On the Use of Loop Subdivision Surfaces for Surrogate Geometry

Per-Olof Persson[1], Michael J. Aftosmis[2], and Robert Haimes[3]

[1] Massachusetts Institute of Technology persson@mit.edu
[2] NASA Ames Research Center aftosmis@nas.nasa.gov
[3] Massachusetts Institute of Technology haimes@mit.edu

Abstract. This work examines the use of Loop subdivision surfaces as a surrogate for CAD or analytic-based geometry. The modeler begins by constructing a subdivision surface from a full-resolution imported surface triangulation, and then queries this surface to return information about the model. Evaluations on the surface are performed using a simplified implementation of Stam's exact evaluation procedure for Loop subdivision surfaces. The paper presents details of this simplified approach and shows how it can be used to provide surface coordinates, derivatives, and curvatures for evaluations at arbitrary parameter values. The implementation also provides the ability to tag *hard*-edges (and implicitly *hard*-vertices) in the imported geometry to preserve creases and points using a one-dimensional cubic-spline scheme which preserves C^2 continuity along hard-edges. Away from hard-edges and vertices, the Loop surface is C^2 continuous everywhere except in the immediate vicinity of irregular vertices in the control-net where it still retains C^1 continuity. Examples are presented using control-nets built from a variety of legacy triangulations with widely varying complexity. To demonstrate the modeler, we use a simplified meshing application which queries arbitrary locations on the surface to support either uniform or curvature-adaptive triangle refinement. The simple evaluation rules for surface coordinates and derivatives make the scheme extremely fast and robust. Since the input triangulation becomes the control-net of the subdivision surface, it is not necessarily an interpolant for the input data. Various approaches for making the surface interpolating are discussed, and this area remains one of active research.

1 Introduction

With increasing computing power pushing simulation technologies toward ever-higher fidelity, modeling applications in a broad spectrum of disciplines have sought ever-tighter integration with the underlying geometry. CAD-based modeling techniques have appeared in fields as diverse as Earthmoving,

Aerodynamic Vehicle Design, and Endoscopic Surgery. In the most tightly-coupled approaches, applications communicate with the underlying CAD-kernel either through vendor-provided "direct" CAD interfaces or vendor-neutral programming interfaces. In many situations, these CAD-based approaches are very attractive since they avoid translation errors and provide the modeling application with direct access to the latest revision of the fully detailed geometry. This provides important geometric consistency, since the same CAD models are used for creation, design, analysis and manufacturing.

Nevertheless, CAD-based modeling is not always either feasible or desirable. "Legacy geometry" is one example of an area which can cause difficulties for CAD-based simulations. Most simulation systems have to cope with models that date from pre-CAD eras, or even old CAD models that are somehow inconsistent when imported into current CAD systems. An aircraft geometry from two decades ago may only have been defined as a series of loftings, a faceted polyhedral mesh, or a surface triangulation that is considered coarse by the standards of modern simulations. A modern finite element solver may require position and curvature information *within* the triangles of a surface triangulation to achieve higher-order accuracy. CAD definitions for legacy models get lost, they get damaged by translation, or often they simply never existed.

In other situations, CAD-based modeling is simply not desirable. Using the CAD system to perform geometry queries and manipulation means that this system must be running and available to serve your application. This consumes CAD licenses that are typically both expensive and limited in number. Moreover, it requires the CAD system to be running either on the same platform as the simulation code or in direct communication with the platform running the simulation code. When running simulations on thousands of processors of massively parallel hardware, license consumption alone is an issue of real concern. And in addition, such high-powered computing hardware is typically protected by firewalls and other perimeter defenses that restrict communication, and accessing an non-local CAD-engine may be difficult.

All of these situations require a CAD-free surrogate for the queries that the CAD system usually fields in CAD-based simulations. In this work we investigate the use of Loop subdivision surfaces as an underlying surface model. Loop subdivision surfaces [3] have been extensively studied over the past two decades in the fields of computer graphics, animation [10] and scattered data surface reconstruction [4]. They can be used to represent globally smooth locally manifold surfaces of arbitrary global topology. The surface is defined by a control-mesh which is a watertight triangulation (simplicial polytope) constructed using some existing discrete representation of a model. While subdivision surfaces have simple rules for updating existing node locations

and performing new insertions, our use of them for geometric modeling is enabled by Stam's exact evaluation procedure for Loop subdivision surfaces [6, 7].

