Joint EUROGRAPHICS - IEEE TCVG Symposium on Visualization (2004)
O. Deussen, C. Hansen, D.A. Keim, D. Saupe (Editors)

Interactive High Quality Trimmed NURBS Visualization Using Appearance Preserving Tessellation

M. Guthe[1], Á. Balázs[1] and R. Klein[1]

[1] Institute of Computer Science II, Computer Graphics, Bonn University, Bonn, Germany

Abstract

Trimmed NURBS models are the standard representation used in CAD/CAM systems and accurate visualization of large trimmed NURBS models at interactive frame rates is of great interest for industry. To visualize the quality of a surface several techniques like isophotes, reflection lines, etc. are used. Most existing approaches transform the NURBS surfaces into a fine polygonal representation and build static levels of detail from this representation. This polygonal approximation together with its normals are adjusted in a semi-automatic procedure to achieve the desired visual fidelity during visualization. Since this approach allows only for a fixed maximum accuracy and does not support deformable models, another more recent approach is to keep the NURBS representation and generate view-dependent LODs on the fly up to the currently required preciseness.
However, so far this approach took only into account the geometric error of an approximation and thus neglected the various illumination artifacts introduced by the chosen (possibly view-dependent) triangulation. Although this problem can be solved by using normal maps, the resolution of the normal maps again limits the accuracy. Furthermore, the normal map generation requires a preprocessing step which prevents the support of deformable NURBS models. In this paper we present a novel automatic tessellation algorithm that considers the illumination artifacts and is well suited both for the generation of static and dynamic LOD schemes 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.

Categories and Subject Descriptors (according to ACM CCS): H.5.1 [Information Interfaces and Presentation]: Animations I.3.3 [Computer Graphics]: Display Algorithms, I.3.5 [Computer Graphics]: Curve, Surface, Solid, and Object Representations J.2 [Physical Sciences and Engineering]: Engineering

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. Consequently, the NURBS representation is also used to generate animations in movies or for computer games.

Especially in CAD, but also in the growing field of virtual prototyping accurate real-time visualization of these NURBS models together with additional information visualizing the quality of the model like isophotes and reflection lines becomes more and more important. Since even today's advanced graphics hardware is unable to directly render trimmed NURBS surfaces, they need to be transformed into a suitable (e.g. polygonal) representation. This process is usually referred to as "tessellation".

The two main approaches for real-time visualization of trimmed NURBS models are based on either generating a sufficiently fine tessellation and build static levels of detail from it, or on generating the required tessellations on the fly. Since complex models may consist of several thousand trimmed NURBS patches, a fast tessellation algorithm that produces few triangles is required for both approaches. Therefore previous algorithms only considered the geometric distance between the polygonal representation and the original surface for the tessellation. Additionally normal

maps were used to reduce shading artifacts as well as artifacts in subsequent visualization techniques like isophotes and reflection lines. This technique can be applied to both static levels of detail and on the fly tessellation.

For sophisticated modelling applications (e.g. mirror design, creation of smooth objects, etc.) it is beneficial to interactively visualize deformable trimmed NURBS models with reflections or other surface attributes. For such applications however, normal maps cannot be used, since they require a preprocessing step. To overcome this limitation we present a tessellation algorithm that is capable of generating an appearance preserving tessellation, that does not use normal maps and requires no preprocessing at all.

While our new method was originally developed for the approximation of surface normals, it is also well suited for high quality visualization of various other surface properties, such as curvature, temperature distribution, or basically any surface information that can be represented using scalar values or vectors.

The main advantages of our new algorithm are:

- Support of interactive appearance preserving visualization of deformable trimmed NURBS models since no preprocessing is required.
- Furthermore, interactive, high quality information visualization of various surface properties like curvature or temperature is supported.
- The additional number of vertices/triangles compared to previous tessellation algorithms is marginal.
- The frame rate of our method is higher compared to previous techniques using textures or normal maps.

2. Previous Work

Since we combine trimmed NURBS tessellation with appearance preserving levels of detail in our work, we give a short overview of both fields. Then we take a short look at the field of surface property visualization.

2.1. Trimmed NURBS Tessellation

Researchers have put a lot of effort into the visualization of trimmed NURBS surfaces due to its industrial relevance. Different approaches emerged for visualization, e.g. ray-tracing the surfaces (e.g. [NSK90]), pixel level subdivision (e.g. [SC88]), or polygon tessellation (e.g. [HB87, RHD89, FK90]), of which the triangle based methods are generally much faster due to recent advances in graphics hardware. On a multiprocessor system these triangulated models can be rendered at interactive rates [BSGM02], but this requires massive amounts of memory for storing the hierarchical static levels of detail, since every vertex of the finest triangulation needs approximately 65 bytes of memory (including vertex normals) [FMEP02].

