Pacific Graphics 2012
C. Bregler, P. Sander, and M. Wimmer
(Guest Editors)

Volume 31 (2012), Number 7

Improving the Parameterization of Approximate Subdivision Surfaces

Lei He[1], Charles Loop[2], and Scott Schaefer[1]

[1]Texas A&M University, USA
[2]Microsoft Research, USA

Figure 1: *A textured and displaced subdivision surface and three zooms of the same area (from left to right) showing the original subdivision surface, the approximate subdivision surface and the result of our reparameterization method. Without reparameterization, the texture and displacements are stretched and pulled along edges (see the highlighted region) touching extraordinary vertices due to the geometric continuity constraints of the surface. Without altering the geometry of the surface, our reparameterization technique removes this distortion with little computational cost.*

Abstract

We provide a method for improving the parameterization of patching schemes that approximate Catmull-Clark subdivision surfaces, such that the new parameterization conforms better to that of the original subdivision surface. We create this reparameterization in real-time using a method that only depends on the topology of the surface and is independent of the surface's geometry. Our method can handle patches with more than one extraordinary vertex and avoids the combinatorial increase in both complexity and storage associated with multiple extraordinary vertices. Moreover, the reparameterization function is easy to implement and fast.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

Catmull-Clark subdivision surfaces [CC78] have become a standard for representing highly detailed, smooth and free form shapes such as animated characters in films or computer games. To create a subdivision surface, artists construct a coarse base control mesh to approximate the desired shape. The base mesh is refined by applying a set of rules that depend only on the local topology of the mesh using a linear combination of vertices. Repeating this subdivision process generates a sequence of increasingly finer meshes that converge to a smooth limit surface. Catmull-Clark surfaces are C^2 everywhere except at a finite collection of *extraordinary* vertices (vertices with other than 4 incident edges) where the surface is C^1. This process also naturally partitions the surface into *patches* that correspond to the image of a base control mesh face after repeated subdivision.

Until recently, subdivision surfaces have been used primarily for offline applications such as computer generated movies [DKT98] partially because the size and complexity of the refined meshes increase exponentially with the level

of subdivision. Recent GPU advances have given rise to the possibility of using subdivision surfaces for real-time applications. For example, DirectX 11 [DLO08] adds hardware support for tessellation of arbitrary parametric surfaces. Yet directly tessellating subdivision surfaces can be difficult [LSNCn09] since the extraordinary patches, patches containing one or more extraordinary vertices, are composed of an infinite set of polynomials [HKD93].

Stam [Sta98] provides an exact evaluation technique that efficiently selects the correct polynomial to evaluate for a given point from the infinite set of polynomials using the eigen-decomposition of the local subdivision matrix. However, Loop [Loo10] has shown that exact evaluation can require up to an order of magnitude more floating point operations than evaluating an approximate subdivision surface. Therefore, many researchers have explored approximating subdivision surfaces by replacing extraordinary patches with a single polynomial patch [LS08, MYP8a, LSNCn09] or some small number of polynomial patches [YNM*08, MNP8b]. These methods typically rely on creating geometrically smooth joins to the ordinary patches on the surface rather than the parametrically smooth joins that the underlying subdivision surface possesses (a necessary consequence of using only a finite number of polynomials). While this geometric continuity does not affect the approximation quality of the patching scheme, it will distort the parameterization of the underlying subdivision surface. When texture mapping is used with these approximate representations, the image on the surface is distorted by these artifacts leading to a poor approximation of the desired texture or displacement map on the surface.

Figure 1 shows such an example for a subdivision surface with a texture/displacement map. The center zoomed image shows an approximate Catmull-Clark surface without reparameterization. The displaced bump in the upper left of the image is distorted and no longer circular. Moreover, the large green spot in the texture in the middle-right of the image is pulled so that it is no longer round.

Therefore, we propose to reparameterize such approximate representations to better match the underlying subdivision surface parameterization. Given that the purpose of approximating subdivision surfaces is to perform fast GPU evaluation, any reparameterization must be fast to compute and evaluate. Also, like these approximate representations, we must also be able to handle extraordinary patches that may contain one or more extraordinary vertices.

