-cubic approximation would remain. Furthermore, the long fragment shader required for per-pixel normals would lead to unacceptably low performance.
In this paper we present a novel bi-cubic approximation algorithm that takes the normal approximation error into account. In addition, we also define a new error measure to calculate the required grid resolution for the bi-linear approximation. In combination, this allows GPU-based NURBS tessellation with guaranteed visual fidelity. Our new method is also capable of high quality visualization of further attributes like curvature, temperature, etc. on surfaces with little or no modification.

Keywords GPU-based algorithms, NURBS tessellation, appearance preservation

1 INTRODUCTION

CAD/CAM systems used in industry for the design of models for prototyping and production are usually based on trimmed NURBS surfaces, since they have the ability to describe almost every shape conveniently. Additionally, the NURBS representation is also used more and more frequently to generate animations in movies or even for computer games.
Especially in CAD, but also in the growing field of virtual prototyping the accurate, real-time visualization of these NURBS models together with additional information – like reflection lines visualizing the quality of the model – becomes more and more important. However, recently developed techniques for real-time trimming and tessellation on commodity GPUs like [GBK05] do not take the appearance of the surface into account and only control the geometric error (see Figure 1). Such negligence of the normals required for correct shading can lead to severe visual artifacts.

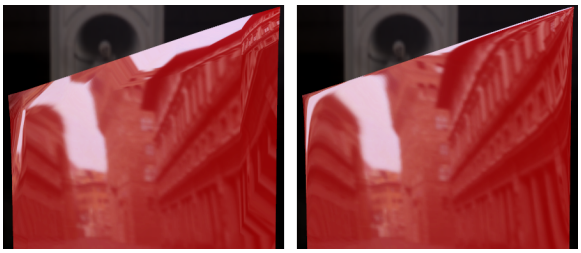

Figure 1: Difference between geometric (left) and appearance preserving (right) GPU-based tessellation.

Journal of WSCG, Vol.14, ISSN 1213-6972
WSCG'2006, January 30-February 3, 2006
Plzen, Czech Republic.

In this paper we present a novel GPU-based rendering algorithm that also takes shading artifacts into account, but requires only a slight overhead compared to the original GPU-based tessellation algorithm [GBK05].
Even though the introduced error measures were originally developed for the approximation of surface normals, they are also well suited for high quality visualization of various other surface properties or attributes, such as curvature, temperature distribution, or basically any surface information that can be represented using scalar values or vectors.

2 PREVIOUS WORK

As our new method exploits ideas of appearance preserving tessellation and GPU-based tessellation, we give a short overview of both fields. Since higher order surfaces cannot be evaluated on GPUs efficiently, GPU-based methods are restricted to bicubic surfaces and, as a result, have to use degree reduction methods. Therefore, we also review prior work in this field. Finally, we give a brief survey of the state of the art in the field of surface property visualization.

2.1 Appearance Preserving Tessellation

An approach for view-dependent refinement of multiresolution meshes was developed by Klein et al. [KSS98] which could theoretically also be used for the tessellation of trimmed NURBS models. However, since their error measure is highly dependent on the position of the highlight and derivatives are calculated in screen space, the exact position and orientation of the surface on the screen to be known. This makes a complete retessellation of the model necessary in each frame. Since only a small portion of the surfaces can be retessellated on the CPU per frame, this approach was modified by Guthe et al. [GBK04] to become view-independent. However, this method still suffers from the high latency and inflexibility of CPU-based tessellation.

2.2 GPU-based Tessellation

Abi-Ezzi et al. [AES94] and Bóo et al. [BAD+01] proposed an additional adaptive tessellation unit at the front of the rendering pipeline for NURBS and subdivision surfaces respectively. However, neither of these were built into commodity graphics hardware. Bolz and Schröder [BS03] developed an algorithm to evaluate Catmull-Clark subdivision surfaces on programmable graphics hardware. After the transmission of the tessellation textures to the GPU, only control points instead of triangles need to be send and thus the fragment shader can be saturated with marginal bus bandwidth consumption. With different tessellation textures this approach can also be used for bicubic B-Spline surfaces since they are equivalent to this subdivision scheme on a regular quad mesh. The algorithm generates an adaptive tessellation on a per-patch basis, which is rendered into an offscreen buffer – a so called pixel buffer or p-buffer – and then used as input for a second rendering pass. This method can achieve up to 30 million vertices per second on recent GPUs, but trimming of the surfaces is not possible. Based on this work, Kanai and Yasui [KY04] developed an algorithm to calculate accurate per-pixel normals on a tessellated subdivision surface. Although the produced images are very convincing, it is too slow for real time rendering at reasonable resolutions.
Recently a GPU-based trimming and tessellation algorithm for NURBS [GBK05] was developed. This method however, only takes the geometric error and not the shading error introduced due to incorrectly interpolated normals into account, which can lead to visual artifacts.