While these first approaches dealt with individual curves or surfaces and usually made little or no attempt to overcome the problems caused by individual tessellation of patches, more recent approaches are able to render trimmed NURBS at interactive frame rates by combining several patches to so-called super-surfaces. An example for this group of algorithms is the work of Kumar et al. [KMZH97], which introduces the notion of supersurfaces. Based on a priori known connectivity information sets of trimmed NURBS patches are clustered into so-called super-surfaces. An individual view-dependent triangulation is generated at run-time for each super-surface and in a final step these view-dependent triangulations are sewn together in order to avoid cracks. The computationally complex sewing part is parallelized to achieve real-time frame rates even for huge models. Another approach [KSSP01] only deals with very specific configurations of trimmed NURBS surfaces that are stacked on top of each other. [GMK02] shifted the complex sewing part to a preprocessing step and introduced the seam graph, which consists of all trimming curves contained in a model and manages all connectivity information between the individual LODs. At runtime a view-dependent LOD of the seam graph is generated in each frame and the individual patches in the model are re-tesselated according to the LOD of the seam graph.

All these methods have the common disadvantages that they either rely on connection information to be supplied, and/or require significant preprocessing time. Therefore, the connectivity information between patches cannot be changed at runtime, which makes these algorithms unsuitable for deformable models or models with dynamic neighborhood relations. To overcome this limitation another approach was presented by Balázs et al. [BGK04b], where the cracks resulting from the independent tessellation of neighboring patches are closed by extruding the patch boundaries. Since this extrusion only depends on the corresponding patch, no neighborhood information is required and therefore, the method supports deformable models.

2.2. Appearance Preserving LOD

In the field of appearance preserving levels of detail, two main approaches exist. The first approach is to use textures, the so called normal maps [COM98], to store the information required for correct shading. These normal maps can be used for efficient shading in software or on programmable graphics hardware [TCRS00]. Recently Cole applied normal maps to view-dependent levels of detail [Col01] and showed their efficiency. Using normal maps dramatically reduces popping artifacts due to incorrect shading, but generating a normal map texture with a fixed size for every patch like [COM98] usually needs too much memory.

Therefore, some approaches to compress textures on polygonal models without loss of quality have been proposed in the recent years. Sloan et al. [SWB98] generate an importance map for a given 2D parametrization and warp the

square texture to evenly distribute this scalar field. Another approach to generate an optimized texture map [TV91] uses a dynamic simulation, where grid edge weights are set according to local image content. This method was extended to 3D surfaces by Balmelli et al. [BTB02] and also by Sander et al. [SGSH02] using a pre-integrated signal stretch metric. They have proven to dramatically reduce the texture size compared to a non-specialized parametrization without loss of quality. This method was modified for trimmed NURBS models in [GK03].

Due to the preprocessing required, such methods based on the normal maps are not applicable to deformable models. Therefore, an appearance preserving LOD scheme is required. Garland et al. modified their error quadrics [GH97] to preserve color, texture coordinates and normals [GH98] aswell. However, due to the overestimation of the quadric error metric, an efficient simplification to a specified error is difficult. Another approach for view-dependent refinement of multiresolution meshes was developed by Klein et al. [KSS98]. This approach is also applicable to trimmed NURBS models, but their error measure requires the exact position and orientation of the surface on the screen to be known. Their error measure is also highly dependent on the position of the highlight, and the derivatives are calculated in screen-space so a complete retessellation of almost all surfaces is necessary in each frame. Since all interactive NURBS visualization systems rely on the fact that only a small portion of the surfaces need to be retessellated per frame, this error measure is not directly applicable.

2.3. Surface Property Visualization

The display 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, isophotes, reflection lines and focal surfaces. In the context of our work we only concentrate on isophotes and reflection lines, since they can be visualized on the surface. Additionally to these properties, the visualization of the curvature and curvature regions [EC93b, EC93a] deliver valuable information for surface design. For visualization so called property surfaces are generated in this approach. However, the calculation and rendering of these property surfaces are often computationally expensive and therefore, this method is not well suited for complex and/or dynamic models.

Another important surface property for CAD and virtual prototyping is the surface temperature generated by finite element simulations [KSZ*96]. This method however relies on a fine enough polygonal mesh representation for visualization, but for complex models, a static tessellation which is independent of the current temperature distribution quickly becomes too large for interactive visualization. More recently van Wijk [vW03] employed flow visualization techniques to surfaces based on triangular meshes in order to enhance the visualization of the shape and features of such models. As the root of this approach lies in flow visualization, it is more geared towards the visualization of time-varying data on static models, while our method is rather suited to the visualization of properties of deformable parametric surfaces.

3. Trimmed NURBS Rendering Framework

Our appearance preserving tessellation can be applied for both static and dynamic LOD schemes. To support deformable and dynamic NURBS models which require dynamic LODs, we base our appearance preserving tessellation on a view-dependent dynamic LOD approach. Since dynamic models may also have a dynamically changing neighborhood, we cannot rely on any connectivity information and therefore, we tessellate all trimmed NURBS surfaces independently. This independent tessellation can lead to gaps between neighboring NURBS surfaces, which we close using the Fat Borders method [BGK04b]. The artifacts introduced by this algorithm are independent from the tessellation method as long as the geometric error along the trimming stays the same which is the case with our tessellation methods.