Contributions

We provide a method for modifying the parameterization of approximating subdivision surfaces such that it conforms better to the original subdivision surfaces. In particular, we

- reparameterize the approximate subdivision surface at runtime using a method that is *independent* of the geometry of the surface and depends solely on the local topology of the control mesh
- provide a method that is fast to evaluate on the GPU, which consists of a single 4×4 matrix product and a single division
- create a simple method to handle patches with multiple extraordinary vertices that avoids combinatorial increases in complexity and storage

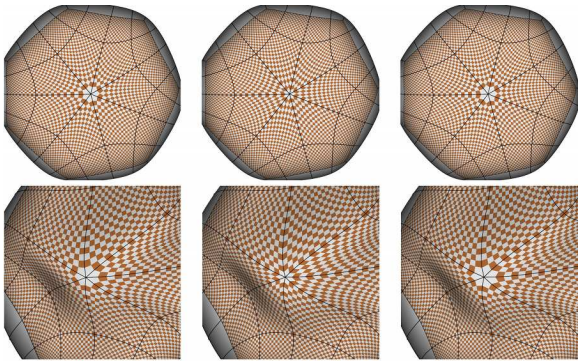

Figure 2: *Demonstration of geometry independence. From left to right: subdivision surface, Gregory approximation, Gregory approximation with our reparameterization. A valence 7 vertex before deformation (top) and after deformation (bottom) where the reparameterization function is identical for both surfaces.*

2. Related Work

Since little work has focused on texturing approximate subdivision surfaces, we separately review previous work on two facets: approximate subdivision surfaces and texture mapping.

Approximate Subdivision Surfaces. Many researchers have considered techniques for approximating Catmull-Clark subdivision surfaces with parametric patches that are more convenient and faster to evaluate than the underlying subdivision surface. Since ordinary patches on the subdivision surface consist of a single polynomial, all approaches retain these portions of the surface and focus on replacing extraordinary patches. Loop and Schaefer [LS08] design bicubic patches to approximate the subdivision surfaces, which were extended to creased subdivision surfaces by Kovacs et al. [KMDZ09]. For each quadrilateral face, they construct a geometry patch and a pair of tangent patches to respectively approximate its 3D shape and tangent fields. Myles et al. [MYP8a] uses a bi-degree 5 patch to approximate the faces containing one or more extraordinary vertices. Yeo et al. [YNM*08] replaces extraordinary

patches with multiple polynomial patches that join together smoothly.

Although Catmull-Clark subdivision surfaces are generally composed of quad patches, artists occasionally add triangles to the control mesh. Myles et al. [MNP8b] extended the work of Yeo et al. [YNM*08] to create Pm-patches for additional triangle patches and pentagonal regions of the shape. More recently, Loop et al. [LSNCn09] investigated a method for approximating Catmull-Clark subdivision surfaces containing both quads and triangles with Gregory patches [Gre74] suitable for programmable hardware implementation. While our approach could be applied to any of these approximation methods, we concentrate here on the Gregory construction of [LSNCn09].

Texture Mapping. When 2D textures are used, the key problem is to find a one-to-one mapping between the 3D surface and 2D texture domain that minimizes the inevitable metric distortion. This process is referred to as parameterization, which has received a lot of attention in recent decades; see [FH05, SPR06] for a comprehensive overview. Most approaches try to minimize an energy function to reduce the distortion measured by different kinds of metrics, such as angle distortion [HG00, LPRM02, KSS06, SSP08], stretch [SSGH01], or a balance between angles and area [DMK03, LZX*08, PTC10]. These parameterization approaches are mainly designed for polygon meshes. He et al. [HSH10] present a method of parameterizing subdivision surfaces by incorporating the inherent subdivision structure.

All of these parameterization approaches rely on decomposing the surface into a set of charts that must be pieced together. Moreover, if the topology of the surface is altered, the surface must be reparameterized. To combat these problems several authors have suggested storing texture information in 3D using an octree to avoid parameterization altogether [BCGB08, DGPR02]. Unfortunately, since the octree is separate from the surface representation, any geometric modification/deformation of the shape will require the octree to rebuilt and resampled. As an alternative, many researchers have advocated using the parameterization of the patches on the surface itself to store texture information [BL08, YKH10]. The advantage of this approach is that each patch is essentially given its own chart, which avoids the need to optimize the parameterization of large regions of the surface. Moreover, when the surface is modified, the texture information deforms with the surface. However, these methods rely on using the parameterization of the patches to store texture information.

Unlike most parameterization methods, our goal is not to create a parameterization that minimizes distortion. Instead, we attempt to modify the parameterization of an approximate subdivision surface such that its parameterization matches that of the original subdivision surface. We focus on per-face-texture mapping [BL08] for purposes of this paper.

In a related work, Boier-Martin et al [BMZ04] explored methods for reparameterizing subdivision surfaces to provide something more akin to a spline patch parameterization by inverting the characteristic map of the subdivision surface. The result is a geometry independent method for reparameterizing the subdivision surface. Our method also provides a geometry independent method for reparameterizing the surface. However, our technique is given two geometrically similar surfaces and reparameterizes one to match the parameterization of the other. We perform an offline optimization procedure to find a simple, computationally efficient reparameterization function that works well with the GPU and operate on surfaces containing multiple extraordinary vertices per patch.

