Approximating Catmull-Clark Subdivision Surfaces with Bicubic Patches

Charles Loop
Microsoft Research

Scott Schaefer
Texas A&M University

April 24, 2007

Technical Report
MSR-TR-2007-44

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Abstract

We present a simple and computationally efficient algorithm for approximating Catmull-Clark subdivision surfaces using a minimal set of bicubic patches. For each quadrilateral face of the control mesh, we construct a geometry patch and a pair of tangent patches. The geometry patches approximate the shape and silhouette of the Catmull-Clark surface and are smooth everywhere except along patch edges containing an extraordinary vertex where the patches are C^0. To make the patch surface appear smooth, we provide a pair of tangent patches that approximate the tangent fields of the Catmull-Clark surface. These tangent patches are used to construct a continuous normal field (through their cross-product) for shading and displacement mapping. Using this bifurcated representation, we are able to define an accurate proxy for Catmull-Clark surfaces that is efficient to evaluate on next-generation GPU architectures that expose a programmable tessellation unit.

1 Introduction

Catmull-Clark subdivision surfaces [4] have become a standard modeling primitive in computer generated motion pictures and 3D games. To create a subdivision surface, an artist constructs a coarse polygon mesh that approximates the shape of the desired surface. A subdivision algorithm recursively refines this shape to produce a sequence of finer shapes that converge to a smooth surface. In practice, the user needs to perform only 3 or 4 refinement steps to produce a dense mesh suitable for rendering. For real-time applications, this uniform refinement provided by subdivision is undesirable due to the large number of polygons generated. Therefore, adaptive tessellation techniques are needed to avoid overwhelming the graphics pipeline with triangles.

Recently [3], [12], have used the GPU to dynamically tessellate Catmull-Clark surfaces. While these methods utilize graphics hardware, the performance of these schemes is not impressive. As GPUs continue to evolve, support for higher order tessellation directly in hardware has become a reality [8]. The tessellator unit in the Xbox 360's GPU provides hardware support for adaptive tessellation of parametric surfaces. Based on user-provided tessellation factors, the tessellator adaptively creates a sampling pattern of the underlying parametric domain and automatically generates a set of triangles connecting these samples. The programmer then provides a special shader program that the tessellator calls with the parametric coordinates (u, v) for each sample in the parametric patch; the shader then emits a vertex that corresponds to the patch evaluated at those coordinates.

This approach allows the GPU to triangulate arbitrary parametric surfaces because the evaluation details are provided by the programmer in the form of a shader. Furthermore, this technique allows the GPU to exploit parallelism because multiple arithmetic units can be running the same evaluation shader

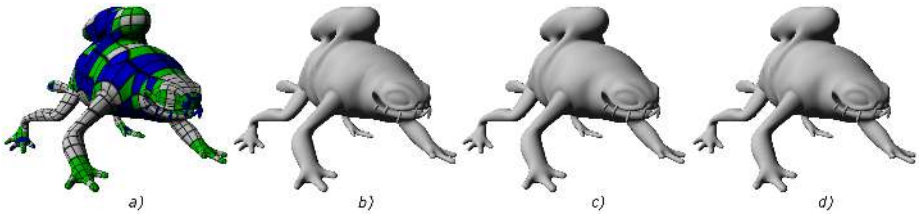

Figure 1: *a)* The patch structure we associate with a Catmull-Clark subdivision surface. The grey patches contain only valence 4 vertices, green have one extraordinary vertex and blue have more than one extraordinary vertex. *b)* Our approximation to the Catmull-Clark subdivision surface using geometry patches and *c)* our final approximation using geometry and tangent patches compared with *d)* the actual Catmull-Clark limit surface.

in lock-step. We expect hardware tessellation to appear in the next generation of GPU architectures [2] as it is an obvious way to overcome the bottleneck of bus bandwidth caused by scene/model complexity and amplifies data directly on the GPU. The resulting variable tessellation representation has the added benefit of being a good level-of-detail management strategy.

Catmull-Clark subdivision surfaces are in fact piecewise parametric and therefore amenable to hardware tessellation. Each quadrilateral polygon in a Catmull-Clark control mesh corresponds to a single bicubic patch except for quadrilaterals that contain an extraordinary vertex (a vertex not touched by exactly four quadrilaterals). These *extraordinary patches*, patches containing one or more extraordinary vertices, are actually composed of an infinite collection of bicubic patches. Using this polynomial structure, Stam developed an algorithm for efficiently evaluating the parametric form of Catmull-Clark surfaces in constant time [14].

While programmable tessellation hardware will be capable of running Stam's algorithm, there are a number of issues that suggest alternative surface schemes will have much better performance. Stam's method requires that extraordinary patches contain only one extraordinary vertex. If there are patches that contain more than one extraordinary vertex, one level of subdivision must be performed first yielding 4 times as many patches to evaluate (see Figure 2). Furthermore, Stam's algorithm projects control points into an *eigenspace* (that varies with the valence) before evaluation begins, which eliminates the possibility of sharing control points among adjacent patches. The evaluation itself requires that $2n+8$ bicubic *eigenbasis* functions be evaluated (for a patch containing an n-valent extraordinary vertex); the $32n + 128$ coefficients of these functions must read from memory for each evaluation instance. Hence, the large number of patches, the inability to share patch data and large number of constants will tax hardware resources.