Any mesh generation or mesh adaptation application uses geometry evaluation and inverse evaluation to place points on the entity of interest. These applications read and hold the geometry (usually in the form of a Boundary Representation – BRep) and may support a large number of curve and surface primitives. Some applications depend on a geometry kernel (or Direct CAD interface) to deal with the complexities of the individual entities. These kernels directly provide evaluation functions that have one degree of freedom for curves (t) and two for surfaces (u, v). In either case, vertices are placed directly on geometry by direct evaluation or "snaps" (inverse evaluation) which can be performed by the use of derivatives of the forward evaluation. When using a subdivision surface as the underlying geometry, Stam's evaluation procedure answers the same type of requests by providing geometry for arbitrary parameter values. In the theoretical development, we describe a simplified implementation of this procedure and algorithms for determining the first and second derivatives of the surface. The capability to construct surface geometry and its derivatives at arbitrary parameter values provides a very fast and lightweight modeler and gives a simple and uniform view of the surface (i.e. one type – triangles) that is locally manifold. Moreover, unlike classical tensor-product spline or BRep surfaces, this approach is independent of the global complexity of the surface and is not constrained to rectangular patches.

An approximating subdivision surface does not necessarily pass through the nodes of the control-mesh (it is not an interpolant of this node set). The literature in computer graphics and scattered data reconstruction highlights several approaches to make the surface interpolating, and displacement [5], multi-resolution [9] and adaptation [4] are addressed in our discussion.

2 Surface Parametrization by Subdivision

2.1 Subdivision Surfaces

The main idea behind subdivision surfaces is to represent a smooth surface by a *control-mesh*. This mesh can be enhanced by various refinement schemes, and in the limit of infinitely many refinements it approaches the "true" surface. In practice these refinements are repeated until the surface is sufficiently fine. This is the approach used in applications from computer graphics and visualization where subdivision surfaces are most commonly employed.

A distinction is made between *interpolating* and *approximating* subdivision schemes. In the interpolating schemes the nodes of the control-mesh stay fixed during the refinement. The resulting surface is then an interpolant of the

nodes in the control-mesh, which is an attractive property. However the surfaces generated with these schemes are sometimes not sufficiently smooth, and the convergence to the "true" surface is relatively slow. The approximating schemes, on the other hand, generally do not produce surfaces which interpolate the control-mesh, but they do result in smooth surfaces with continuous second derivatives everywhere except for at certain isolated points.

When considering the functions essential to a geometry kernel (when used for mesh construction and adaptation) there are two basic requirements. The first is the ability to evaluate, that is, given parameters within the geometric support produce 3D coordinates of the point on the entity. In the case of subdivision surfaces, the support is the control-mesh triangles and the parameters are the barycentric coordinates within each triangle. The second required function is the ability to find the nearest point on the geometry to a given set of coordinates. This "snap" (inverse evaluation) is usually cast as a minimization problem and is efficiently solved by the use of an iterative Newton solver. In order to construct this solution the forward evaluation is required along with both first and second derivatives at the evaluated point. The resultant position may not be in the current control-mesh triangle, which will be reflected in the barycentric coordinates (being outside the valid range). The neighboring control triangle opposite the vertex with the largest negative weight is set as the current support and the Newton solver is restarted within this triangle. This converges quickly and is robust requiring little additional intervention (but can be sensitive to starting locations when the geometry is concave).

In this work we are concerned about second derivatives, both for computing curvatures and for applying the Newton solver (as described above). Therefore our current focus is on approximating schemes. In Section 3.3 we discuss various alternatives for obtaining an interpolating model.

2.2 Spline Curves

For one-dimensional curves the subdivision schemes are particularly simple. The control-mesh is a polygon and the limiting procedure gives a smooth curve, see Fig. 1 for an example. At each refinement step, the polygon edges are divided in two. In this example we use an approximating scheme where the inserted midpoint at level $j+1$ is simply the average of the two neighboring nodes at level j,

$$\boldsymbol{x}_{2i+1}^{j+1} = (\boldsymbol{x}_i^j + \boldsymbol{x}_{i+1}^j)/2. \tag{1}$$

The original nodes are moved to a linear combination of their previous locations as well as their neighbors',

$$\boldsymbol{x}_{2i}^{j+1} = (\boldsymbol{x}_{i-1}^j + 6\boldsymbol{x}_i^j + \boldsymbol{x}_{i+1}^j)/8. \tag{2}$$

The limiting curve is a cubic spline and it is C^2 continuous. We note that the subdivision rules (1) and (2) are local, and therefore a node in the

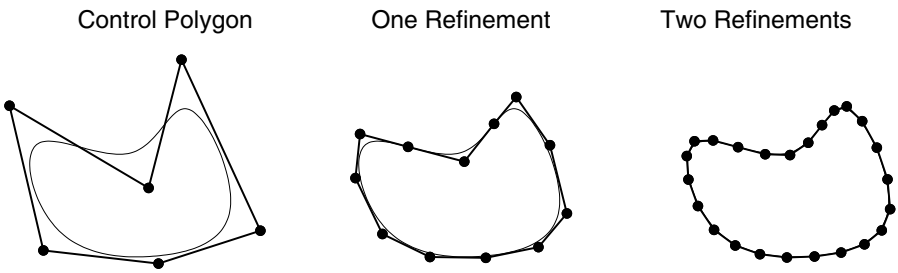

Fig. 1. Subdivision of a polygon using an approximate scheme. The limiting curve, which is the "true" smooth curve being represented, is shown with thin line.