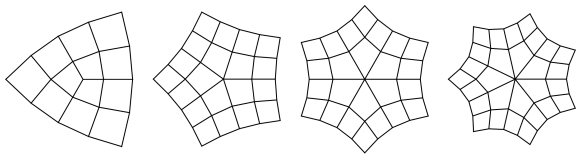

Figure 3: *Characteristic maps for valences 3, 5, 6, and 7.*

3. Texturing Approximate Subdivision Surfaces

Per-face-texture mapping [BL08] relies on using the intrinsic parameterization of each patch to store texture information. This method can be viewed as assigning a single texture to each parametric patch $p_i(u,v)$. The color associated with the point $p_i(u,v)$ is simply the texture sampled at (u,v). Therefore, to minimize texture distortion, we must minimize the difference between the parameterization of the subdivision surface and the approximate subdivision surface.

Even though the geometry of the subdivision surface and the approximate subdivision surface are visually similar, parametric distortion must exist due to the G^1 smoothness conditions from the approximate subdivision surface. Figure 8 (upper right) shows an example of equally spaced parameter lines in the u and v directions on the subdivision surface (blue) and the approximate subdivision surface (red). Subdivision surfaces are parametrically smooth, which causes parameter lines to be smooth across patch boundaries. The red lines from the approximate subdivision surface lie on a geometrically smooth surface, but contain kinks.

We minimize this distortion by defining a reparameterization function $s_i(u,v) : \mathbb{R}^2 \to \mathbb{R}^2$. Given a patch $p_i(u,v)$ of the subdivision surface and the corresponding patch $g_i(u,v)$ of the approximate subdivision surface represented as a Gregory patch [Gre74], we define our problem as finding a function s such that

$$\min_s \int_{u=0}^{1} \int_{v=0}^{1} |p_i(u,v) - g_i(s_i(u,v))|^2 \qquad (1)$$

for each patch of the surface subject to the constraint that s_i

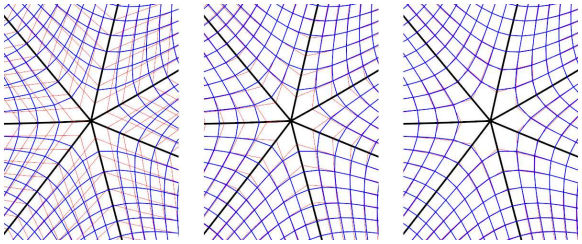

Figure 4: *In each picture the subdivision surface is shown in blue, the approximate subdivision surface in red, and patch boundaries are shown in black. From left to right: no reparameterization, reparameterization with a bicubic function, and reparameterization with a rational bicubic function. The rational bicubic gives a substantially better fit to the subdivision surface, especially along patch boundaries.*

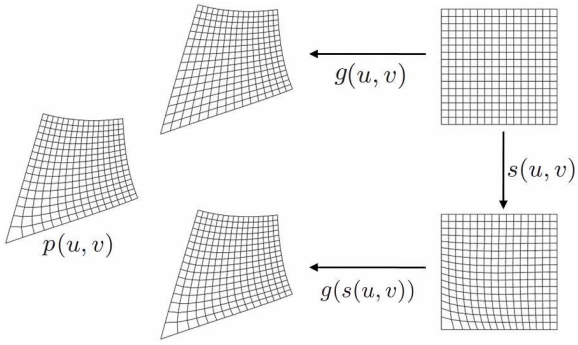

Figure 5: *The various mappings used to create our reparameterization function $s(u,v)$ are illustrated. The Gregory patch approximation $g(u,v)$ of a characteristic map patch $p(u,v)$, along with the resulting reparameterization $g(s(u,v))$.*

forms a bijection from $[0,1)^2$ to $[0,1)^2$. This constraint requires that s_i does not fold back on itself $\left|\begin{array}{cc}\frac{\partial s_i}{\partial u} & \frac{\partial s_i}{\partial v}\end{array}\right| > 0$ and, for any point (u_j,v_j) in the domain of p_i, there exists a point (u_k,v_k) such that $s_i(u_k,v_k) = (u_j,v_j)$. Then, for each point $g_i(u,v)$, we define its texture coordinates with respect to per-face-texture mapping as $s_i(u,v)$. Note that, although we visualize the reparameterized surface, the actual vertex locations in the Gregory surface are unchanged. We only modify the texture coordinates of the vertices and use $s_i(u,v)$ for the texture coordinate instead of (u,v). Note that the evaluation point (u,v) for the surface $g_i(u,v)$ is unchanged by $s_i(u,v)$.

Since the approximate subdivision surface is identical to the subdivision surface over patches containing only ordinary vertices, Equation 1 is minimized with the identity transform $s_i(u,v) = (u,v)$ for these patches. However, this minimization is much more difficult for patches that contain one or more extraordinary vertices. Our goal is not simply to minimize this error, but to build a reparameterization function s_i that is simple to construct and easy to evaluate on the GPU.