3.1. Tessellation

Traditional runtime tessellation algorithms either use a grid or a quadtree to subdivide the surface for approximation. Since even the quadtree is not completely adaptive, we have developed a new approximation algorithm based on kd-tree subdivision [BGK04a]. The approximation error for the current subdivision can be calculated using the distance between the control points and the bilinear surface approximation. Since the two triangles that would be generated for this tree node cannot resemble a bilinear quad patch, an additional approximation error needs to be taken into account which leads to the following estimated error [KBK02]:

$$\varepsilon_{conservative} \leq \varepsilon_{bilin} + \frac{1}{4}\|P_{00} - P_{m0} + P_{0n} - P_{mn}\|, \text{ with}$$

$$\varepsilon_{bilin} \leq \max_{i=0,j=0}^{i\leq m,j\leq n} \|P_{ij} - \tilde{S}(\frac{i}{m}, \frac{j}{n})\|, \text{ where}$$

$$\tilde{S}(a,b) = (1-b)((1-a)P_{00} + aP_{0n}) + b((1-a)P_{m0} + aP_{mn}).$$

Since this error measure is still a (sometimes significant) overestimation, an approximate error measure can also be used if the approximation inside a patch has not to be guaranteed. In order to calculate this approximate error we still use the above equations, but replace the control point P_{ij} with $S(\alpha_i, \beta_j)$ where α_i and β_j are the parameter values corresponding to the control point P_{ij}.

If the estimated approximation error exceeds the desired error for the NURBS surface the tree node must be subdivided. If a quadtree is used, the node is split at the midpoint

in the parameter domain. On surfaces with high curvature in one direction of the parameter domain and low curvature in the other direction (e.g. a cylindrical surface) this leads to an unnecessarily high subdivision in the low curvature direction. Although using a binary subdivision scheme solves this, it still suffers from the problem that unnecessary subdivisions are applied if the curvature of the surface is highly variant.

This can be solved by using a more general binary subdivision of the surface. Since a NURBS surface can only be subdivided either in the u or in the v direction, this leads to a kd-tree subdivision. We subdivide at $(\frac{k}{m}, \frac{l}{n})$ for which the following equation holds:

$$\|S(\frac{k}{m},\frac{l}{n}) - \tilde{S}(\frac{k}{m},\frac{l}{n})\| = \max_{i=0,j=0}^{i\leq m,j\leq n} \|P_{ij} - \tilde{S}(\frac{i}{m},\frac{j}{n})\|,$$

where $0 \leq k \leq m$, $0 \leq l \leq n$ and $\tilde{S}$ is the bilinear approximation of S. As the direction of the subdivision we choose the one for which the line subdividing the kd-tree node is closer to $S(\frac{k}{m}, \frac{l}{n})$.

4. Appearance Preserving Tessellation

For appearance preserving tessellation, the geometric distance (the Hausdorff error [KLS96]) between the approximated and the original surface is not sufficient. Instead of this error, the distance between a point on the approximated surface and the closest point on the original surface of the same color (i.e. normal) has to be measured. This distance is usually higher that the Hausdorff distance and much harder to calculate.

Since previous NURBS tessellation algorithms used an estimation of the geometric distance as error measure for approximation, this has to be modified accordingly.

4.1. Modified Error Measure

Since the generated tessellation has to be independent of the exact position of the viewer, we assume that the Blinn-Phong shading model or any other algorithm using normal vector interpolation is used.

Because the tessellation algorithm approximates the surface using bilinear quad patches, we need to estimate the maximum combined error over each quad patch. Since the tessellation algorithm already uses discrete points on the surface to estimate the geometric error, they provide a straightforward basis for the approximation of the shading error. Let us assume that the derivatives ($\vec{n_u}$ and $\vec{n_v}$) of the surface normal $\vec{n}$ are locally smooth around a sample point. This leads to the following problem:

$$\vec{n}'(\vec{d}) = \vec{n} + \begin{pmatrix} \vec{n_u}_x & \vec{n_v}_x \\ \vec{n_u}_y & \vec{n_v}_y \\ \vec{n_u}_z & \vec{n_v}_z \end{pmatrix} \vec{d} \qquad (1)$$

$$\vec{n}_{bilin} \approx \frac{\vec{n}'(\vec{d})}{\|\vec{n}'(\vec{d})\|}, \qquad (2)$$

where $\vec{n}_{bilin}$ is the bilinear approximation of the normal on the quad patch and $\vec{d}$ is the offset of the next correctly shaded pixel in the two dimensional domain space. Since we assume $\vec{d}$ to be small, we can also assume the denominator to be one. Therefore a singular value decomposition can be used to find the smallest $\vec{d}$. Then the position of the correctly shaded pixel can be calculated in Euclidian space and the distance between this point and the sample point on the bilinear quad patch delivers a good estimation of the combined error.

However, while this method is fairly straightforward and provides a relatively tight error bound, it suffers from high computational requirements. This is easy to see, as the method first involves calculating the sample point – as described for the tessellation algorithm – with partial derivatives, then the nearest correct pixel on the bilinear patch has to be found by solving Equation 1, and finally the surface has to be evaluated once more to calculate the distance between this new point and the original point on the bilinear patch in order to decide whether further subdivisions are necessary or not. In total the surface has to be evaluated twice, and an additional eigenvalue problem must be solved. Since this would be computationally too expensive for interactive visualization, we use the following further simplified version of the approximation method just introduced.

The combined approximation error ε can be viewed as the orthogonal combination of the geometric distance between the approximated patch and the surface and the distance between the surface pixel and the nearest correctly shaded surface pixel, as shown in Figure 1, these can be combined by:

$$\varepsilon^2 = \varepsilon^2_{geometric} + \varepsilon^2_{shading}.$$

In this case we are able to take advantage of the fact that the estimation of the geometric approximation error remains the same as for the non-appearance preserving tessellation and thus the shading error can be estimated without calculating the position of the closest correctly shaded point in Euclidean space.