2.3 Degree Reduction

The idea of approximating high degree Bézier curves using degree reduction already came up more than 30 years ago [For72]. As shown by Park and Choi [PC95], the error can be reduced drastically by subdividing the curve before degree reduction. With a standard degree reduction algorithms, the degree of continuity between the composite curves cannot be controlled directly. Either, the continuity is preserved up to the maximum possible for the current curve degree (e.g. [For72]), or completely lost (e.g. [Eck93]). Therefore, Zheng and Wang [ZW03] developed a method to explicitly control the continuity classes of the curve at its endpoints. In [GBK05] a degree reduction method to preserve geometric continuity only has been presented. In contrast to all existing algorithms, which only consider the geometric error introduced by the degree reduction, the error measure proposed in this paper also takes the introduced shading error into account.

2.4 Surface Property Visualization

The rendering of surface properties is an important topic for surface interrogation and scientific visualization. Hagen et al. [HHS+92] give an overview of different surface interrogation methods, like orthonomics, reflection lines and focal surfaces. In the context of our work we only concentrate on reflection lines, since they can be visualized on the surface. In addition to these properties, the visualization of the curvature and curvature regions [EC93a, EC93b] also delivers valuable information for surface design. For visualization so called property surfaces are generated

in this approach. Since the calculation of these property surfaces is often computationally expensive, this method is not suited for complex or dynamic models.

3 GPU-BASED TESSELLATION

The overall workflow of the GPU-based trimming and tessellation algorithm [GBK05] is shown in Figure 2. First, the trimming curves are sampled with sufficient accuracy and evaluated on the GPU (2). Then the resulting polygons are rendered into a texture of appropriate size using a p-buffer (3). In the second rendering pass, the patch is sampled using a regular grid of sufficient resolution, such that a given geometric screen space error is guaranteed. For this purpose, predefined grids of different resolutions are stored on the graphics card in advance. At runtime, the grid index is calculated on the CPU and then sent to the GPU. Then the patch is evaluated at all grid vertices on the GPU (6). For the trimming, the trim-texture is simply bound and all pixels outside the trimming region are removed in the fragment stage by a lookup into this trim-texture (7).

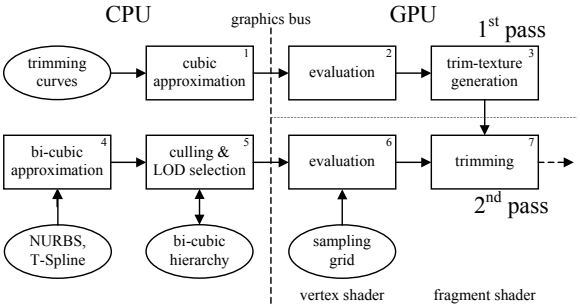

Figure 2: Main workflow of the GPU-based trimming and tessellation algorithm [GBK05].

As data dependent loops are only supported by very recent GPUs, a conversion from NURBS or T-Spline to piecewise rational Bézier representation is necessary, since the current knot spans, needed to calculate the sample points, differ. For cards not having texture access in the vertex shader, the amount of input data for a vertex program is limited to 16 vertex attributes and 8 program matrices and thus only low degree Bézier patches can be evaluated. To work with any graphics card supporting at least vertex shader 1.0, only 12 temporary registers can be used, which limits the maximum degree to bi-cubic. Therefore, the overall algorithm first approximates each NURBS or T-Spline surface and its trimming curves with a coarse hierarchy of rational bi-cubic Bézier patches, or cubic rational Bézier curves respectively, on the CPU (1+4). During rendering this hierarchy is traversed and patches with sufficient accuracy are selected to guarantee a given geometric screen space error (5). If the traversal reaches a leaf node, additional bi-cubic patches are generated. Then the control points of each patch are sent to the GPU before selecting a grid of appropriate resolution for evaluation.