To that end, we represent $s_i(u,v)$ as a rational bicubic function with a 4*x*4 matrix of control points C where each control point has a parametric location (u,v) as well as a rational weight w. If we represent C as two matrices, C_{uv} and C_w, containing the parametric and rational components of each control point separately, we can represent $s_i(u,v)$ as

$$s_i(u,v) = \frac{\mathbf{B}^3(u)\cdot(\mathbf{C}_{uv} * \mathbf{C}_w)\cdot\mathbf{B}^3(v)}{\mathbf{B}^3(u)\cdot\mathbf{C}_w\cdot\mathbf{B}^3(v)},$$

where $\mathbf{B}^d(\cdot)$ are degree d Bézier basis functions and $*$ denotes the element-by-element product of two matrices.

While we could have picked some other representation of $s_i(u,v)$, we have found that this choice performs well in all of our examples. Moreover, this function is easily computed, because evaluating $g_i(u,v)$ already requires evaluating both $\mathbf{B}^3(u)$ and $\mathbf{B}^3(v)$. Therefore, evaluating $s_i(u,v)$ comes at very little additional cost (a 4×4 matrix-vector product, a dot product, and a division). We choose a rational bicubic rather than a non-rational bicubic as a reparameterization function not only for the additional degrees of freedom, but because rational functions fit the fractal scaling of the subdivision surface well along patch edges. Figure 4 shows the benefit of choosing rational bicubic reparameterizations over non-rational parameterizations.

Even though we have defined an explicit form for $s_i(u,v)$, minimizing Equation 1 is difficult due to the non-linear correspondence between $g_i(u,v)$ and $p_i(u,v)$. Therefore, we discretize Equation 1 using a dense set of parametric points (u_j,v_j) from p_i and points (u_k,v_k) from g_i. For each point $p_i(u_j,v_j)$, we find the closest point $g_i(u_{k(j)},v_{k(j)})$. We then fit our reparameterization function s_i by minimizing

$$\min_{s_i}\sum_j \left|(u_j,v_j) - s_i(u_{k(j)},v_{k(j)})\right|^2 dp_j \qquad (2)$$

where $k(j)$ gives the index of the closest point (u_k,v_k) to (u_j,v_j) and dp_j is the differential area of the point $p_i(u_j,v_j)$. We use a uniform grid of size 17^2 for (u_j,v_j), a grid of size 5000^2 for (u_k,v_k), and the area of the barycentric cells [MDSB03] for dp_j.

3.1. Geometry Independence

While it is possible to evaluate and minimize Equation 2, this equation is dependent on the geometry of both the subdivision surface p_i and the Gregory approximation g_i. As the user changes the geometry, the correspondence between points may change. Unfortunately, computing the correspondence between surfaces and performing the nonlinear mini-

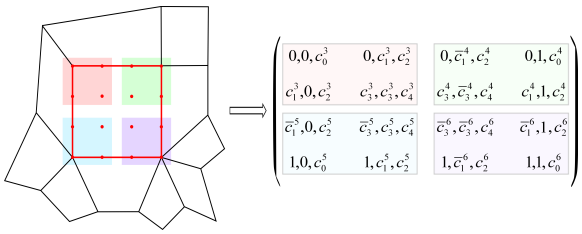

Figure 6: *Construction of $s_i(u,v)$ for a patch with valences 3, 4, 5, and 6 with the 16 control points of s_i visualized as red dots. The 2×2 block of control points closest to a vertex, highlighted in different colors, are solely a function of the valence of that vertex.*

Valence	c_0^n	c_1^n	c_2^n	c_3^n	c_4^n
3	0.0759	0.1014	0.6548	0.2105	0.9240
4	1.0000	0.3333	1.0000	0.3333	1.0000
5	0.3296	0.0850	0.7913	0.3223	0.8958
6	0.4188	0.0759	0.8259	0.3636	0.8768
7	0.4830	0.0686	0.8487	0.3949	0.8595
8	0.5293	0.0629	0.8645	0.4189	0.8451
9	0.5634	0.0586	0.8757	0.4372	0.8335
10	0.5890	0.0553	0.8840	0.4515	0.8242

Table 1: *Precomputed parameters for Equation 3.*

mization in Equation 2 cannot be performed in realtime even on the GPU.

Our solution is to create a reparameterization function that is independent of the geometry of the surface. To do so, we note that the rules that refine the subdivision surface are solely dependent on the topology of the surface. The limit of this subdivision process is related to the characteristic map [Rei95] of the subdivision scheme, which is only dependent on the topology of the surface (see Figure 3 for examples of characteristic maps). After repeated subdivision, an ε-neighborhood about an extraordinary point will approach an affine transformation of this characteristic map. Therefore, the characteristic map can provide a first order approximation of the subdivision surface and is independent of the embedding of the subdivision surface in $\mathbb{R}^3$.