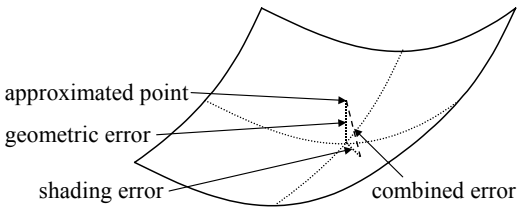

Figure 1: *Combination of error measures.*

For this estimation, we use the curvatures (c_u and c_v), which are defined for any point on the surface as the magnitude of the normal derivatives divided by the magnitude of

the surface derivatives (δ_u and δ_v):

$$c_u = \frac{\|\vec{n_u}\|}{\|\delta_u\|}, \quad c_v = \frac{\|\vec{n_v}\|}{\|\delta_v\|}.$$

For a curve, the normal deviation error can then be written as:

$$\varepsilon_{shading} \approx \frac{\|\vec{n} - \vec{n}_{lin}\|}{c}.$$

When transferring this approximation to a surface, a choice has to be made whether the minimum, maximum or mean curvature is used. The minimum curvature has the advantage that it is conservative, but if it becomes zero, even a slight normal deviation leads to further subdivisions, no matter how high the maximum curvature (and therefore how low the shading error) is. The mean curvature seems to be a good compromise at first sight, but since the normal deviation mainly occurs in the direction of the maximum curvature, the mean curvature can lead to an unnecessarily high number of subdivisions. Therefore, we choose the maximum curvature resulting in the following formula:

$$\varepsilon_{shading} \approx \frac{\|\vec{n} - \vec{n}_{bilin}\|}{\max(c_u, c_v)}.$$

To save the costly computation of normal derivatives for each sample point, we simplify our approach even further and only calculate the maximum curvature of the bilinear patch instead of the local curvatures for the sample points. This leads to:

$$\begin{aligned}
c_1 &= \frac{\|\vec{n}(u_{min}, v_{min}) - \vec{n}(u_{min}, v_{max})\|}{\|S(u_{min}, v_{min}) - S(u_{min}, v_{max})\|} \\
c_2 &= \frac{\|\vec{n}(u_{min}, v_{min}) - \vec{n}(u_{min}, v_{max})\|}{\|S(u_{min}, v_{min}) - S(u_{max}, v_{min})\|} \\
c_3 &= \frac{\|\vec{n}(u_{max}, v_{max}) - \vec{n}(u_{min}, v_{max})\|}{\|S(u_{min}, v_{min}) - S(u_{min}, v_{max})\|} \\
c_4 &= \frac{\|\vec{n}(u_{max}, v_{max}) - \vec{n}(u_{min}, v_{max})\|}{\|S(u_{min}, v_{min}) - S(u_{max}, v_{min})\|} \\
\varepsilon_{shading} &\approx \frac{\|\vec{n} - \vec{n}_{bilin}\|}{\max(c_1, c_2, c_3, c_4)},
\end{aligned}$$

where $\vec{n}(u,v)$ is the normal of the surface S at (u,v). When using this method to estimate the shading error, the surface has to be evaluated for only once each sample point on the surface. Therefore, the only additional calculation required per sample for the appearance preserving tessellation is the calculation of the vertex normal which needs little extra computation time.

The applicability of this error measure is not limited to surface normals, it can be employed to accurately visualize any other surface property as well for example, temperature distribution, or curvature. The only modification is that instead of – or additionally to – the normal the deviation of these attributes have to be taken into account. For vector data, the error estimation is identical to the estimation of the normal error and for scalar values the norm of the vector difference is simply replaced by the absolute difference of the scalar values.

5. Shading

As already mentioned we assume a shading model using normal vector interpolation. On current graphics hardware the Blinn-Phong shading model is supported using vertex and fragment programs. Furthermore, other shading models like Lafortune [LFTG97] or environment mapping can be used. For the additional surface attributes, we assume that these are linearly interpolated over the surface as well.

5.1. Environment Maps

For the environment mapping required for reflection lines, etc., we use prefiltered environment maps. The prefiltering is basically achieved by folding the environment with the kernel of the diffuse and specular part of the Blinn-Phong shading model.

To speed up this folding process, we apply two strategies. Since the kernel of the Blinn-Phong model covers half of the environment, we cut it off using a threshold value (e.g. less than 10^{-8}% of the contribution at the kernel center). This greatly reduces the filtering time for high exponents without reducing the quality of the generated environment. When prefiltering an environment with a low exponent (e.g. for the diffuse environment map), we reduce the size of the environment cube before filtering using mipmapping. To calculate the required resolution, we first determine the radius of n% (we use 90%) contribution and then we choose such a resolution for the filtered environment that this radius is between 1 and 2 pixel. This is reasonable since the kernel is a low pass filter and thus only little information is lost. After the reduction of the kernel size and the environment resolution, we simply multiply the kernel with the environment for each pixel. We cannot use a fourier transformation, since the kernel is slightly different for each point due to the projection from the sphere onto the cube.

These optimizations allow the interactive prefiltering of environment maps for models containing a couple of materials, as shown in the results section.