An appearance – i.e. normal – preserving tessellation, based on this method, needs to preserve the normal in both approximation steps of the surface, namely the bi-cubic approximation on the CPU and the tessellation of the bi-cubic patch on the GPU. The remaining part of the algorithm however does not need to be changed. Therefore, the following two Sections describe only the modifications of the GPU-based NURBS rendering method necessary to preserve the appearance of the surfaces.

4 NORMAL PRESERVATION

When a Bézier surface S is approximated with a surface $\tilde{S}$, the visual approximation error on each point of the approximating surface $\tilde{S}(p)$, with the parameter value $p = (u, v)$, is the distance to the closest point on the original surface with the same color after shading, i.e. with the same normal $n = (n_x \ n_y \ n_z)^T$ when fragment based shading (e.g. Blinn-Phong or environment mapping) is used. This leads to the problem of finding a point $S(q)$ in the vicinity of $\tilde{S}(p)$, with $n(q) = \tilde{n}(p)$. Since we assume that S is smooth we can use a Taylor expansion of n around the parameter value p:

$$n(p+\Delta p) = n(p) + \begin{pmatrix} \frac{\partial n_x(p)}{\partial u} & \frac{\partial n_x(p)}{\partial v} \\ \frac{\partial n_y(p)}{\partial u} & \frac{\partial n_y(p)}{\partial v} \\ \frac{\partial n_z(p)}{\partial u} & \frac{\partial n_z(p)}{\partial v} \end{pmatrix} \Delta p + O(\|\Delta p\|^2)$$

Assuming $\|\Delta p\|$ to be small, a singular value decomposition could be used to find the smallest Δp and then $\|S(p + \Delta p) - \tilde{S}(p)\|$ is an upper bound for the visual error in object space. However, as shown in [GBK04], it is much more efficient to interpret the visual approximation error ε as the orthogonal combination of the geometric distance ε_{geom} of the point $\tilde{S}(p)$ on the approximating surface to the point $S(p)$ on the original surface and the distance ε_{norm} of $S(p)$ to the closest point $S(p + \Delta p)$ with the same normal $n(p + \Delta p) = \tilde{n}(p)$. As shown in Figure 3, these two distances can be combined by:

$$\varepsilon(p)^2 = \varepsilon_{geom}(p)^2 + \varepsilon_{norm}(p)^2.$$

In this case we are able to exploit the fact that the estimation of the geometric approximation error remains the same as for the non-appearance preserving tessellation. Thus the shading error can be estimated without actually calculating the position of the closest correctly shaded point in Euclidean space, using the approximate error measures deduced in the following.

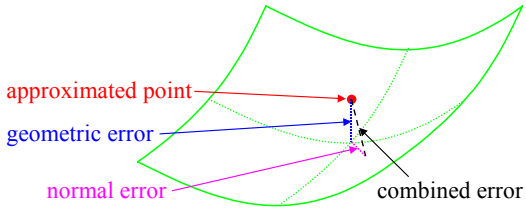

Figure 3: Combination of error measures.

4.1 Bi-cubic Approximation

When evaluating a Bézier surface, the normal is calculated as the cross-product of the first derivatives in u- and v-direction. This implies, that the normal on an approximating surface $\tilde{S}$ at parameter p equals that of the original surface S at parameter q, if

$$\frac{\partial \tilde{S}(p)}{\partial u} = \frac{\partial S(q)}{\partial u} \quad \text{and} \quad \frac{\partial \tilde{S}(p)}{\partial v} = \frac{\partial S(q)}{\partial v}.$$

Since the bi-cubic approximation of the arbitrary degree Bézier surface S with a bi-cubic Bézier surface $\tilde{S}$ is performed first in the u- and then in the v-direction, preserving the normal can be achieved by preserving the first derivative of each iso-parametric curve. The derivative approximation error ε_d when approximating a curve C with $\tilde{C}$ is then

$$\varepsilon_d(t) = \|C'(t) - \tilde{C}'(t)\|.$$

Since this error is defined in the space of the first derivative, it needs to be projected into object-space. This projection needs to map a distance δ_d in derivative space to a distance δ_o in object-space. Again we approximate the distances on the curves using a Taylor expansion around t:

$$\begin{aligned} \delta_o(t+\Delta t) &= \Delta t C'(t) + O(\|\Delta t\|^2) \\ \delta_d(t+\Delta t) &= \Delta t C''(t) + O(\|\Delta t\|^2) \end{aligned}$$