Using this observation, our solution is to find correspondence using the characteristic map of the subdivision scheme instead of the actual 3D geometry of the surface. Therefore, for each vertex of a particular valence (number of edges incident on the vertex), we use the characteristic map as the control mesh for the subdivision scheme. This control mesh defines both the subdivision surface p_i and its Gregory approximation g_i and we compute our correspondence points using the patches in the one-ring of the central vertex.

Figure 5 depicts this process for a single patch from the characteristic map of a valence 7 vertex. The $[0,1)^2$ domain is transformed to the Gregory approximation by the map $g(u,v)$ (top) but does not resemble the subdivision surface $p(u,v)$. We use these two shapes to compute correspondence points to fit the reparameterization $s(u,v)$. Applying $s(u,v)$ warps the domain such that if we then apply $g(u,v)$, we obtain a surface that matches the subdivision surface well as shown on the bottom of the figure.

We make the additional restriction that the 2×2 block of control points closest to each corner should only be a function of the valence of the corner vertex. While optimizing the positions of all 16 control points would generate a more accurate fit, our experiments show that adjusting just one quarter of the entries creates a close fit. This restriction allows us to easily handle multiple extraordinary vertices in Section 3.2. Since characteristic maps have only one extraordinary vertex all of the control points except for a 2×2 block of C are known because the identity map is the optimal solution for valence 4 vertices. Characteristic maps are also symmetric and requiring $s_i(u,v)$ to be a bijection contrains some of the coordinates for control points along the boundary. Therefore, we can write the control points C of s_i as

$$C = \begin{pmatrix} (0,0,c_0^n) & (0,c_1^n,c_2^n) & (0,\frac{2}{3},1) & (0,1,1) \\ (c_1^n,0,c_2^n) & (c_3^n,c_3^n,c_4^n) & (\frac{1}{3},\frac{2}{3},1) & (\frac{1}{3},1,1) \\ (\frac{2}{3},0,1) & (\frac{2}{3},\frac{1}{3},1) & (\frac{2}{3},\frac{2}{3},1) & (\frac{2}{3},1,1) \\ (1,0,1) & (1,\frac{1}{3},1) & (1,\frac{2}{3},1) & (1,1,1) \end{pmatrix}. \quad (3)$$

Therefore, s_i only contains 5 degrees of freedom corresponding to the parameters $c_i^n, i = 0,1,2,3,4$. We optimize the nonlinear function in Equation 2 with the constraint that the Jacobian is positive $\left| \frac{\partial s_i}{\partial u} \quad \frac{\partial s_i}{\partial v} \right| > 0$ using the Levenberg-Marquardt algorithm [Lev44]. Table 1 gives the result of the optimization for various valences. Note that, because the optimization is only dependent on the local topology of the surface, only Table 1 is needed to construct the reparameterization function.

3.2. Multiple Extraordinary Vertices

While the construction in Section 3.1 can handle patches with one extraordinary vertex, our goal is to be able to operate on patches that contain any number (up to all four corners) extraordinary vertices. Moreover, we desire a simple reparameterization method that will avoid the potentially combinatorial increase in the number of topological cases depending on the valence of each vertex of a quadrilateral patch, which is quartic in the number of different valences.

We provide a simple solution where we replace the 2×2 block of control points in C corresponding to each vertex with the optimized solution from Table 1. Figure 6 shows an example of this process for a patch with valence 3, 4, 5, and 6 vertices. For each 2×2 block of control points (shown in different colors), we use the optimized results from Table 1 for that valence where $\bar{c} = 1 - c$. This technique yields a simple

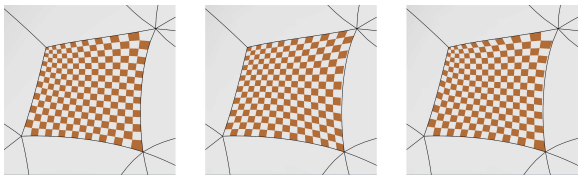

Figure 7: *A patch containing only extraordinary vertices of valence 3, 5, 6, and 7. From left to right: subdivision surface, Gregory surface without reparameterization, our reparameterization method.*

method for handling patches with arbitrary valence vertices using only a small table of precomputed numbers. While this approach is not optimal with respect to the minimization in Equation 2 for patches that contain multiple extraordinary vertices, the number of these cases is typically low in most surfaces. Even so, we have found that this solution works well in practice.