6. Results

To evaluate the efficiency of our method, we compare it with standard tessellation guaranteeing only a geometric error both with and without using previously generated normal map textures. Then we give some examples for the visualization of various surface properties like surface continuity using isophotes and reflection lines. Finally, we show the applicability of our approach to deformable NURBS models. During the evaluation of our method, we use the two trimmed NURBS models listed in Table 1.

model	materials	trimmed NURBS
wheel rim	1	151
golf	9	8036

Table 1: *Trimmed NURBS models used for evaluation.*

6.1. Performance

All performance tests were made using a PC with an Athlon 2800+ CPU, 512 MB main memory and a Radeon 9800 Pro graphics card. We used the same dynamic LOD and load balancing approach as described in [BGK04b]. As retesselation time, we allow 20ms for each frame, and restrict the screen-space error to 0.5 pixels unless noted otherwise.

	standard	normal maps	app. pres.
max. triangles	314,726	314,736	336,484
min. triangles	136,153	139,163	143,737
avg. triangles	245,857	245,960	260,773

Table 2: *Number of triangles rendered for the golf video sequence using the different tessellation algorithms.*

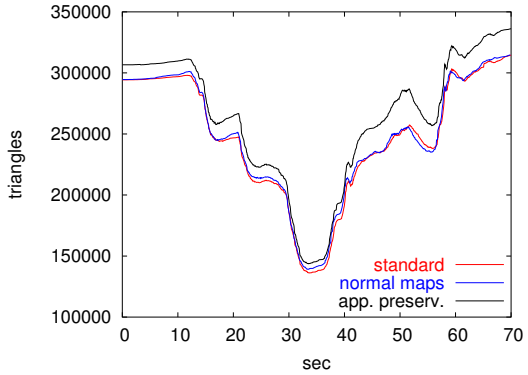

Figure 2: *Number of triangles rendered for the golf video sequence using the different tessellation algorithms.*

Both Table 2 and Figure 2 show that the number of additionally required triangles for appearance preserving tessellation is marginal. Only about 6% additional triangles are required on average.

	standard	normal maps	app. pres.
max. fps	24.32	21.83	24.32
min. fps	10.26	7.52	11.09
avg. fps	18.52	15.22	18.02

Table 3: *Frame rates for the golf video sequence using the golf video sequence using the different tessellation algorithms.*

The frame rates of the appearance preserving tessellation method are almost identical compared to the geometric error

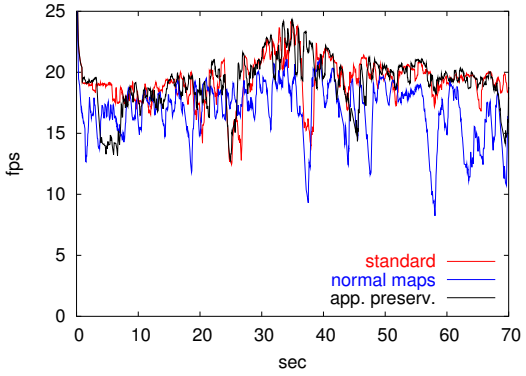

Figure 3: *Frame rates for the golf video sequence using the different tessellation algorithms.*

only approach as shown in Table 3 and Figure 3. The loss of performance is only about 2.7% on average, in contrast to 17% when using normal maps (the relatively high cost of the normal map method comes from the fact that whenever a material changes, the normal map texture has to be changed aswell and texture change is an expensive operation in OpenGL).

exponent	64×64	128×128	256×256	512×512
1	0.004 s	0.010 s	0.021 s	0.078 s
2	0.004 s	0.010 s	0.021 s	0.078 s
5	0.031 s	0.042 s	0.053 s	0.105 s
10	0.031 s	0.042 s	0.053 s	0.105 s
20	0.303 s	0.314 s	0.324 s	0.377 s
50	0.193 s	0.201 s	0.211 s	0.265 s
100	1.776 s	1.779 s	1.788 s	1.912 s
∞	0.024 s	0.081 s	0.322 s	0.359 s

Table 4: *Prefiltering times for different environment resolutions and exponents.*

For interactive environment changes the prefiltering time with the diffuse and specular Phong exponents has to be as low as possible. Using our optimizations, interactive environment switching is possible, as shown by the timings in Table 4. Even for large environments, the prefiltering is always achieved in less that 2 seconds per material. Note that exponents ≥ 128 are treated as ∞, since this is the maximum exponent used in the trimmed NURBS format we use (OpenInventor).

6.2. Image Quality

The most important measure of image quality is the screen space error of the approximation compared to the original.