For a small Δt, we can approximate the projection with $\delta_o(t+\Delta t) \approx \Delta t C'(t)$ and $\delta_d(t+\Delta t) \approx \Delta t C''(t)$ such that the object-space derivative error $\varepsilon_{der}(t)$ between the two curves at parameter t is then

$$\varepsilon_{der}(t) \approx \left(\|C'(t) - \tilde{C}'(t)\| \frac{\|C'(t)\|}{\|C''(t)\|} \right).$$

The object-space derivative deviation error ε_{der} between the original and approximating curve is now defined as the maximum of $\varepsilon_{der}(t)$ along the curve:

$$\varepsilon_{der} \approx \sup_{0 \le t \le 1} \left(\|C'(t) - \tilde{C}'(t)\| \frac{\|C'(t)\|}{\|C''(t)\|} \right).$$

Arguing similarly to [GBK04], we can assume that the maximum derivative deviation error on a curve will probably occur at the point, where $\|C''(t)\|$ has its maximum and therefore the following approximation can be used without loss of visual fidelity:

$$\varepsilon_{der} \approx \sup_{0 \le t \le 1} \|C'(t) - \tilde{C}'(t)\| \frac{\sup_{0 \le t \le 1} \|C'(t)\|}{\sup_{0 \le t \le 1} \|C''(t)\|}.$$

4.2 Sampling

To generate less rendering primitives (e.g. for surfaces of revolution), the sampling resolution in u- and v-direction is separated as in [GBK05]. According to Filip et al. [FMM86], the error ε when approximating a C^2-continuous surface with a regular triangle mesh, where each pair of triangles spans the bilinear parameter space rectangle $D = [(u_i, v_j), (u_{i+1}, v_{j+1})]$ with the constant sizes $\Delta u = u_{i+1} - u_i$ and $\Delta v = v_{j+1} - v_j$ is bounded by

$$\varepsilon \le \frac{1}{8}(\Delta u^2 M_u + 2\Delta u \Delta v M_{uv} + \Delta v^2 M_v),$$

with

$$M_u = \sup_{p \in D} \left\| \frac{\partial^2 S}{\partial u^2} \right\|, \quad M_{uv} = \sup_{p \in D} \left\| \frac{\partial^2 S}{\partial u \partial v} \right\|,$$
$$\text{and } M_v = \sup_{p \in D} \left\| \frac{\partial^2 S}{\partial v^2} \right\|.$$

The sampling densities are then separated by exploiting the fact that $ab \le \frac{1}{2}(a^2 + b^2)$ and thus the approximation error is bound by

$$\varepsilon \le \frac{1}{8} \left(\Delta u^2 (M_u + M_{uv}) + \Delta v^2 (M_v + M_{uv}) \right),$$

which is a simple addition of the two approximation errors in u- and v-directions. This means, that ε is an upper bound for the approximation error, if the error in both directions is not greater than $\frac{\varepsilon}{2}$. This can further be simplified to calculating the piecewise linear approximation error of $n + m$ curves.
Following the estimations proposed in Section 4.1, we again assume that the maximum derivative error occurs at the point where $\|C''(t)\|$ has its maximum, which leads to the following approximate derivative deviation error:

$$\varepsilon_{der}(t) \approx \|C'(t) - \tilde{C}'(t)\| \frac{\sup_{0 \le t \le 1} \|C'(t)\|}{\sup_{0 \le t \le 1} \|C''(t)\|}.$$

Since $\varepsilon_{der}(t)$ is C^2 continuous, if C is C^3 continuous, which is the case for cubic Bézier curves, the theorem of Filip et al. [FMM86] gives an approximate upper bound ε_{der} of a piecewise linear approximation with a constant step size d of

$$\varepsilon_{der} \approx \frac{1}{8} d^2 \sup_{0 \le t \le 1} \|C'''(t)\| \frac{\sup_{0 \le t \le 1} \|C'(t)\|}{\sup_{0 \le t \le 1} \|C''(t)\|}.$$