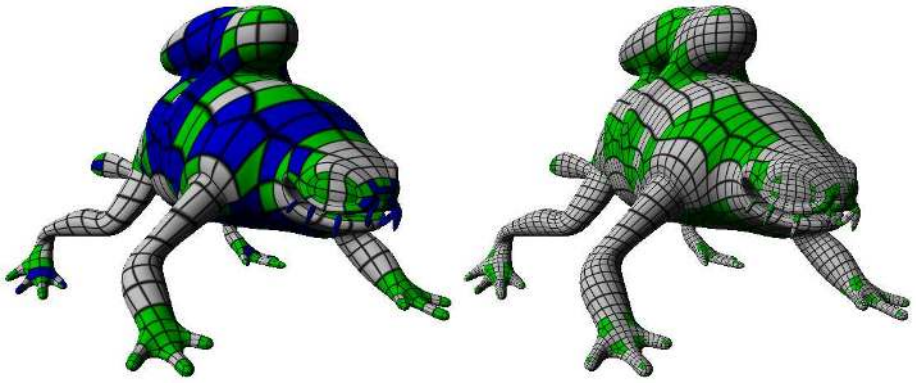

Figure 2: Blue patches (left) contain more than one extraordinary vertex and cannot be evaluated using Stam's method. The subdivided shape (right) contains patches with one or less extraordinary vertices but increases the number of patches by a factor of 4.

Contributions
We propose an algorithm for visually approximating Catmull-Clark subdivision surfaces, possibly with boundaries, using a collection of bicubic patches (one for each face of a quad-mesh). These patches are smooth everywhere except along edges leading to an extraordinary vertex where they are only C^0; therefore shading discontinuities may result. We overcome this difficulty by creating independent tangent patches that conspire to produce a continuous normal field and, hence, the appearance of a smooth surface. When each vertex of the patch has valence 4, our geometry and tangent patches are identical to the Catmull-Clark subdivision surface. Furthermore, we place no restriction on the number or valence of extraordinary vertices per patch.

2 Previous Work

Some of the early work in this area used Gregory patches [5] to create surfaces that interpolate networks of curves and allow the user to specify cross-boundary derivatives. While these patches could be used to approximate Catmull-Clark surfaces, the patches are rational polynomials whose denominators vanish at patch corners complicating evaluation. Furthermore, these patches contain few degrees of freedom that can be used to approximate Catmull-Clark surfaces and suffer from shape problems like flat spots at extraordinary vertices.

[10] describes an algorithm that converts Catmull-Clark surfaces into a NURBS approximation of the subdivision surface. This method creates one bicubic polynomial patch for each face of a quad mesh. The surfaces produced are

C^2 everywhere except near extraordinary vertices where they are C^1. However, this method requires that the base quad mesh be subdivided at least once (twice if there are vertices of even valence > 4) to create sufficient separation of extraordinary vertices resulting in at least 4 times as many patches as the base subdivision surface.

[12] present a method for directly subdividing Catmull-Clark surfaces on programmable graphics hardware. Since their technique actually simulates subdivision, the surfaces do not suffer from any approximation artifacts. Though patches can be individually tessellated to different levels of resolution, their method requires that the samples are evenly spaced at intervals of $\frac{1}{2^L}$ because Catmull-Clark subdivision performs a binary split at each level of subdivision. However, true adaptive sampling cannot be accomplished with this method and is incompatible with the sampling patterns produced by GPU tessellator units.

Finally, Curved PN Triangles (sometimes known as N-Patches) [15] bear the most similarity to our work. This method takes as input a set of triangles with normals specified at the vertices and attempts to build an interpolating, smooth surface consisting of cubic Bézier triangles. Unfortunately, the patches are not smooth across their edges. To combat this effect, the authors create a separate normal field that gives the surface the appearance of being smooth. The advantage of this method is that the computations are local and a patch can be constructed using only the information present in a single triangle. The disadvantage is that the surfaces suffer from various shading artifacts and the lack of smoothness can typically be seen in the silhouette of the object.

In contrast, our method produces geometry patches that are smooth almost everywhere and the lack of smoothness is rarely if ever visible in the silhouette of the model (the silhouettes of all of our examples appear smooth even under extreme conditions). Our surfaces are also identical to the Catmull-Clark subdivision surfaces everywhere except for patches containing one or more extraordinary vertices. Finally, we use larger neighborhoods of vertices when constructing our patches. The larger support allows us to create better approximations to the underlying smooth surface than Curved PN Triangles, which use minimal information and exhibit shape artifacts.

3 Geometry Patches

For each face in a quad-mesh, we construct a bicubic patch to approximate the Catmull-Clark surface over the corresponding region. We represent these bicubic patches in Bézier form with the labeling scheme illustrated in Figure 3.

Our geometry patch construction is a generalization of B-spline knot insertion, used to convert from the B-spline to Bézier basis. If all four vertices of a quad-mesh face have valence 4, then the construction reproduces the standard uniform B-spline patch in Bézier form. There are three types of masks needed to construct the control points of a Bézier patch from a uniform B-spline control

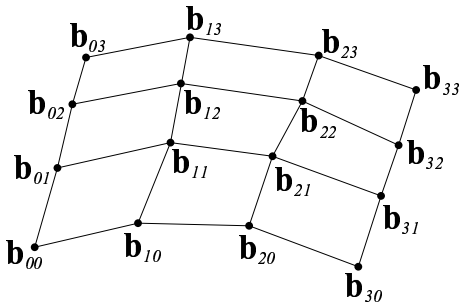

Figure 3: Control point labeling for a bicubic Bézier patch.

mesh as shown in Figure 4. These masks encode a set of coefficients that are applied by summing the products of these coefficients and the corresponding points. For masks that generate points (such as the masks in this figure) there is an implied normalization that these masks sum to 1. However, masks that generate vectors must sum to 0 and, thus, do not have an implied normalization.