Table 5 and Figure 4 show the screen space error for the different tessellation algorithms. Note that while the standard and normal map methods only calculate the geometric

	standard	normal maps	app. pres.
max. error	3.71	3.47	3.67
min. error	0.50	0.50	0.50
avg. error	1.27	1.06	1.32

Table 5: *Screen space error for the golf video sequence using the different tessellation algorithms (for the standard and normal map algorithms only the geometric error).*

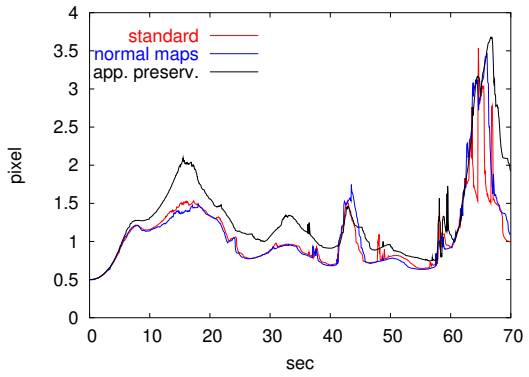

Figure 4: *Screen space error for the golf video sequence using the different tessellation algorithms (for the standard and normal map algorithms only the geometric error).*

error, the screen space error listed for the appearance preserving tessellation gives the combined geometric and shading error. Thus even though the average error seems to be somewhat higher for the appearance preserving method (due to the somewhat larger tessellation and rendering time) in practice the visual quality of our method is much higher, as can be seen in Figure 10.

In order to compare with the previous normal map based approach, we perform a pixel by pixel comparison of the approximated normals with the real normals from the NURBS model in a frame. The difference between the real and approximated normals can be extracted using simple image processing as shown in Figure 5. It is clearly visible that the normal approximation is much better when using the appearance preserving tessellation.

	normal maps	appearance preserving
max. error	1.22^o	1.01^o
min. error	0.48^o	0.28^o
avg. error	0.76^o	0.59^o

Table 6: *Normal approximation error for the golf video sequence using the different visualization algorithms.*

6.3. Surface Properties

In order to present the surface property visualization capabilities of our tessellation algorithm, we implemented the visu-

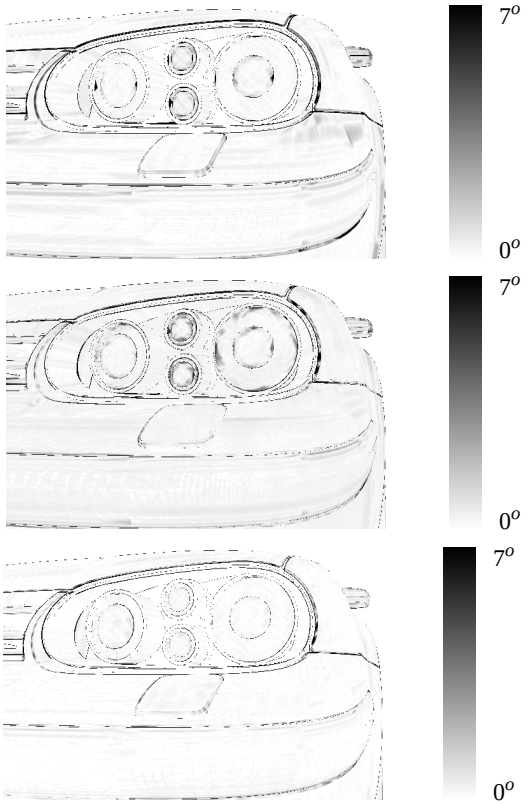

Figure 5: *Normal deviation error for a frame of the rendered animation using geometric approximation without (top) and with normal maps (middle) and appearance preserving tessellation (bottom).*

alization of isophotes (Figure 6) using the intensity value of each pixel after shading to perform a lookup into a striped one-dimensional texture. As shown on Figure 7 our method

Figure 6: *Wheel rim model rendered with isophotes (32 units).*

can also easily show important continuity characteristics of the surface using isophotes. In this case a discontinuity at patch boundaries becomes visible. This is not possible in such a high quality using the normal map approach.

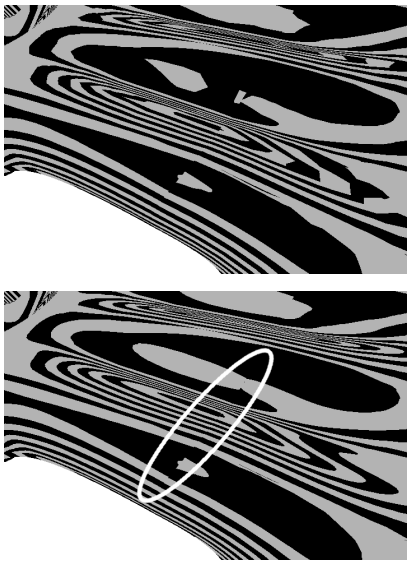

Figure 7: *Close up of the wheel rim model with standard tessellation (top) and appearance preserving tessellation (bottom) using isophotes visualization. The discontinuity is clearly visible only when using appearance preserving tessellation.*

The discontinuity shown in Figure 7 becomes even more apparent with a reflection lines environment as shown in Figure 8. To visualize these reflection lines with our approach, only the environment map has to be generated. No modifications to the algorithm itself are necessary.

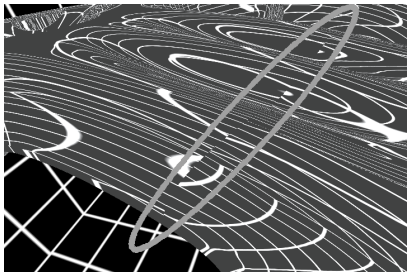

Figure 8: *The discontinuity becomes even more apparent with a reflection lines environment.*

6.4. Deformable NURBS Models

To demonstrate the ability of our method to handle deformable NURBS models, we created an animation of a single NURBS surface where the control points are moved in every frame. This animation achieves about 17 frames per second on average with a guaranteed screen space error of one pixel. Figure 9 shows three frames of the animation sequence.