4. Results

We provide several examples of our technique in Figures 1, 8, and 9 applied to the Gregory patch subdivision surface approximation [LSNCn09]. Figure 1 shows an example of a displaced subdivision surface with a texture map. Without reparameterization, the texture and displacement are stretched. Our reparameterization (right) removes this distortion creating a result almost identical to the original subdivision surface.

While Figure 1 provides a real-world example, the ability to perceive these differences is dependent on the data in the texture itself. For example, regions of constant color do not show distortion no matter how bad the parameterization is. Therefore, in Figure 8 we visualize the difference in the parameterization between the true subdivision surface and the Gregory approximation using a pointwise evaluation of Equation 1 shown as color where blue represents low error and red is high error. Before reparameterization, the error is high and concentrated around extraordinary vertices of the surface. After our reparameterization, the geometry of the surface is unchanged, but the parameterization error is substantially diminished. The right of the figure shows a close-up of patches containing a single extraordinary vertex of valence 5. The blue and red edges on the top are equally spaced parameter lines from the subdivision surface and original Gregory patches respectively. Ideally these lines would be identical, but differences exist especially along patch boundaries. These differences are necessary because subdivision surfaces are parametrically smooth, and hence have smooth parameter lines across patch boundaries, while Gregory patches only meet with geometric continuity, which generates only continuous parameter lines. The bottom of the image shows the same picture after reparameterization and the parameter lines are nearly identical even though our reparameterization is geometry independent.

Figure 2 demonstrates this geometry independence on a simple example of a valence 7 vertex. The left of the figure shows the subdivision surface, the center the Gregory approximation, and the right the Gregory surface with our reparameterization. On the bottom is the same figures except we have deformed the center vertex to alter the geometry. In both cases, our reparameterization matches the parameterization of the original subdivision surface well.

Figure 9 demonstrates surfaces containing patches with multiple extraordinary vertices per patch. The top of the figure shows a patch containing a valence 3 and a valence 6 vertex that are face-adjacent, while the bottom shows a patch containing a valence 3 and a valence 5 vertex that are edge-adjacent. Before reparameterization, the difference between the parameterization of the two surfaces in each example is quite high, which can lead to significant distortion of the texture. After using our reparameterization function, the parameter lines are well-aligned. Compared with only patches containing only a single extraordinary vertex, the reparameterization is not as accurate, but still fits well. Figure 7 shows an extreme case of a single patch containing a valence 3, 5, 6 and 7 vertex. Even with every vertex an extraordinary vertex, our reparameterization matches the subdivision surface well.

Our method is efficient to compute in a DirectX 11 domain shader. Evaluating a Gregory patch relies on values of the Bernstein basis functions at a (hardware provided) u,v domain location. Since $s_i(u,v)$ is also represented in the Bernstein basis and is rational, evaluating the reparameterization function amounts to performing a multiply with the values of the basis functions (already computed) and dividing by the rational component, which is a single division. In our implementation, we represent the Gregory patch control points passed into the domain shader as a `float3` pair, effectively a 6-tuple $\{x,y,z,u,v,w\}$, where x,y,z are position coordinates, and u,v,w are the corresponding rational control vertices of our reparameterization. For the 8 interior Gregory control points, we duplicate the reparameterization control vertices pairwise, to exactly reproduce our bicubic reparmeterization function.

5. Conclusion

Our method provides an inexpensive method for modifying the texture coordinates of an approximation subdivision surface to match that of the original subdivision surface, which reduces texture distortion when using per-face texture mapping approaches [BL08, YKH10]. Since our method is independent of the geometry of the surface, our reparameterization is both easy to compute and capable of handling animating surfaces in real-time.

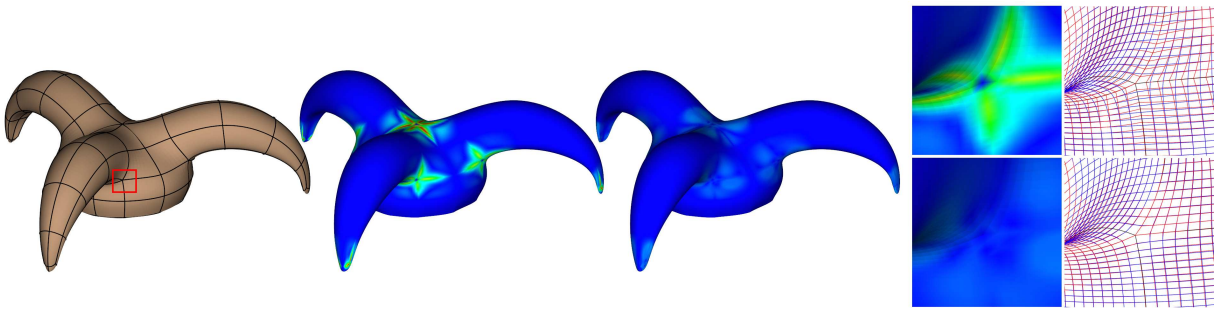

Figure 8: *From left to right: The patch structure of a subdivision surface, the difference between the parameterization of the subdivision surface and the approximate subdivision surface (blue=low error, red=high error), the difference using our reparameterization method, and a zoom-in of the highlighted region. The zoom shows both the false coloring of the error function as well as equally spaced parameter lines for the subdivision surface (blue) and the approximate subdivision surface (red) both before (top) and after (bottom) reparameterization.*