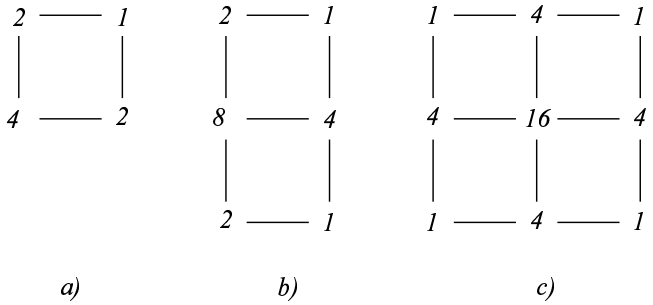

Figure 4: Masks for determining Bézier control points from a uniform bicubic B-spline surface.

Referring to Figure 4, mask *a)* is used (in four orientations) to create the four *interior points* $\mathbf{b}_{11}, \mathbf{b}_{21}, \mathbf{b}_{12}$ and $\mathbf{b}_{22}$, corresponding to each quad face; mask *b)* is used to create the *edge points* $\mathbf{b}_{10}, \mathbf{b}_{20}, \mathbf{b}_{01}, \mathbf{b}_{02}, \mathbf{b}_{31}, \mathbf{b}_{32}, \mathbf{b}_{13}$ and $\mathbf{b}_{23}$ corresponding to the edges of the quad-mesh; finally, mask *c)* is used to create the *corner points* $\mathbf{b}_{00}, \mathbf{b}_{30}, \mathbf{b}_{03}$ and $\mathbf{b}_{33}$ corresponding to the vertices of the quad-mesh. Note that each edge point lies at the midpoint of two interior points, belonging to adjacent faces; and each corner point lies at the centroid of the 4 interior points that surround that vertex. Our general quad-mesh patch construction is inspired by these geometric relationships.

In the ordinary case (all vertices of the patch have valence 4), the corner points $\mathbf{b}_{00}, \mathbf{b}_{30}, \mathbf{b}_{03}$ and $\mathbf{b}_{33}$ interpolate the limit position of the Catmull-Clark

surface. Therefore, in the extraordinary case, we also choose these control points to interpolate the limit position of the Catmull-Clark surface. [7] showed that the left eigenvector corresponding to the dominant eigenvalue of the Catmull-Clark subdivision matrix corresponds to the mask that generates the limit position of an extraordinary vertex. Figure 5 *c)* illustrates this limit position mask.

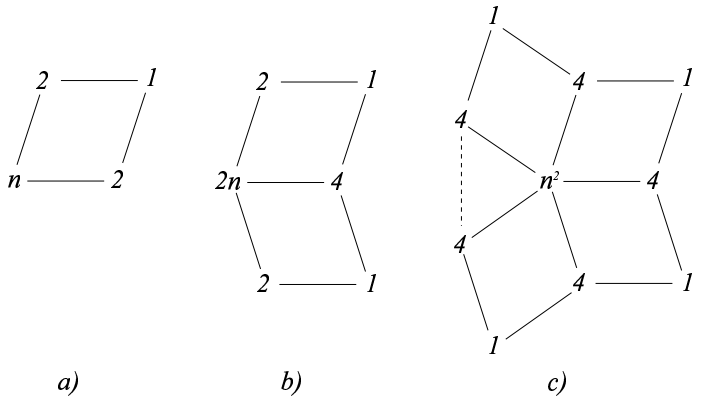

Figure 5: Generalized masks for interior, edge and corner points.

If the centroid of the surrounding interior Bézier points creates the corner point with the mask shown in Figure 5 *c)*, then we can solve directly for the mask for the interior points (shown in Figure 5 *a)*. Note that the value n in the generalized interior point mask corresponds to the valence of the vertex whose weight is n. Furthermore, this valence may differ for each interior point $\mathbf{b}_{11}, \mathbf{b}_{21}, \mathbf{b}_{12}$ and $\mathbf{b}_{22}$. Finally, the edge points are found as midpoints of the adjacent interior points leading to the mask shown in Figure 5 b). Notice that, if $n = 4$, the masks in Figure 5 reproduce the knot insertion masks in the uniform case shown in Figure 4.

4 Tangent Patches

In general, replacing the Catmull-Clark surface with the geometry patches from the previous section results in a surface that is smooth everywhere except along edges containing extraordinary vertices. For some applications, this lack of smoothness may be acceptable. However, for techniques such as displacement mapping, we need a surface that has a continuous normal field over the entire surface. The normal field of a bicubic surface is biquintic, which produces a large number of control vectors in Bézier form to exactly represent the biquintic polynomial (36 control vectors). Furthermore, the control vectors do not depend linearly on the underlying control mesh complicating animation. Therefore, our approach uses a pair of tangent patches that approximate the ∂u and ∂v tangent

fields of the geometry patches allowing us to recover the biquintic normal field via their cross product. These tangent patches have fewer control vectors (they are degree 3×2) and depend linearly on the control mesh.