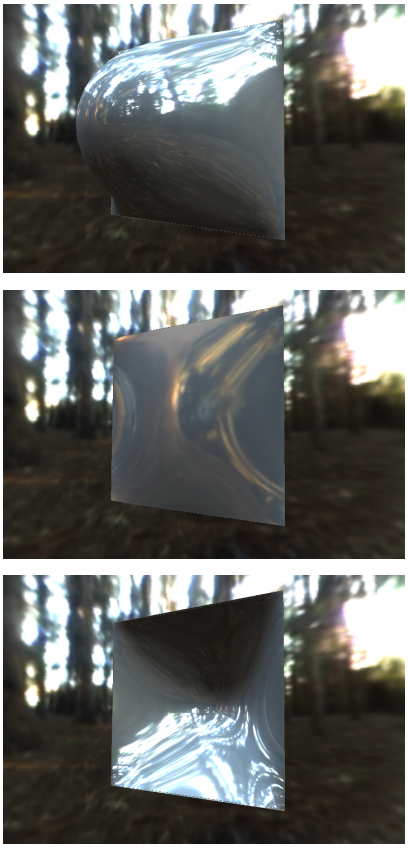

Figure 9: *Three frames from an animation sequence showing a deformable NURBS surface. The elevation in the middle frame is about* 0.1% *of the edge length. Note that standard tessellation would only generate two triangles in this case.*

Due to the amount of preprocessing required, even this simple example would be non interactive (much less then 1 frame per second) if the normal map method is used.

The videos of the golf sequence and the deformable NURBS animation are available at http://cg.cs.uni-bonn.de/project-pages/opensg-plus.

7. Conclusions

In this work, we presented a novel method for automatic appearance preserving tessellation of NURBS surfaces. We demonstrated the weaknesses of previous algorithms in dealing with various illumination/shading artifacts introduced by the discrete nature of tessellation. We also showed how our

algorithm only needs a marginal amount of additional triangles, achieves nearly the same tessellation and rendering speed as standard tessellation algorithms, and performs even better than the normal map based methods. Yet it provides a much higher visualization quality than it was possible with previous approaches. We also demonstrated the ability of our new method to visualize surface properties such as continuity using isophotes and reflection lines. Since our method needs no preprocessing, it is also suitable for the visualization of deformable models and to have immediate feedback during the design and virtual prototyping process.

As future work, we want to exploit the ability of our new algorithm to visualize various important surface properties like temperature distribution or curvature.

Acknowledgements

This work was partially funded by the German Ministry of Education and Research (BMBF) under the project of OpenSG Plus and by the European Union under the project of RealReflect (IST-2001-34744).

We would like to thank Volkswagen AG for providing us with the trimmed NURBS models.

References

[BGK04a] BALÁZS Á., GUTHE M., KLEIN R.: Efficient trimmed nurbs tessellation. In *Journal of WSCG* (February 2004), vol. 12, pp. 27–33. 3

[BGK04b] BALÁZS Á., GUTHE M., KLEIN R.: Fat borders: Gap filling for efficient view-dependent lod rendering. *Computers & Graphics 28*, 1 (2004), 79–86. 2, 3, 6

[BSGM02] BAXTER W. V., SUD A., GOVINDARAJU N. K., MANOCHA D.: Gigawalk: Interactive walkthrough of complex environments, 2002. 2

[BTB02] BALMELLI L., TAUBIN G., BERNADINI F.: Space-optimized texture maps. In *Computer Graphics Forum (Eurographics 2002)* (2002), vol. 21(3), pp. 411–420. 3

[Col01] COLE F.: View dependent appearance preserving simplification. In *Computer Graphics Special Topics)* (2001). 2

[COM98] COHEN J., OLANO M., MANOCHA D.: Appearance-preserving simplification. In *Computer Graphics (Proceedings of SIGGRAPH 98)* (1998). 2

[EC93a] ELBER G., COHEN E.: Hybrid symbolic and numeric operators as tools for analysis of freeform surfaces. In *Working Conference on Geometric Modeling in Computer Graphics* (1993), Falcidieno B., Kurnii T., (Eds.), pp. 275–286. 3

[EC93b] ELBER G., COHEN E.: Second-order surface analysis using hybrid symbolic and numeric operators. *ACM Transactions on Graphics 12*, 2 (1993), 160–178. 3

[FK90] FORSEY D. R., KLASSEN R. V.: An adaptive subdivision algorithm for crack prevention in the display of parametric surfaces. In *Graphics Interface '90* (May 1990), Canadian Information Processing Society, pp. 1–8. 2

[FMEP02] FLORIANI L. D., MAGILLO P., ENRICO PUPPO D. S.: A multi-resolution topological representation for non-manifold meshes. In *7th ACM Symposium on Solid Modeling and Applications* (Saarbrucken, Germany, 2002). 2

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. *Computer Graphics (Proceedings of SIGGRAPH 97) 31* (1997), 209–216. 3

[GH98] GARLAND M., HECKBERT P. S.: Simplifying surfaces with color and texture using quadric error metrics. In *IEEE Visualization '98* (1998), pp. 263–270. 3

[GK03] GUTHE M., KLEIN R.: Efficient nurbs rendering using view-dependent lod and normal maps. In *Journal of WSCG* (February 2003), vol. 11. 3