The number of required samples n to achieve a maximum given deviation of ε is then:

$$n = \left\lceil \sqrt{\frac{\sqrt{E_{geom}^2 + E_{der}^2}}{8\varepsilon}} \right\rceil,$$

with

$$E_{geom} = \sup_{0\le t\le 1} \|C''(t)\|$$

$$E_{der} = \sup_{0\le t\le 1} \|C'''(t)\| \frac{\sup_{0\le t\le 1}\|C'(t)\|}{\sup_{0\le t\le 1}\|C''(t)\|}.$$

$C'''(t)$ can be written as a rational Bézier curve with a degree nine nominator $\check{P}(t) = \sum_{i=0}^{9} \check{P}_i B_i^9(t)$ and a degree twelve denominator $\check{w}(t) = \sum_{i=0}^{12} \check{w}_i B_i^{12}(t)$. Since all w_i are positive by construction, all $\check{w}_i$ are also positive. Therefore, an upper bound of the norm of the third derivative is given by:

$$\sup_{0\le t\le 1} \|C'''(t)\| \le \frac{\max(\|\check{P}_0\|, \ldots, \|\check{P}_9\|)}{\min(\check{w}_0, \ldots, \check{w}_{12})}.$$

The upper bounds for $\|C''(t)\|$ and $\|C'(t)\|$ are calculated as in [GBK05], i.e. $\|C''(t)\|$ from a degree seven/nine and $\|C'(t)\|$ from a degree five/six rational polynomial curve. Since the calculation of these upper bounds is only required when extending the bi-cubic hierarchy, the additional computation time can be expected to be marginal for static models.

5 OTHER ATTRIBUTES

The appearance preserving error measure derived in Section 4 is not limited to normals – which are preserved when preserving the first derivatives – but can easily be extended to higher derivatives or arbitrary attributes. If $A(u, v)$ is a general attribute defined as a tensor product, we can again reduce the problem into a piecewise curve representation and project the approximation error from attribute- to object-space with

$$\varepsilon_A(t) = \|C_A(t) - \tilde{C}_A(t)\| \frac{\|C'(t)\|}{\|C'_A(t)\|}.$$

Starting from this definition, the approximation error required for the bi-cubic approximation and the regular tessellation can be derived using the same assumptions and estimations as in Section 4. For the bi-cubic approximation, we then have

$$\varepsilon_A \approx \sup_{0\le t\le 1} \|C_A(t) - \tilde{C}_A(t)\| \frac{\sup_{0\le t\le 1}\|C'(t)\|}{\sup_{0\le t\le 1}\|C'_A(t)\|},$$

and for the sampling resolution

$$\varepsilon_A \approx \frac{1}{8d^2} \sup_{0\le t\le 1} \|C''_A(t)\| \frac{\sup_{0\le t\le 1}\|C'(t)\|}{\sup_{0\le t\le 1}\|C'_A(t)\|}.$$

Finally, the approximation errors of all attributes are combined with the geometric approximation error as an orthogonal combination of partial errors.

6 RESULTS

To evaluate the efficiency of our method, we first compare its performance with the previous GPU-based tessellation method, that only guarantees a geometric error. Then we examine the image quality improvements provided by our new method and finally we test its applicability in the field of surface property visualization, especially in comparison with the previous method.

6.1 Performance

All benchmarks were performed on an Athlon 64 3200+ with 1.5 GByte memory and a GeForce 7800 GTX at a resolution of 1280×1024 (unless noted otherwise) with 0.5 pixel screen space error.

First, we compare the tessellation performance of our method with the performance of the previous GPU-based algorithm using a single bi-cubic trimmed and untrimmed patch (see Figure 4). To analyze the tessellation performance we render this patches at different screen-sizes, where a larger screen-size implies a lower object-space error and a higher sampling rate.

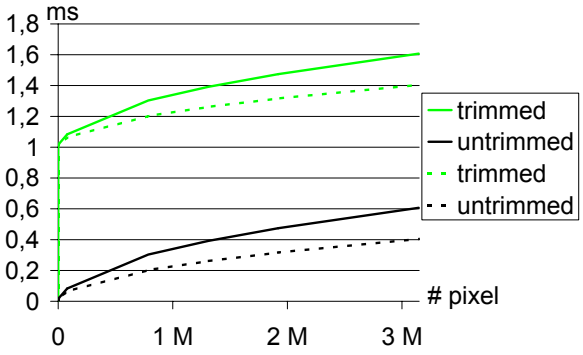

Figure 4: Tessellation performance in dependance of screen size for geometric (dashed) and appearance preserving GPU-based tessellation.