Consider the tangent patch ∂u. This patch will be bidegree 2×3, since differentiating the bidegree 3×3 geometry patch with respect to u will lower the degree by one in the u-direction. Similarly, the ∂v patch will be bidegree 3×2. Since the ∂u and ∂v patches represent vector fields, their coefficients are control vectors, as illustrated in Figure 6. The construction of tangent patches

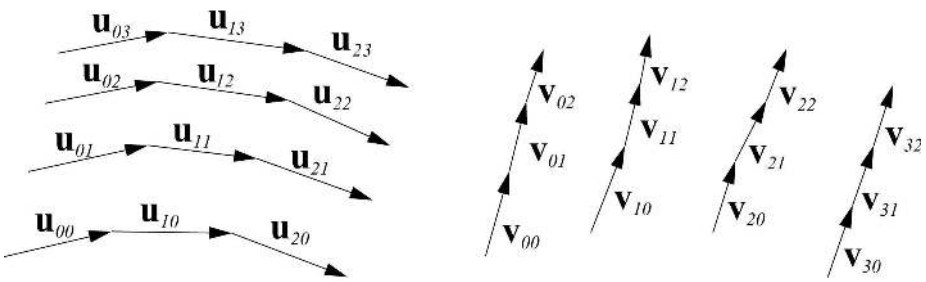

Figure 6: Control *vectors* for tangent patches ∂u and ∂v.

is symmetric; that is, the constructions are identical up to an interchange of principle directions, with appropriate change of signs. Therefore, we limit our discussion to the ∂v patch.

For Bézier patches, the ∂v patch can be found using differences of the control points. If $\mathbf{b}_{i,j}$ are the coefficients of the geometry patch and $\mathbf{v}_{k,l}$ are the coefficients of the tangent patch, then

$$\mathbf{v}_{i,j} = 3\left(\mathbf{b}_{i,j+1} - \mathbf{b}_{i,j}\right), \quad i = 0, \ldots, 3 \quad j = 0, \ldots, 2. \tag{1}$$

These control vectors represent a Bézier patch that exactly encodes the tangent field in the v-direction of the corresponding geometry patch. However, since the geometry patches do not meet with C^1 continuity everywhere, the tangent patches ∂u and ∂v will not create a continuous normal field. In particular, the tangent field must create a unique normal at the corners of the patch (shared by multiple patches) and along the edges of the patch (shared by two patches). Therefore, we must modify the control points along the edges of ∂v such that we create a continuous normal field over the entire surface.

4.1 Tangent Patch Corners

To modify our tangent patch ∂v, we begin with the corner vertices $\mathbf{v}_{00}, \mathbf{v}_{02}, \mathbf{v}_{30}$, and $\mathbf{v}_{32}$, which should produce a unique tangent plane among all patches sharing this corner. Unfortunately, our geometry patches are not smooth for an

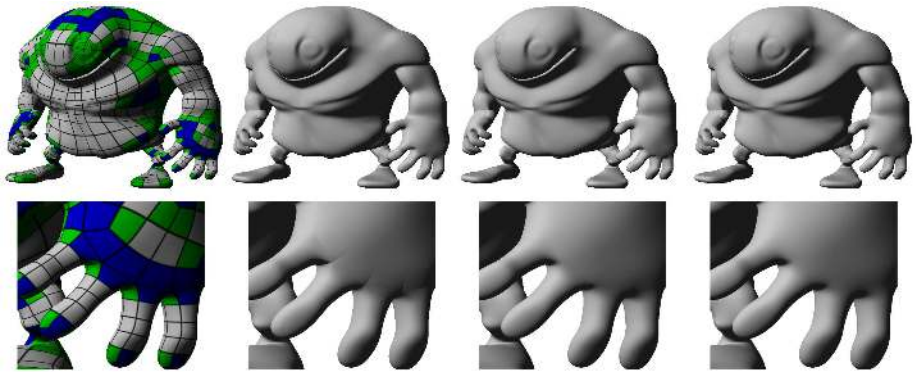

Figure 7: An example mesh (top) and a zoomed in region of a complex patch structure (bottom). From left to right: Catmull-Clark patch structure, Geometry patch approximation, Geometry/Tangent patch approximation and Catmull-Clark limit mesh.

arbitrary valence vertex so our construction from Equation 1 does not produce a unique tangent plane. Given that the geometry patches are meant to approximate the Catmull-Clark surface, we use the limit tangent mask of the Catmull-Clark surface to create a unique tangent plane at the corners of the patch.

[7] showed that the tangent limit masks for Catmull-Clark surfaces correspond to the left eigenvectors of the subdivision matrix associated with the subdominant eigenvalue pair. Using these eigenvectors, we arrive at a tangent mask

$$\begin{aligned} \alpha_i^L &= \cos\left(\tfrac{2\pi i}{n}\right), \\ \beta_i^L &= \left(\frac{\sqrt{4+\cos\left(\frac{\pi}{n}\right)^2}-\cos\left(\frac{\pi}{n}\right)}{4}\right)\cos\left(\tfrac{2\pi i+\pi}{n}\right). \end{aligned}$$

where α^L and β^L are the coefficients for the left eigenvector and use the labeling shown in figure 8 *b)*. Unfortunately, this relationship between the left eigenvectors of the subdivision matrix and the tangent mask only generates a mask that determines the direction of the tangent vector and not its length (eigenvectors are independent of scale). Therefore, we must find an appropriate length for this vector to ensure a well behaved tangent field.