[GMK02] GUTHE M., MESETH J., KLEIN R.: Fast and memory efficient view-dependent trimmed nurbs rendering. In *proceedings of Pacific Graphics 2002* (2002), IEEE Computer Society, pp. 204–213. 2

[HB87] HERZEN B. V., BARR A. H.: Accurate triangulations of deformed, intersecting surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 89)* (Anaheim, July 1987), vol. 21, pp. 103–110. 2

[HHS*92] HAGEN H., HAHMANN S., SCHREIBER T., NAKAJIMA Y., WÖRDENWEBER B., HOLLEMANN-GRUNDSTEDT P.: Surface interrogation algorithms. In *IEEE Visualization and Computer Graphics* (1992), pp. 53–60. 3

[KBK02] KAHLESZ F., BALÁZS Á., KLEIN R.: Multiresolution rendering by sewing trimmed NURBS surfaces. In *7th ACM Symposium on Solid Modeling and Applications 2002* (Saarbrucken, Germany, June 2002), pp. 281–288. 3

[KLS96] KLEIN R., LIEBICH G., STRASSER W.: Mesh

reduction with error control. In *IEEE Visualization '96* (1996), Yagel R., Nielson. G. M., (Eds.), pp. 311–318. 4

[KMZH97] KUMAR S., MANOCHA D., ZHANG H., HOFF K. E.: Accelerated walkthrough of large spline models. In *1997 Symposium on Interactive 3D Graphics* (April 1997), ACM SIGGRAPH, pp. 91–102. ISBN 0-89791-884-3. 2

[KSS98] KLEIN R., SCHILLING A. G., STRASSER W.: Illumination dependent refinement of multiresolution meshes. In *Proceedings of Computer Graphics International* (1998), Wolter F.-E., Patrikalakis N. M., (Eds.), IEEE Comput. Soc., pp. 680–687. Conference held in Hannover, Germany, 22–26 June 1998. 3

[KSSP01] KUMAR G. V. V. R., SRINIVASAN P., SHASTRY K. G., PRAKASH B. G.: Geometry based triangulation of multiple trimmed nurbs surfaces. *Computer-Aided Design 33*, 6 (May 2001), 439–454. ISSN 0010-4485. 2

[KSZ*96] KORZEN M., SCHRIEVER R., ZIENER K.-U., PAETSCH O., ZUMBUSCH G. W.: Real-time 3-D visualization of surface temperature fields measured by thermocouples on steel structures in fire engineering. In *Proceedings of International Symposium Local Strain and Temperature Measurements in Non-Uniform Fields at Elevated Temperatures* (1996), Ziebs J., Bressers J., Frenz H., Hayhurst D. R., Klingelhöffer H.„ Forest S., (Eds.), Woodhead Publishing, pp. 253–262. 3

[LFTG97] LAFORTUNE E. P. F., FOO S.-C., TORRANCE K. E., GREENBERG D. P.: Non-linear approximation of reflectance functions. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 117–126. 5

[NSK90] NISHITA T., SEDERBERG T. W., KAKIMOTO M.: Ray tracing trimmed rational surface patches. In *Computer Graphics (Proceedings of SIGGRAPH 90)* (Dallas, Texas, August 1990), vol. 24, pp. 337–345. ISBN 0-201-50933-4. 2

[RHD89] ROCKWOOD A. P., HEATON K., DAVIS T.: Real-time rendering of trimmed surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 89)* (Boston, Massachusetts, July 1989), vol. 23, pp. 107–116. 2

[SC88] SHANTZ M., CHANG S.-L.: Rendering trimmed nurbs with adaptive forward differencing. In *Computer Graphics (Proceedings of SIGGRAPH 88)* (Atlanta, Georgia, August 1988), vol. 22, pp. 189–198. 2

[SGSH02] SANDER P. V., GORTLER S. J., SNYDER J., HOPPE H.: Signal-specialized parametrization. In *13th Eurographics Workshop on Rendering* (2002). 3

[SWB98] SLOAN P.-P. J., WEINSTEIN D. M., BREDERSON J. D.: Importance driven texture coordinate optimization. In *Computer Graphics Forum (Eurographics 1998)* (1998), Ferreira N., Göbel M., (Eds.), vol. 17(3), pp. 97–104. 2

[TCRS00] TARINI M., CIGNONI P., ROCCHINI C., SCOPIGNO R.: Real time, accurate, multi-featured rendering of bump mapped surfaces. In *Computer Graphics Forum (Eurographics 2000)* (2000), Gross M., Hopgood F. R. A., (Eds.), vol. 19(3). 2

[TV91] TERZOPOULOS D., VASILESCU M.: Sampling and reconstruction with adaptive meshes. In *Proceedings of the 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Lahaina, HI, 1991), pp. 70–75. 3

[vW03] VAN WIJK J. J.: Image based flow visualization for curved surfaces. In *IEEE Visualization* (2003), Turk G., van Wijk J.„ Moorhead R., (Eds.), pp. 123–130. 3

Figure 10: *A frame from the golf video sequence showing the results of standard tessellation (left), normal maps (middle) and appearance preserving tessellation (right). Note that although the reflections seem to be correct for the normal map method, they are all shifted due to the discretization of the normals leading to false conclusions or even fake discontinuities.*