control-polygon only affects the curve in a neighborhood around the node. As we pointed out above, the refined polygons will approach the "true" curve in the limit, but it is also possible to compute the limiting positions of the nodes at any level by the simple expression

$$\boldsymbol{x}_i^{j,\infty} = (\boldsymbol{x}_{i-1}^j + 4\boldsymbol{x}_i^j + \boldsymbol{x}_{i+1}^j)/6. \tag{3}$$

where $\boldsymbol{x}_i^{j,\infty}$ denotes the limiting position of point i at refinement level j.

2.3 The Loop Subdivision Scheme

For surfaces, various types of elements can be used for the control-mesh. While the original approximating subdivision schemes of Catmull and Clark [1] were based on quadrilateral meshes, we have worked exclusively with the Loop scheme [3] for triangular meshes. The control-mesh is defined by a set of nodes as well as a triangulation. In a refinement step, each triangle is split into four, and we need rules for the location of the new midpoint of the edges as well as the new location of existing nodes, see Fig. 2. The number of neighbors of existing nodes is k (the valance), and the value we use for the weight ω was proposed by Loop [3], see Warren [8] for a simpler alternative.

Again we can compute the limiting location for the nodes at any level of the refinement. The formula is the same as the one to advance one level (right plot in Fig. 2), but with ω replaced by $1/(k + 3\omega/8)$, see Fig. 3. We can also compute the tangent vectors (and from them the surface normal) using the expressions

$$\boldsymbol{t}_1 = \sum_{i=0}^{k-1} \cos\frac{2\pi i}{k}\boldsymbol{x}_i, \quad \boldsymbol{t}_2 = \sum_{i=0}^{k-1} \sin\frac{2\pi i}{k}\boldsymbol{x}_i \tag{4}$$

where $\boldsymbol{x}_i$ is the ith neighbor of the considered node.

Figure 4 shows an example of a control-mesh with the limiting Loop subdivision surface. Note again that this is an approximate scheme and the nodes of the control-mesh are not located on the surface. The surface is C^2 continuous

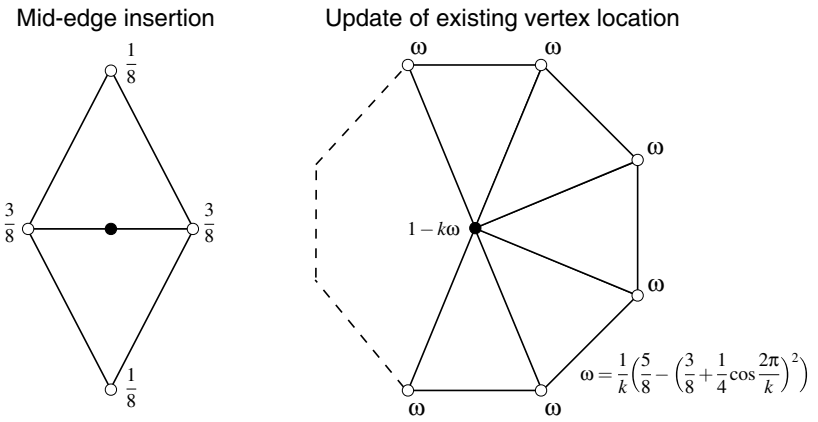

Fig. 2. The Loop subdivision scheme. The figures show the weights used to compute a new mid-edge point (left) and how to update existing nodes (right).

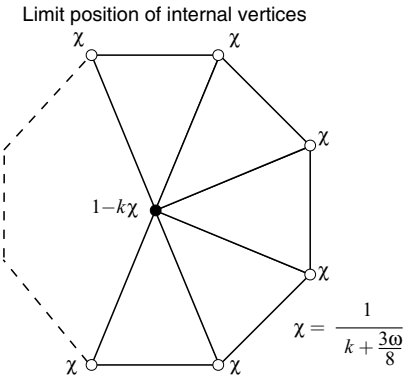

Fig. 3. The node positions in the limit of infinitely many refinements using the Loop scheme. This effectively moves the nodes of the control-mesh to the "true" surface.

everywhere except at irregular nodes (nodes that do not have six neighbors). The subdivision process converges fast and for visualization purposes a few refinements are sufficient to obtain a good approximation of the surface.