Our approach conceptually uses the *characteristic map* of the subdivision scheme as a local parameterization of the surface [11]. Similar to the tangent mask, the coordinates of the characteristic map are given by the pair of right eigenvectors corresponding to the subdominant eigenvalues. If we allow α^R, β^R to be points in the plane, then the one-ring control points of the characteristic

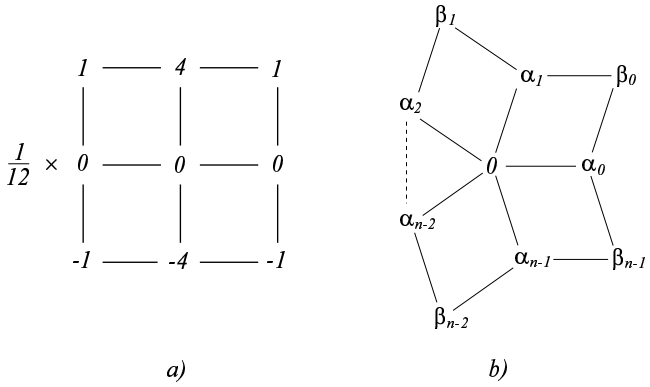

Figure 8: *a)* Tangent mask for uniform bicubic B-splines surfaces *b)* Mask for Catmull-Clark limit tangent.

map are

$$\begin{aligned} \alpha_i^R &= \left(\cos\left(\tfrac{2\pi i}{n}\right), \sin\left(\tfrac{2\pi i}{n}\right)\right) \\ \beta_i^R &= \frac{4}{\sqrt{4+\cos\left(\frac{\pi}{n}\right)^2}+\cos\left(\frac{\pi}{n}\right)} \left(\cos\left(\tfrac{2\pi i+\pi}{n}\right), \sin\left(\tfrac{2\pi i+\pi}{n}\right)\right). \end{aligned}$$

The characteristic map is also independent of scale, which we are free to choose. We pick a scale for the map such that $\alpha_0^R = (1, 0)$.

If we apply the limit tangent mask to the control points of the characteristic map, the result will be a vector with non-unit length. We then find a scalar σ such that

$$\sigma \sum_{i=0}^{n-1} \alpha_i^L \, \alpha_i^R + \beta_i^L \, \beta_i^R \;=\; (1, 0).$$

Solving for σ yields

$$\sigma \;=\; \frac{1}{n} + \frac{\cos\left(\frac{\pi}{n}\right)}{n\sqrt{4+\left(\cos\left(\frac{\pi}{n}\right)\right)^2}}.$$

Multiplying the previous tangent mask by σ produces the final tangent mask.

$$\begin{aligned} \alpha_i &= \left(\frac{1}{n} + \frac{\cos\left(\frac{\pi}{n}\right)}{n\sqrt{4+\left(\cos\left(\frac{\pi}{n}\right)\right)^2}}\right) \cos\left(\tfrac{2\pi i}{n}\right), \\ \beta_i &= \left(\frac{1}{n\sqrt{4+\left(\cos\left(\frac{\pi}{n}\right)\right)^2}}\right) \cos\left(\tfrac{2\pi i+\pi}{n}\right). \end{aligned} \tag{2}$$

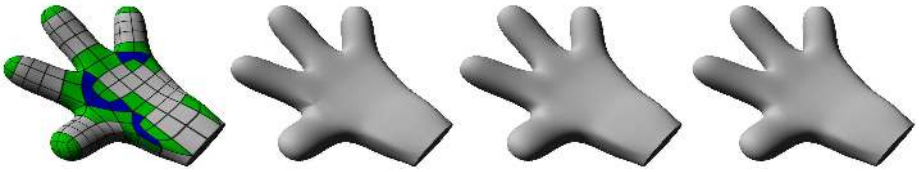

Figure 9: An example of a mesh with a boundary. From left to right: Catmull-Clark patch structure, Geometry patch approximation, Geometry/Tangent patch approximation and Catmull-Clark limit surface using Biermann et al.'s boundary rules.

This tangent mask is used to construct the vectors $\mathbf{v}_{00}, \mathbf{v}_{02}, \mathbf{v}_{30}$, and $\mathbf{v}_{32}$ resulting in a unique tangent plane at each of the patch corners. Also, note that all these vectors must be consistently aligned. In particular, the tangent vector directions must be reversed (multiplied by -1) for $\mathbf{v}_{02}$ and $\mathbf{v}_{32}$. The construction of tangent field vectors $\mathbf{u}_{00}, \mathbf{u}_{20}, \mathbf{u}_{03}$, and $\mathbf{u}_{23}$ is identical. Finally, notice that if $n = 4$, this tangent mask exactly reproduces the tangent mask for bicubic B-splines in Figure 8 *a)* including scale.

4.2 Tangent Patch Edges

Given the tangent patch from Equation 1 with corner vertices specified by Equation 2, the tangent patches create a unique tangent plane everywhere except along the edges of a patch. Therefore, we must modify the control vectors along the edges of the patch as well.

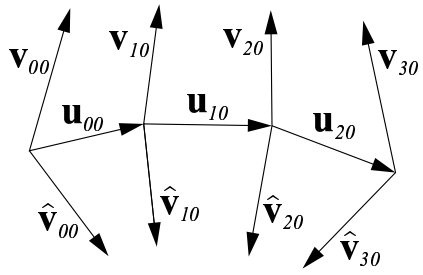

Figure 10: Control vectors involved in smooth edge conditions.