As shown by these graphs, the number of additional vertices and thus the additional rendering time for the untrimmed surface is approximately 50%. However, when the surface is very small on screen (a few pixel), the additional number of vertices and the performance loss is approaching zero. It can also be observed, that the trimming overhead remains constant, since the trimming itself does not alter the appearance of the surface and remained unchanged.

As second example, we compare the performance of the bi-cubic approximation for surfaces of different degrees with that of the original GPU-based tessellation.

In Figure 5 the performance of both methods is shown for a single animated trimmed NURBS surface with 100 control points and degrees of 3×3, 5×5, and 7×7 respectively.

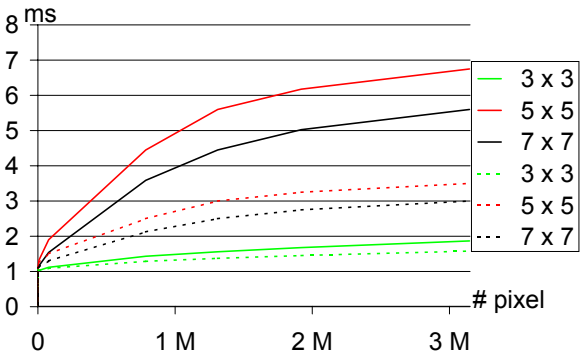

Figure 5: Total rendering performance of a single animated trimmed NURBS surface with 100 control points and of different degrees using geometric (dashed) and appearance preserving GPU-based tessellation.

For large surfaces of higher degree, where a bi-cubic approximation is required, the total rendering time increases by up to 93%. This performance drop is mainly due to the computationally more expensive error measure for the bi-cubic approximation, as the percentage of additional bi-cubic patches and rendered vertices is significantly lower than this. Since the bi-cubic approximation error measure needs to be calculated only when a surface is modified, the impact on static models will be significantly lower.

To evaluate the performance on more complex static models, resembling a real application setting, we render the industrial CAD models shown in the Figures 6-8.

Figure 6: Mini model: 629 trimmed surfaces.

Detailed statistics on the number of NURBS and underlying Bézier surfaces, as well as the number of non-trivially trimmed NURBS surfaces, of these models are given in Table 1.

Table 2 compares the average number of rendered bi-cubic Bézier patches, the average number of generated vertices and the frame-rate of the unmodified GPU-based trimming and tessellation algorithm [GBK05] with the appearance preserving method presented in this paper.

Figure 7: Golf model: 8,138 trimmed surfaces.

Figure 8: C-Class model: 67,571 trimmed surfaces.

	Mini	Golf	C-Class
NURBS surfaces	629	8,138	67,571
non-trivially trimmed	203	1,486	35,230
Bézier patches	25,648	17,936	396,535

Table 1: Details of the models used for evaluation.

	Mini	Golf	C-Class
geometric only approximation			
bi-cubic patches	11,683	8,008	105,442
vertices	210,529	239,221	2,216,352
frame-rate	12.8 fps	9.1 fps	1.3 fps
appearance preserving approximation			
bi-cubic patches	11,685	8,017	159,176
vertices	283,333	280,400	2,538,128
frame-rate	11.0 fps	7.9 fps	1.2 fps

Table 2: Performance comparison between geometric and appearance preserving approximation.

Even though the number of bi-cubic patches has increased by 21% to 51% and the number of vertices increased by 14% to 39%, the frame-rate difference is only between 8% and 14%. The very loose coupling between the number of vertices and the rendering performance is mainly due to the massively parallel architecture of modern GPUs. With n parallel vertex units,

the evaluation and transformation time t of each bicubic Bézier patch with v vertices is

$$t = c \left\lceil \frac{v}{n} \right\rceil ,$$

where c depends on the GPU performance. Note, that for these models, the average number of vertices per Bézier patch (16 to 30) is in the same order of magnitude as the number of parallel vertex units in current GPUs, even for the appearance preserving tessellation. In addition to this, each Bézier patch requires a constant time for initialization of the vertex array, uploading of the control points for evaluation, and setting the domain interval for trimming, regardless of the size of the regular grid used for evaluation. Furthermore, the time required for trimming remains constant as well.