It should be noted that the subdivision scheme defines a hierarchy of "control-nets" each having the same limiting result. The left part of Fig. 4 results in the middle part after applying one subdivision operation. This set of vertices can be considered a new control-polygon and then results in the right part of Fig. 4 after another application of the operator.

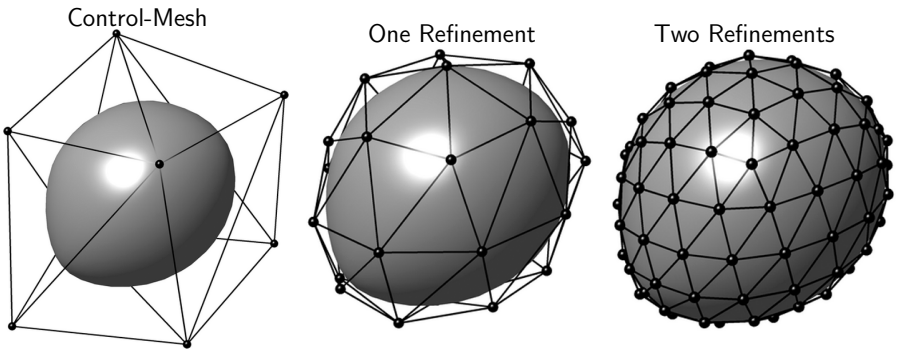

Fig. 4. Subdivision using the Loop subdivision scheme. The figures show the limiting surface together with the initial control-mesh, and two refined meshes.

2.4 Hard vs. Soft Edges

The Loop scheme produces smooth surfaces everywhere, including at the boundaries of the mesh (Fig. 5, left). Sometimes it is desired to have sharp boundaries ("creases"), which requires special treatment in the subdivision process. We tag the edges of the control-mesh corresponding to the sharp boundaries as *hard edges*. In the refinement of these nodes we use the one-dimensional spline scheme (1)-(3) instead of the Loop scheme. This will represent a smooth boundary curve with a jump in the tangent plane on the two sides of the curve (Fig. 5, middle).

In a similar way, the vertices of the boundaries can be tagged as *hard vertices* to produce a jump in the tangent along the boundary curve. Since a vertex can not be subdivided, the scheme for hard vertices is simply to leave them fixed at their original positions (Fig. 5, right).

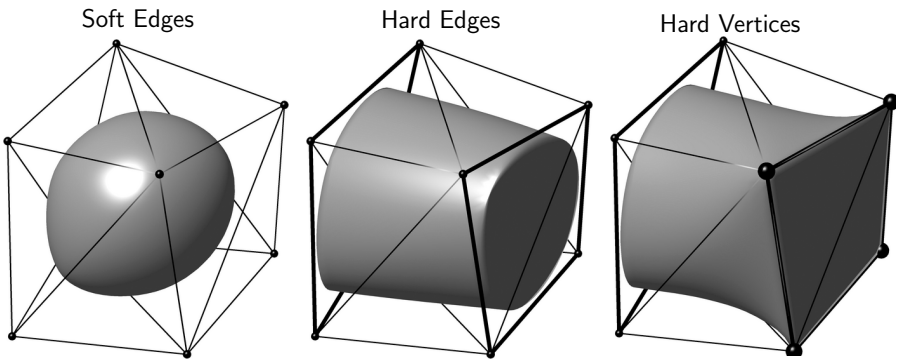

Fig. 5. Hard and soft edges and vertices. The hard edges are shown in thick lines and the hard vertices are shown with large spheres.

3 Parametrization and Evaluation

Using the subdivision scheme and the limiting expressions, we can evaluate the surface properties at the control nodes and at the nodes of any refined mesh. However, in order to be useful in a general geometry setting we need a parameterization of the surface and the ability to evaluate at arbitrary locations. There is not much literature on the parameterization of subdivision surfaces, presumably because the refinement process fulfills most all the needs in computer graphics and animation. Stam showed how to evaluate the Catmull-Clark surfaces for arbitrary parameter values [7], and later extended this analysis to Loop surfaces [6]. We use a simplified form of these methods, and we describe the parameterization for regular triangles in this section, while the next section discusses arbitrary triangulations.

The parameterization of spline curves are well-known, and for a "smooth" edge segment (no hard vertices or end points) we number the four consecutive polygon nodes $\boldsymbol{x}_{i-1}, \ldots, \boldsymbol{x}_{i+2}$ and introduce a parameter value $t \in [0, 1]$ between node i and $i+1$. The explicit expression for the spline curve is then:

$$\boldsymbol{x}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_{i-1} \\ \boldsymbol{x}_i \\ \boldsymbol{x}_{i+1} \\ \boldsymbol{x}_{i+2} \end{bmatrix}$$

Similar expressions are available for edge segments neighboring hard vertices (interpolating control points), see de Boor [2] for details.