Consider the patch edge in Figure 10 shared by two patches. These control vectors define two cubic functions $v(t), \hat{v}(t)$ and one quadratic function $u(t)$. These three vector fields will be linearly dependent if

$$((1-t)c_0 + tc_1)\,\mathbf{u}(t) = \frac{1}{2}\left(\mathbf{v}(t) + \hat{\mathbf{v}}(t)\right) \quad \forall t \in [0,1]$$

where $c_i = \cos\left(\frac{2\pi}{n_i}\right)$ and n_0, n_1 are the valence of the left and right endpoints. Solving for the Bézier coefficients results in four conditions:

$$\begin{aligned} c_0\,\mathbf{u}_{00} &= \tfrac{1}{2}\left(\mathbf{v}_{00} + \hat{\mathbf{v}}_{00}\right), && (3)\\ \tfrac{1}{3}\left(2\,c_0\mathbf{u}_{10} - c_1\mathbf{u}_{00}\right) &= \tfrac{1}{2}\left(\mathbf{v}_{10} + \hat{\mathbf{v}}_{10}\right), && (4)\\ \tfrac{1}{3}\left(c_0\mathbf{u}_{20} - 2\,c_1\mathbf{u}_{10}\right) &= \tfrac{1}{2}\left(\mathbf{v}_{20} + \hat{\mathbf{v}}_{20}\right), && (5)\\ -c_1\,\mathbf{u}_{20} &= \tfrac{1}{2}\left(\mathbf{v}_{30} + \hat{\mathbf{v}}_{30}\right). && (6) \end{aligned}$$

Conditions (3) and (6) are satisfied by construction using Equation 2. Condition (4) will be satisfied if

$$\begin{aligned} \mathbf{v}_{10} &= \tfrac{1}{3}\left(2\,c_0\mathbf{u}_{10} - c_1\mathbf{u}_{00}\right) + \mathbf{x},\\ \hat{\mathbf{v}}_{10} &= \tfrac{1}{3}\left(2\,c_0\mathbf{u}_{10} - c_1\mathbf{u}_{00}\right) - \mathbf{x} \end{aligned}$$

for any choice of $\mathbf{x}$. We choose $\mathbf{x} = 3\left(\mathbf{b}_{11} - \mathbf{b}_{10}\right)$ since, by construction, we get the same vector $\mathbf{x}$ (up to sign) when processing either patch sharing an edge. Furthermore, this construction reproduces the regular case ($n = 4$). The control vector $\mathbf{v}_{20}$ can be found in a similar fashion. To summarize, we set

$$\begin{aligned} \mathbf{v}_{10} &= \tfrac{1}{3}\left(2\,c_0\mathbf{u}_{10} - c_1\mathbf{u}_{00}\right) + 3\left(\mathbf{b}_{11} - \mathbf{b}_{10}\right),\\ \mathbf{v}_{20} &= \tfrac{1}{3}\left(c_0\mathbf{u}_{20} - 2\,c_1\mathbf{u}_{10}\right) + 3\left(\mathbf{b}_{21} - \mathbf{b}_{20}\right). \end{aligned}$$

The construction for $\hat{\mathbf{v}}_{10}$, and $\hat{\mathbf{v}}_{20}$ follows in a similar manner.

5 Results

Over the ordinary patches in the mesh (no extraordinary vertices), our construction for the geometry and tangent patches exactly reproduces the surface and tangent field of the Catmull-Clark surface. Therefore, the only regions that our surfaces differ from the actual Catmull-Clark surface are those patches containing one or more extraordinary vertices. Furthermore, our method interpolates the limit position and normal of the Catmull-Clark surface at each vertex of the mesh. Though our geometry patch approximation is only C^0, the lack of smoothness is rarely if ever visible in the silhouette of the model (we have not been able to discern the C^0 regions of the models along the silhouette in any of our examples).

Figures 1, 7 and 12 show examples of subdivision surfaces containing patches with one or more extraordinary vertices. The approximation using only Geometry patches matches the Catmull-Clark surface very well, but is noticeably not smooth along some of the edges. Adding the tangent field to the model creates a smooth normal field and results in a surface that is nearly identical visually to the original Catmull-Clark surface. Because the tangent patches create a continuous normal field over the surface, we can use these shapes for displacement

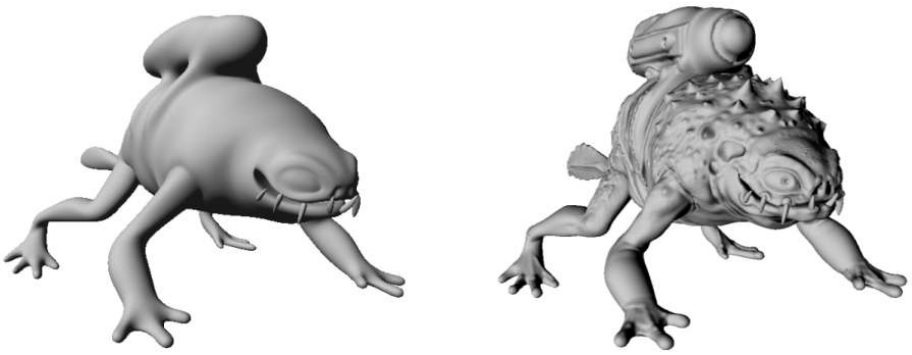

Figure 11: Our Geometry/Tangent patch model (left) with displacement mapping (right).