6.2 Image Quality

In order to compare with the previous, purely geometric approach, we perform a pixel by pixel comparison of the interpolated normals with the real normals from the NURBS model obtained via sub-pixel subdivision. The visual difference between the real and interpolated normals (shown in Figure 9) can be extracted using simple image processing. To estimate the normal deviation in screen space, the surrounding pixels are used to calculate the normal derivatives. Note, that this is only correct on a closed surface but not along contours.

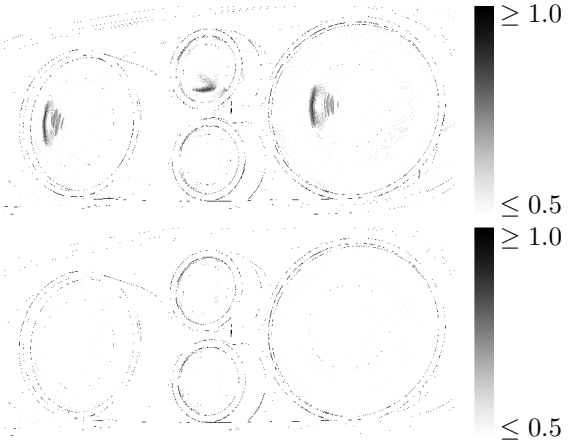

Figure 9: Normal deviation error in pixels for a closeup of the Golf model using GPU-based tessellation without (top) and with normal preserving error measure (bottom).

It is clearly visible that the normal approximation is much better when using the appearance preserving tessellation. Note, that the remaining pixels along contours, where the normal error exceeds the 0.5 pixel threshold are – as already mentioned – due to the normal undersampling in the image processing step and not a shortcoming of the appearance preserving tessellation algorithm.

In addition to the visual comparison, Table 3 compares the average normal deviation of our approach with the previous GPU-based tessellation algorithm that controls the geometric error only.

	angle	pixel	exceeded
geometric only	$0.930°$	0.117	2.409%
normal preserving	$0.752°$	0.074	1.238%

Table 3: Average normal approximation error per foreground pixel and percentage of foreground pixels where desired error is exceeded.

Here again, the remaining pixels where the screen space error threshold is exceeded are located along contours and are therefore the results of aliasing artifacts and not due to incorrect normals.

6.3 Surface Properties

The main goal of surface property visualization in industry, especially in design, is to ensure the continuity of reflections on the surface. For this purpose, so-called reflection lines are mainly used. Figure 10 compares the reflections lines rendered with a grid environment on a closeup of the Golf fender using geometric and appearance preserving tessellation.

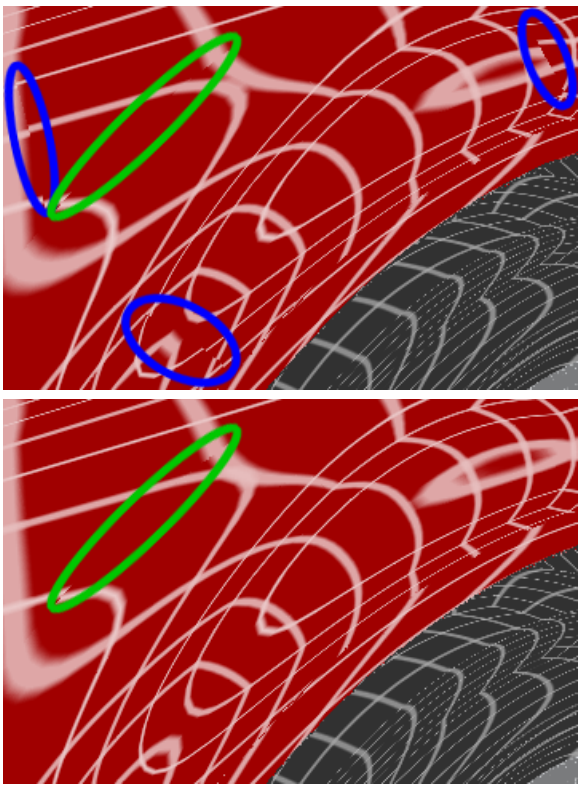

Figure 10: Reflection lines using geometric (top) and appearance preserving (bottom) tessellation. In the top image, the real discontinuity (green) is indistinguishable from tessellation related (blue).

To identify discontinuities in the shading, which occur at ridges or ravines of the model, the tessellation needs to produce meshes with correct normal interpolation. Even a slight normal deviation of a few degree can lead to visual artifacts that are indistinguishable from real surface discontinuities. Using the appearance preserving bi-cubic approximation and tessellation presented in this paper, the normals are correct within a given screen space error and thus shading discontinuities only occur when they are present in the model.