3.1 Parametrization – Regular Triangles

In a regular triangle, the subdivision surface reduces to the common box splines for which analytical expressions are available. Each node has exactly six neighbors and the total number of nodes in the triangle or adjacent to it is 12 (Fig. 6). A natural parameterization is given by the local barycentric coordinates in the triangle (right plot). Since $u + v + w = 1$ we can eliminate one of these coordinates, and we choose to parameterize by v, w.

For evaluation at the local coordinates v, w, we compute all monomials and collect in a column vector:

$$\boldsymbol{c}(v, w) = (1, v, w, v^2, vw, w^2, v^3, v^2w, vw^2, w^3, v^4, v^3w, v^2w^2, vw^3, w^4)^T \quad (5)$$

These basis functions are mapped to a Lagrangian basis by multiplication by the matrix $\boldsymbol{\phi}$ below. We obtained this matrix from the expressions in [6] after substituting $u = 1 - v - w$, renumbering, identifying coefficients, and writing in matrix form.

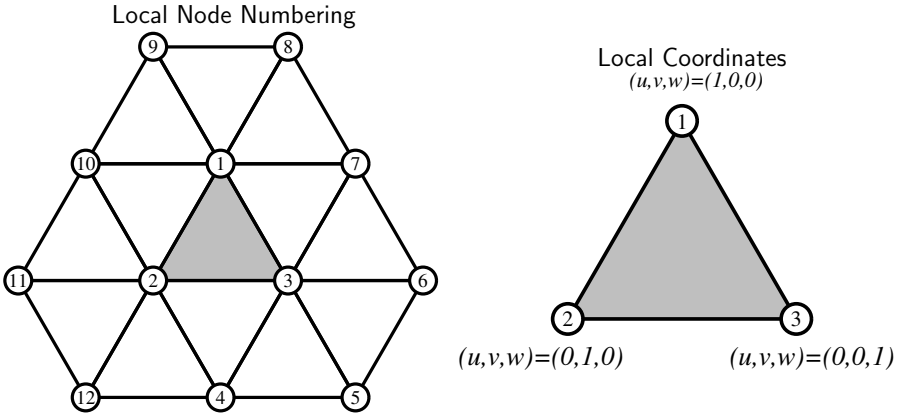

Fig. 6. The local node numbering for regular triangles (left) and the local barycentric coordinates u, v, w where $u + v + w = 1$ (right).

$$\boldsymbol{\phi} = \frac{1}{12}\begin{bmatrix} 6 & 0 & 0 & -12 & -12 & -12 & 8 & 12 & 12 & 8 & -1 & -2 & 0 & -2 & -1 \\ 1 & 4 & 2 & 6 & 6 & 0 & -4 & -6 & -12 & -4 & -1 & -2 & 0 & 4 & 2 \\ 1 & 2 & 4 & 0 & 6 & 6 & -4 & -12 & -6 & -4 & 2 & 4 & 0 & -2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 6 & 6 & 2 & -1 & -2 & 0 & -2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & -2 & -1 \\ 1 & -2 & 2 & 0 & -6 & 0 & 2 & 6 & 0 & -4 & -1 & -2 & 0 & 4 & 2 \\ 1 & -4 & -2 & 6 & 6 & 0 & -4 & -6 & 0 & 2 & 1 & 2 & 0 & -2 & -1 \\ 1 & -2 & -4 & 0 & 6 & 6 & 2 & 0 & -6 & -4 & -1 & -2 & 0 & 2 & 1 \\ 1 & 2 & -2 & 0 & -6 & 0 & -4 & 0 & 6 & 2 & 2 & 4 & 0 & -2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & -1 & -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

Finally we create a node array with the 12 local nodes according to the numbering in Fig. 6 (a 3-by-12 matrix):

$$\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4, \boldsymbol{x}_5, \boldsymbol{x}_6, \boldsymbol{x}_7, \boldsymbol{x}_8, \boldsymbol{x}_9, \boldsymbol{x}_{10}, \boldsymbol{x}_{11}, \boldsymbol{x}_{12}) \tag{7}$$

Using these expressions the surface location is a product of these matrices and the vector $\boldsymbol{c}(v, w)$:

$$\boldsymbol{x}(v, w) = \boldsymbol{x}\boldsymbol{\phi}\boldsymbol{c}(v, w) \tag{8}$$

The first and second derivatives are obtained in a similar way by differentiating the simple monomials in $\boldsymbol{c}(v, w)$.