Figure 12: A complex mesh with boundary. From left to right: Catmull-Clark patch structure, Geometry patch approximation, Geometry/Tangent patch approximation and Catmull-Clark limit surface.

mapping as well. Figure 11 shows an example of displacement mapping applied to our geometry/tangent patch approximation from Figure 1.

Figure 9 illustrates an example of a mesh with a boundary containing both ordinary and extraordinary vertices (our boundary construction can be found in Appendix A). The Catmull-Clark surface uses the boundary rules of [1] to produce a smooth subdivision surface. Again, our geometry patch approximation matches the profile and shape of the Catmull-Clark surface, but the lack of continuity along a few patch edges is evident in the shading of the surface. With the tangent patches, we create a continuous normal field over the entire surface.

Since our method produces only polynomial patches for each quad in the original mesh, the surface is very easy to evaluate at arbitrary parametric values. We have implemented this method on the Xbox 360 using the programmable

tessellator unit to create real-time, adaptive tessellations of our approximate Catmull-Clark surfaces. Even with our preliminary implementation, we have been able to achieve tessellation rates of over 100 million polygons/second (including normal calculations). The accompanying video provides a real-time demonstration of our technique.

Our patches are also substantially faster to evaluate than Catmull-Clark surfaces using Stam's exact evaluation method (our patches approximate the Catmull-Clark surface whereas Stam's algorithm evaluates the true limit surface). Even assuming that all patches contain only one extraordinary vertex (no 4 times multiple in the number of patches) and ignoring the projection step in Stam's algorithm for each patch, which alone contains $(2n + 8)^2$ multiplies, Stam's method still performs at least $100n + 420$ multiply operations per evaluation, which includes operations for extracting the tangents for lighting. Most of these operations are due to repeated multiplication by the b-spline basis functions $2n + 8$ times in the evaluation loop. In contrast, our method performs at most $4n + 226$ multiplications assuming that every control point is recalculated for every evaluation including the evaluation of the tangents. Therefore we expect a minimum speed-up of at least 3x over Stam's method even if all patches only have one extraordinary vertex. In the more realistic scenario, where patches contain more than one extraordinary vertex, we expect at least an order of magnitude speed-up.

6 Future Work

While Catmull-Clark surfaces are typically created from quad-meshes, the subdivision rules are general enough to handle meshes with arbitrary sided polygons. Arbitrary polygons are theoretically possible in our framework, but are not practical for adaptive tessellation on current graphics hardware, which support only triangular and quadrilateral domains. However, it is possible to incorporate triangle patches into the tessellation process.

For simplicity, we only operate on meshes consisting entirely of quads. We can, of course, produce an all-quad mesh by performing one round of subdivision. However, the disadvantage of this approach is the increase in the number of patches similar to Figure 2. In the future we would like to extend our method to triangular patches, and more generally triangle-quad surfaces [13] using Bézier triangles.

Finally, Catmull-Clark surfaces are smooth everywhere while many everyday surfaces contain sharp edges or corners. We can handle these creases by marking edges in the mesh and treating them as boundary edges. However, [6] introduced rules to create semi-sharp creases in Catmull-Clark surfaces. We believe that we may be able to incorporate semi-sharp creases into our method by modifying the patch coefficients as well.

References

[1] Henning Biermann, Adi Levin, and Denis Zorin. Piecewise smooth subdivision surfaces with normal control. In *SIGGRAPH 2000: Computer Graphics Proceedings*, pages 113–120, 2000.

[2] David Blythe. The direct3d 10 system. *ACM Trans. Graph.*, 25(3):724–734, 2006.

[3] Jeffrey Bolz and Peter Schröder. Rapid evaluation of catmull-clark subdivision surfaces. In *Proceeding of the International Conference on 3D Web Technology*, pages 11–17, 2002.

[4] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

[5] H. Chiyokura and F. Kimura. Design of solids with free-form surfaces. *Computer Graphics*, 17(3):289–298, 1983.

[6] T. DeRose, M. Kass, and T Truong. Sudivision surfaces in character animation. In *Siggraph 1998, Computer Graphics Proceedings*, pages 85–94, 1998.

[7] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using catmull-clark surfaces. *Computer Graphics*, 27(Annual Conference Series):35–44, 1993.

[8] Matt Lee. Next-generation graphics programming on xbox 360. http://download.microsoft.com/download/d/3/0/d30d58cd-87a2-41d5-bb53-baf560aa2373/Next_Generation_Graphics_Programming_on_Xbox_360.ppt, 2006.

[9] Ahmad H. Nasri. Polyhedral subdivision methods for free-form surfaces. *ACM Trans. Graph.*, 6(1):29–73, 1987.

[10] Jörg Peters. Patching catmull-clark meshes. In *SIGGRAPH 2000: Computer Graphics Proceedings*, pages 255–258, 2000.

[11] U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12:153–174, 1995.

[12] Le-Jeng Shiue, Ian Jones, and Jörg Peters. A realtime gpu subdivision kernel. *ACM Trans. Graph.*, 24(3):1010–1015, 2005.