7 CONCLUSION

In this work, we presented a novel method for GPU-based appearance preserving tessellation of NURBS surfaces. We demonstrated the problems of the previous algorithm in dealing with various illumination/shading artifacts introduced by the bi-cubic approximation and the following tessellation. We also demonstrated that our algorithm only requires a relatively low number of additional bi-cubic patches and vertices to produce accurate interpolated normals. It achieves almost the same performance as the original method, but nevertheless provides a much higher visual fidelity. Our new method also has the capability to visualize surface properties such as degree of continuity or discontinuities using reflection lines. Due to the real-time trimming and tessellation on the GPU, it is also suitable for the visualization of deformable models and to have immediate feedback during the design and virtual prototyping process.

8 ACKNOWLEDGEMENTS

We would like to thank SGI, DaimlerChrysler AG, and Volkswagen AG for providing for the trimmed NURBS models used in this paper. This work was partially funded by the European Union under the project of “Real Reflect” (IST-2001-34744).

References

[AES94] S. S. Abi-Ezzi and S. Subramanian. Fast dynamic tessellation of trimmed NURBS surfaces. *Computer Graphics Forum*, No.13(3), pp.107–126, 1994.

[BAD+01] M. Bóo, M. Amor, M. Doggett, J. Hirche, and W. Straßer. Hardware support for adaptive subdivision surface rendering. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pp.33–40, 2001.

[BS03] J. Bolz and P. Schröder. Evaluation of subdivision surfaces on programmable graphics hardware, 2003.

[EC93a] G. Elber and E. Cohen. Hybrid symbolic and numeric operators as tools for analysis of freeform surfaces. In *Working Conference on Geometric Modeling in Computer Graphics*, pp.275–286, 1993.

[EC93b] G. Elber and E. Cohen. Second-order surface analysis using hybrid symbolic and numeric operators. *ACM Transactions on Graphics*, No.12(2), pp.160–178, 1993.

[Eck93] M. Eck. Degree reduction of Bézier curves. *Computer Aided Geometric Design*, No.10(3-4), pp.237–252, 1993.

[FMM86] D. Filip, R. Magedson, and R. Markot. Surface algorithms using bounds on derivatives. *Computer Aided Geometric Design*, No.3(4), pp.295–311, 1986.

[For72] A. Forrest. Interactive interpolation and approximation by Bézier polynomials. *The Computer Journal*, No.15(1), pp.71–79, 1972.

[GBK04] M. Guthe, Á. Balázs, and R. Klein. Interactive High Quality Trimmed NURBS Visualization Using Appearance Preserving Tessellation. In *Data Visualization 2004 (Proceedings of TCVG Symposium on Visualization)*, pp.211–220 + 348. EUROGRAPHICS - IEEE, May 2004.

[GBK05] M. Guthe, Á. Balázs, and R. Klein. GPU-based trimming and tessellation of NURBS and T-Spline surfaces. *ACM Transactions on Graphics*, No.24(3), pp.1016–1023, 2005.

[HHS+92] H. Hagen, S. Hahmann, T. Schreiber, Y. Nakajima, B. Wördenweber, and P. Hollemann-Grundstedt. Surface interrogation algorithms. In *IEEE Visualization and Computer Graphics*, pp.53–60, 1992.

[KSS98] R. Klein, A. Schilling, and W. Straßer. Illumination dependent refinement of multiresolution meshes. In *Proceedings of Computer Graphics International (CGI ’98)*, pp.680–687, Los Alamitos, CA, 1998. IEEE Computer Society Press.

[KY04] T. Kanai and Y. Yasui. Per-pixel evaluation of parametric surfaces on GPU. In *ACM Workshop on General Purpose Computing Using Graphics Processors (also at SIGGRAPH 2004 poster session)*, August 2004.

[PC95] Y. Park and U J. Choi. Degree reduction of Bézier curves and its error analysis. *J. Austral. Math. Soc. Ser. B*, No.36, pp.399–413, 1995.

[ZW03] J. Zheng and G. Wang. Perturbing Bézier coefficients for best constrained degree reduction in the L_2-norm. *Graphical Models*, No.65, pp.351–368, 2003.