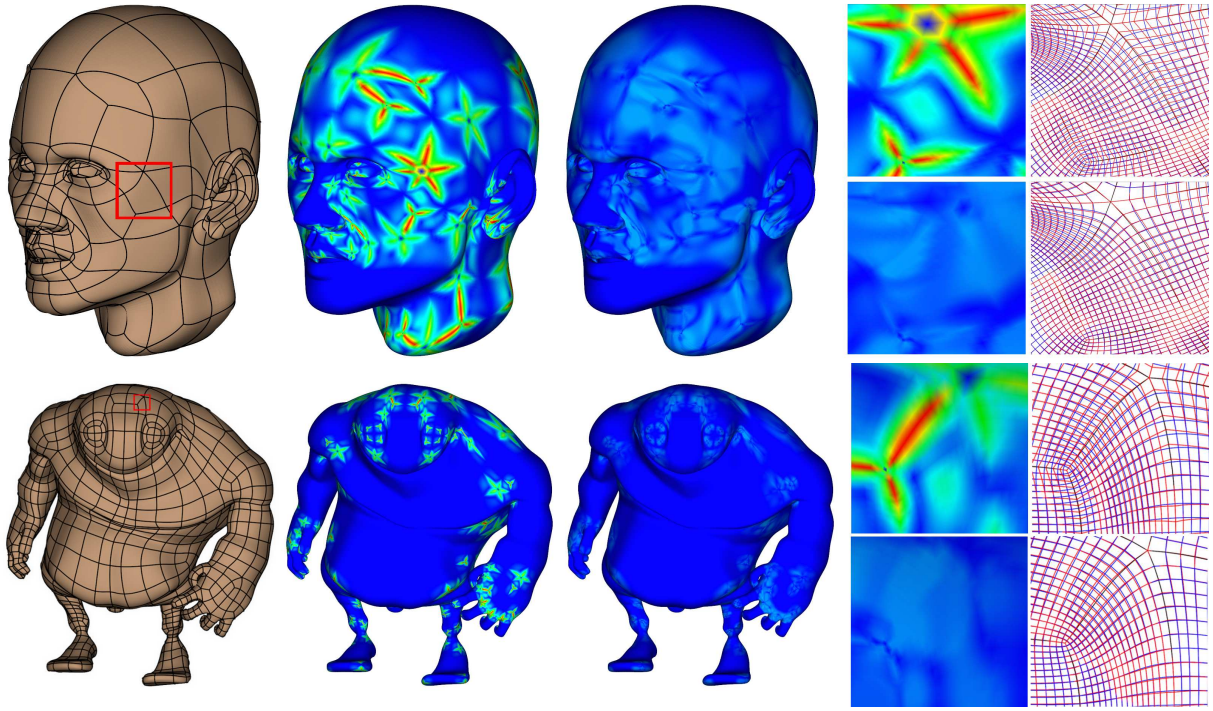

Figure 9: *From left to right: The patch structure of a subdivision surface, the difference between the subdivision surface parameterization and the Gregory surface, the difference using our reparameterization, and a zoom-in of the highlighted region showing equally space parametric lines of the subdivision surface (blue) and the Gregory patch (red) before (top) and after (bottom) reparameterization.*

Acknowledgments

We would like to thank Bay Raitt for the models of the monster frog and big guy as well as Jason Smith for the model of the jester hat. This work was supported in part by NSF CAREER award IIS-1148976.

References

[BCGB08] BEN-CHEN M., GOTSMAN C., BUNIN G.: Conformal flattening by curvature prescription and metric scaling. *Computer Graphics Forum 27*, 2 (2008), 449–458. 3

[BL08] BURLEY B., LACEWELL D.: Ptex: Per-face texture mapping for production rendering. In *Eurographics Symposium on Rendering 2008* (2008), pp. 1155–1164. 3, 6

[BMZ04] BOIER-MARTIN I., ZORIN D.: Differentiable parameterization of catmull-clark subdivision surfaces. In *the Symposium on Geometry Processing* (2004), pp. 155–164. 3

[CC78] CATMULL E., CLARK J.: Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design 10*, 6 (1978), 350–355. 1