3.2 Parameterization – Irregular Triangles

The method Stam suggested for irregular triangles is based on the fact that the subdivision process introduces new triangles with regular neighborhoods

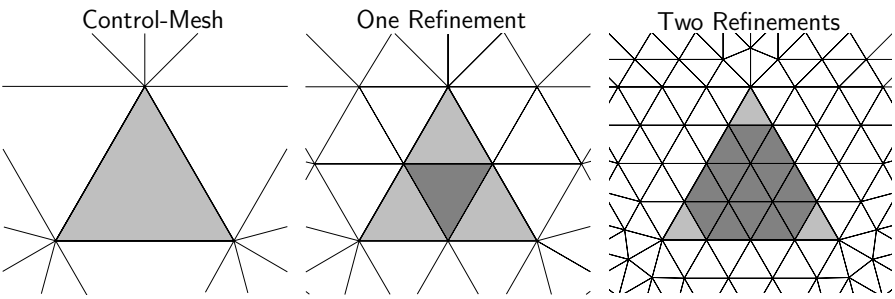

Fig. 7. Refinement of a triangle with irregular nodes. The triangle under consideration is shown in light gray, and the triangles with regular neighborhoods are shown in dark gray.

that can be evaluated using the box spline expressions (see Fig. 7). Clearly, after a sufficient number of refinements any point in the triangle can be covered by a regular triangle and evaluated using (8). However, for points close to an irregular node a very large number of refinements might be required. Stam solved this problem by introducing an eigen-decomposition of the refinement matrix, and was able to evaluate arbitrary close to irregular nodes in a constant number of operations. In our work, we choose a simpler solution in which we refine until the requested position is covered by a regular triangle or until some maximum subdivision depth is reached. If the maximum depth is reached (usually around 25 subdivisions), then we *nudge* the requested position onto the center triangle (which will be regular) and subdivide once again. At this point the evaluation using a regular neighbor can be applied. We use this special treatment for points close to the irregular nodes and in the vicinity of hard edges or vertices.

The crucial step of our scheme is to subdivide around the point v, w, identify the new triangle and the new parameter values v, w, and repeat recursively until the neighborhood is regular. During this refinement we use a local representation of the relevant triangles only. Figure 6 shows our node numbering for this local submesh. It is constructed by selecting a starting vertex in the triangle (this also sets the new v, w). The other two vertices are selected in a right-handed manner. The fourth vertex is selected as the node opposite the first. The rest are defined by winding around the triangle in a right-handed fashion and collecting all of the next level vertices that support any triangle that touches the target. This neighborhood is fully defined by the valance of each of the vertices of the target triangle (3 integers) and the positions of each of the neighborhood nodes, where the number of nodes is $k_1 + k_2 + k_3 - 6$. This small memory footprint does not overwhelm the stack as the recursive procedure continues.

The next level is reached by selecting the appropriate subtriangle (as seen in the center of Fig. 7) based on v, w. A new neighborhood is specified by

Algorithm 1: Subdivision Surface Evaluation

Description: Evaluate a subdivision surface for arbitrary parameter values
Input: Control-mesh, triangle t, local coordinates v, w
Output: Surface location $\boldsymbol{x}$ and its first and second derivatives $\boldsymbol{dx}, \boldsymbol{ddx}$

function $[\boldsymbol{x}, \boldsymbol{dx}, \boldsymbol{ddx}] =$ **loopeval**$(t, v, w, \mathit{depth})$

if t regular
    Evaluate surface location and derivatives using (8)
else
    Subdivide t and all triangles sharing its nodes
    if $\mathit{depth} \leq 25$
        $t' =$ new triangle covering v, w
        $v', w' =$ new local coordinates in t'
    else
        $t' =$ center subtriangle
        $v', w' =$ new local coordinates (on edge of subtriangle)
    end if
    $[\boldsymbol{x}', \boldsymbol{dx}', \boldsymbol{ddx}'] =$ **loopeval**$(t', v', w', \mathit{depth} + 1)$
    Inverse mapping: $\boldsymbol{x} = \boldsymbol{x}',\ \boldsymbol{dx} = \pm\boldsymbol{dx}'/2,\ \boldsymbol{ddx} = \boldsymbol{ddx}'/4$
end if

Fig. 8. High-level pseudo-code for evaluation of subdivision surfaces at arbitrary parameter values.

applying the subdivision rules (as seen in Fig. 4) where the *hard edges* are defined during the splitting of existing *hard edges*. *Hard vertices* need not be explicitly marked because they are defined when the number of *hard edges* touching a node is greater than two. The valence of the 3 subtriangle vertices, the neighborhood coordinates, as well as the new v, w are passed on to the next recursion level. Finally, the computed derivatives are adjusted because of the mapping between the new and the old parameters v, w. Our complete algorithm is described in pseudo-code in Fig. 8.

3.3 Generating an Interpolating Result

The fact that the Loop scheme is not interpolating might be a major problem in some applications. The subdivision schemes that are interpolating [10] produce less smooth surfaces than the Loop scheme, and we would therefore loose the ability to provide geometry "snaps" (solving with Newton's scheme across the triangles). This would make the scheme unusable in our context.