[13] J. Stam and C. Loop. Quad/triangle subdivision. *Computer Graphics Forum*, 22(1):1–7, 2003.

[14] Jos Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. *Computer Graphics*, 32(Annual Conference Series):395–404, 1998.

[15] Alex Vlachos, Jörg Peters, Chas Boyd, and Jason L. Mitchell. Curved pn triangles. In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 159–166, 2001.

A Meshes with Boundaries

Orginally, Catmull-Clark surfaces were assumed to be closed surfaces; however, not all meshes are closed. [9] extended Catmull-Clark subdivision to surfaces with boundaries. Along the boundary, Nasri chose the subdivision rules to reproduce cubic B-splines. To generalize our geometry patches to boundaries, we follow [9] and require that the boundary curves form cubic B-splines.

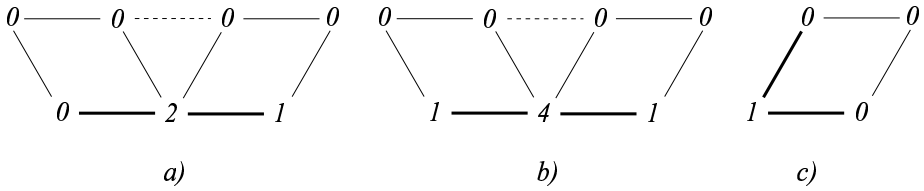

Figure 13: Rules for Bézier control points along the boundary to create a cubic B-spline. Bolded edges indicate boundaries. From left to right: mask for an edge control point, corner control point and a corner control point contained by only one quad.

A.1 Geometry Patches

For a boundary edge, the two edge points are found along the edge at ratios $1:2$ and $2:1$ from the endpoints. For a boundary vertex incident on two or more faces, we find the corner point as the midpoint of the two adjacent edge points. For a boundary vertex contained by only one face, we set the corner point to the boundary vertex. These rules are summarized in Figure 13 with the implied normalization that the masks sum to 1.

Besides modifying the Bézier control points along the boundary, we also modify the interior control point adjacent to a boundary vertex. For boundary vertices contained by more than one quad, we use the mask in Figure 14 *a)* where k is equal to the number of quads containing the boundary vertex. When a boundary vertex is contained by only one quad, the interior control point is given by Figure 14 *b)*. Similar to Section 3, edge points on interior edges are placed at the midpoint of the adjacent interior points. In the interest of

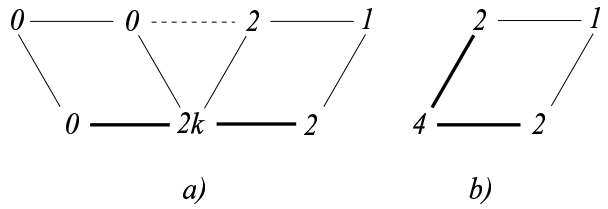

Figure 14: Rules for creating the interior Bézier control point adjacent to a boundary vertex contained by more than one quad *a)* and only one quad *b)*.

simplicity, we ignore topologically anomalous configurations, such as bow-ties and pin-wheels; though in principle, such configuration do not cause problems.

We make a distinction from the valence n used in Sections 3 and 4 and the number of quads k containing a boundary vertex because we treat the boundary as being half of a closed mesh. Therefore, $n = 2k$. In fact, if we apply this identity to Figure 14, we obtain the interior point mask for closed meshes shown in Figure 5.

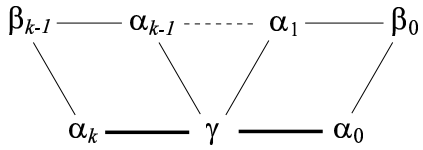

Figure 15: Labeling for vertices around a boundary vertex contained by k quads.

A.2 Tangent Patches

Our tangent patch construction is identical to Section 4 except that we modify the way the limit tangents are computed at boundary vertices. For boundary vertices contained by more than one quad, we use the tangent masks derived for Catmull-Clark surfaces by [1] to compute the tangent vectors. These tangent vectors provide direction, but lack length information and we use the same normalization process from Section 4 to choose an appropriate length. The result is two tangent masks that create two vectors r_0 and r_1 that span the

tangent plane at that vertex. Referring to Figure 15, the mask for r_0 is

$$\begin{aligned} \gamma &= \tfrac{-4s}{3k+c} \\ \alpha_0 = \alpha_k &= -\tfrac{(1+2c)\sqrt{1+c}}{(3k+c)\sqrt{1-c}} \\ \alpha_{i\neq 0,k} &= \tfrac{4s_i}{3k+c} \\ \beta_i &= \tfrac{s_i s_{i+1}}{3k+c} \end{aligned}$$

and for r_1

$$\begin{aligned} \alpha_0 &= \tfrac{1}{2} \\ \alpha_k &= -\tfrac{1}{2} \\ \alpha_{i\neq 0,k} = \gamma = \beta_i &= 0 \end{aligned}$$

where $c = \cos\left(\frac{\pi}{k}\right), s = \sin\left(\frac{\pi}{k}\right)$ and $s_i = \sin\left(\frac{\pi i}{k}\right)$.

The tangent vector along the j^{th} edge is then given by

$$\cos\left(\frac{\pi j}{k}\right) r_0 + \sin\left(\frac{\pi j}{k}\right) r_1.$$

The rest of the tangent patch construction is the same except that we use the substitution $n = 2k$ in Section 4.2 for the edges of the tangent patches.