[DGPR02] DEBRY D. G., GIBBS J., PETTY D. D., ROBINS N.: Painting and rendering textures on unparameterized models. In *SIGGRAPH* (2002), pp. 763–768. 3

[DKT98] DEROSE T., KASS M., TRUONG T.: Subdivision surfaces in character animation. In *SIGGRAPH* (1998), pp. 85–94. 1

[DLO08] DRONE S., LEE M., ONEPPO M.: Direct3D 11 Tessellation. http://www.microsoft.com/downloads/details.aspx?FamilyId=2D5BC492-0E5C-4317-8170-E952DCA10D46, 2008. 2

[DMK03] DEGENER P., MESETH J., KLEIN R.: An adaptable surface parameterization method. In *Proceedings of the 12th International Meshing Roundtable* (2003), pp. 201–213. 3

[FH05] FLOATER M. S., HORMANN K.: Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*. Springer, 2005, pp. 157–186. 3

[Gre74] GREGORY J. A.: Smooth interpolation without twist constraints. *Computer Aided Geometric Design* (1974), 71–87. 3

[HG00] HORMANN K., GREINER G.: MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*. Vanderbilt University Press, 2000, pp. 153–162. 3

[HKD93] HALSTEAD M., KASS M., DEROSE T.: Efficient, fair interpolation using catmull-clark surfaces. In *SIGGRAPH* (1993), pp. 35–44. 2

[HSH10] HE L., SCHAEFER S., HORMANN K.: Parameterizing subdivision surfaces. *ACM Transactions on Graphics 29*, 4 (2010), 120:1–120:6. 3

[KMDZ09] KOVACS D., MITCHELL J., DRONE S., ZORIN D.: Real-time creased approximate subdivision surfaces. In *the Symposium on Interactive 3D graphics and games* (2009), pp. 155–160. 2

[KSS06] KHAREVICH L., SPRINGBORN B., SCHRÖDER P.: Discrete conformal mappings via circle patterns. *ACM Transactions on Graphics 25*, 2 (2006), 412–438. 3

[Lev44] LEVENBERG K.: A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics 2* (1944), 164–168. 5

[Loo10] LOOP C.: *Hardware Subdivision and Tessellation of Catmull-Clark Surfaces*. Tech. Rep. MSR-TR-2010-163, Microsoft Research, 2010. 2

[LPRM02] LÉVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics 21*, 3 (2002), 362–371. 3

[LS08] LOOP C., SCHAEFER S.: Approximating Catmull–Clark subdivision surfaces with bicubic patches. *ACM Transactions on Graphics 27*, 1 (2008), 8:1–8:11. 2

[LSNCn09] LOOP C., SCHAEFER S., NI T., CASTAÑO I.: Approximating subdivision surfaces with gregory patches for hardware tessellation. In *SIGGRAPH ASIA* (2009), pp. 151:1–151:9. 2, 3, 6

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. *Computer Graphics Forum 27*, 5 (2008), 1495–1504. 3

[MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*. Springer-Verlag, 2003, pp. 35–57. 4

[MNP8b] MYLES A., NI T., PETERS J.: Fast parallel construction of smooth surfaces from meshes with tri/quad/pent facets. In *the Symposium on Geometry Processing* (2008b), pp. 1365–1372. 2, 3

[MYP8a] MYLES A., YEO Y. I., PETERS J.: Gpu conversion of quad meshes to smooth surfaces. In *Proceedings of the ACM symposium on Solid and physical modeling* (2008a), pp. 321–326. 2

[PTC10] PIETRONI N., TARINI M., CIGNONI P.: Almost isometric mesh parameterization through abstract domains. *IEEE Trans. on Visualization and Computer Graphics 16*, 2 (2010), 621–635. 3

[Rei95] REIF U.: A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design 12* (1995), 153–174. 5

[SPR06] SHEFFER A., PRAUN E., ROSE K.: Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision 2*, 2 (2006), 105–171. 3

[SSGH01] SANDER P., SNYDER J., GORTLER S., HOPPE H.: Texture mapping progressive meshes. In *SIGGRAPH* (2001), pp. 409–416. 3

[SSP08] SPRINGBORN B., SCHRÖDER P., PINKALL U.: Conformal equivalence of triangle meshes. *ACM Transactions on Graphics 27*, 3 (2008), 77:1–77:11. 3

[Sta98] STAM J.: Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH* (1998), pp. 395–404. 2

[YKH10] YUKSEL C., KEYSER J., HOUSE D. H.: Mesh colors. *ACM Transactions on Graphics 29*, 2 (2010), 15:1–15:11. 3, 6

[YNM*08] YEO Y., NI T., MYLES A., GOEL V., PETERS J.: Parallel smoothing of quad meshes. *The Visual Computer 25*, 8 (2008), 757–769. 2, 3