An alternative method that we have used with some success is to solve for new node locations in the control-mesh such that the "true" Loop surface interpolates the original nodes. This simply amounts to solving

$$S^{\infty}(\boldsymbol{x}_{\text{interp}}) = \boldsymbol{x} \tag{9}$$

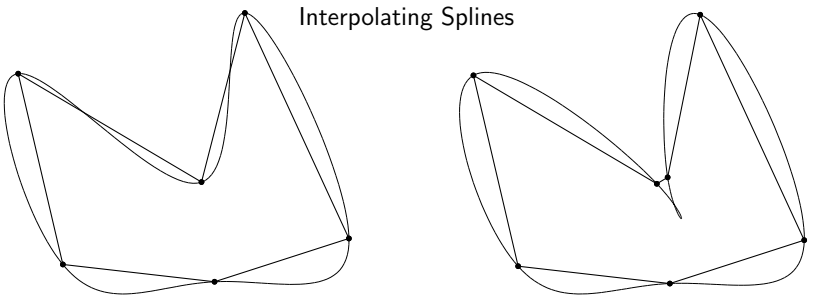

Fig. 9. Splines computed from new control points such that the curve interpolates the original points. The right plot shows that the resulting curve is not always well-behaved. In this case it has a cusp.

for the new nodes $\boldsymbol{x}_{\text{interp}}$, where $\boldsymbol{x}$ are the original node locations and S^∞ the linear operator that produces the limiting node locations. For the Loop scheme, S^∞ is essentially the stencil in Fig. 3, except for hard edges/vertices. The linear system of equations (9) is well conditioned and can be solved in just a few iterations with a Krylov subspace solver.

Our initial experiments show that the surfaces generated by $\boldsymbol{x}_{\text{interp}}$ are well-behaved for uniform control-meshes, see left plot in Fig. 9 for an example of a spline. However, for more general control-polygons, the resulting curve might have cusps (right plot). A similar example for surface meshes using the Loop scheme is shown in Fig. 10.

One correction algorithm that shows more success is to post-process the results in a manner similar to that in [5]. By maintaining v, w and the parent triangle for any vertex, it is possible to adjust the position by applying the linear weights times the displacement of the control-net to the limiting surface. This insures that the adjusted surface passes through the control-net at the expense of C^1. Approaches, such as multi-resolution [9] and adaptation [4] promise both smoothness and interpolation, and these are subject of on-going investigations.

4 Results

This work is aimed at investigating the utility of Loop subdivision surfaces as surrogate geometry. In order to demonstrate their use in this role, we have constructed a very simple surface mesh refinement application. Within this application, all geometry constructors (requests for new points on the surface) are based on the version of Stam's evaluation procedure described in Algorithm 1. We simply provide the local parameterization at the desired vertex insertion point, and this algorithm returns the *xyz*-triple for the constructed vertex. An analogous constructor is central to virtually all geometry/CAD-based mesh generation systems, and is the key ingredient in the use of Loop

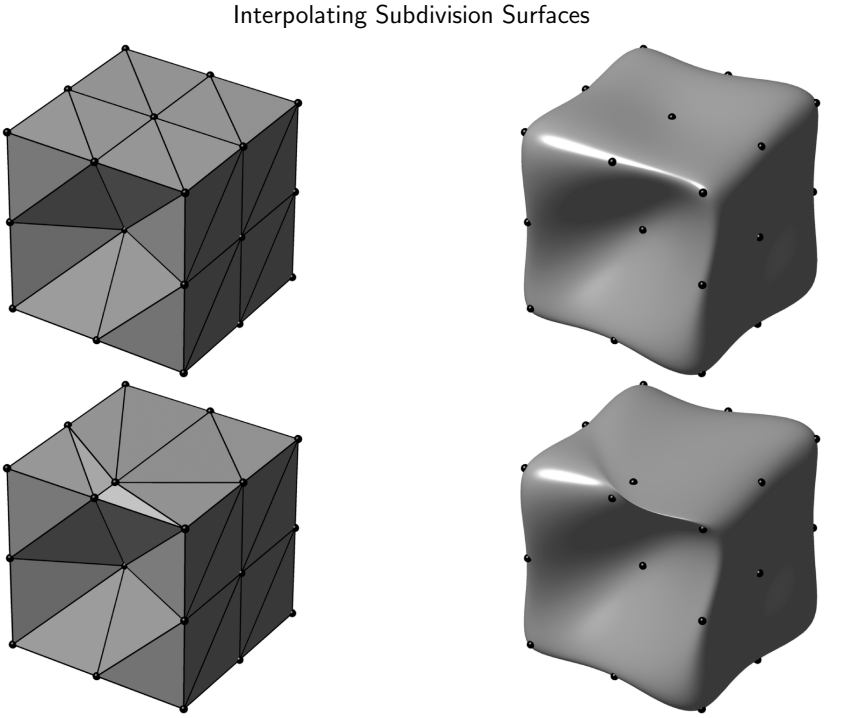

Fig. 10. Loop subdivision surfaces computed from new control points such that the surface interpolates the original points. The left plots show (original) control-meshes, and the right plots show the surfaces. The bottom example shows a (probably undesired) negative effect.

subdivision surfaces as a CAD-free modeler. Algorithm 1 also provides first and second derivatives of the surface at the evaluation location.

Aside from this central feature, the mesh refinement application demonstrated in this section is very simplistic and has relatively few features of merit. It was written simply as a platform for testing/demonstrating the constructor, and is not (in any way) intended as a viable mesh generator. Mesh refinement proceeds by performing a set of centroidal insertions within a set of triangles, followed by an edge-swap pass which performs swaps based upon the evaluation of a *maxmin* predicate. It is important to recognize that the tessellations shown are not generated in the traditional subdivision construction manner as so often seen in the literature.

Each example begins with the control-net for the underlying subdivision surface read in from a legacy triangulation file. No coarsening of the legacy triangulation is performed. For the purpose of demonstration, a simple dihedral-angle criteria on triangles in this control-net is used to establish hard edges.

Figure 11 shows the effect of hard edges and two example meshes generated using this adaptation application. The legacy geometry in this figure is the

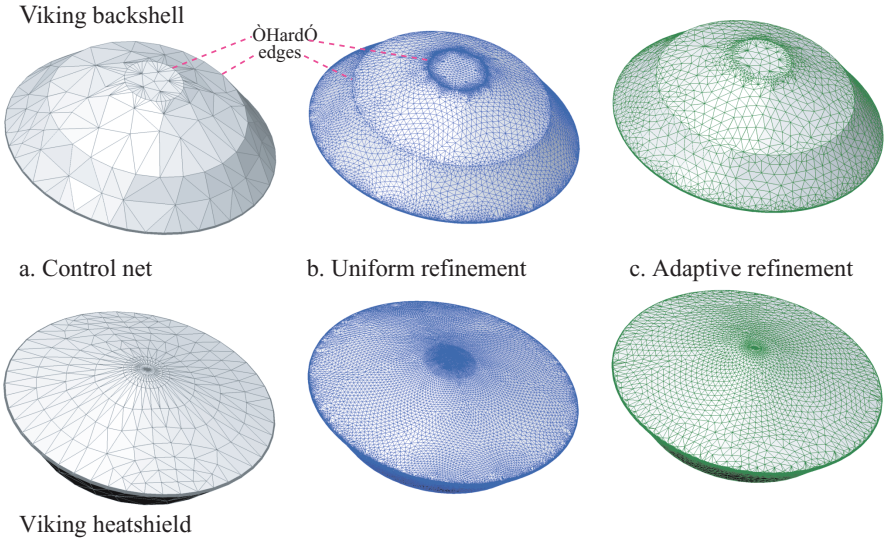

Fig. 11. Control-net, uniform and adaptive tessellations of the Viking spacecraft aeroshell geometry. The upper row of images shows the backshell (back) and the lower shows the heatshield (front). Both tessellations demonstrate preservation and refinement of hard edges in the input mesh.

aeroshell of NASA's Viking spacecraft which was the culmination in a series of exploratory missions to Mars from 1964-1975, and pre-dates much of the modern CAD industry. The upper row of images in Fig. 11 shows the view from the back, while the lower shows the geometry from the front. The back of the aeroshell is a truncated bi-conic. The control-net shown in Fig. 11a is composed of 1700 triangles and the dihedral-angle criteria identifies one circle of hard-edges at the sharp transition between the two conics and another at junction with the flat backface.

The simple mesher described above was run using both uniform refinement and curvature adaptive refinement. In the adaptive example, refinement was triggered using the maximum local surface curvature at each triangle centroid scaled by the triangle's area. Surface curvature was evaluated making use of Algorithm 1's ability to return the first and second derivatives at any point on the surface. The uniformly refined mesh has 35000 triangles while the adaptive triangulation has 9100. The triangulation algorithm produced approximately 7500 triangles-per-second on a 2Ghz CPU, however no serious attempt has been made to optimize the mesher or subdivision surface library.

While the Viking aeroshell is an example of legacy geometry that pre-dates CAD-based modeling, Fig. 12 shows and example of geometry that comes from a 3D scanner and similarly has no underlying CAD definition. The figure shows the control-net (45k triangles) and a curvature adapted triangulation (105k triangles) for an irregularly shaped piece of foam. The 3D scan produced a

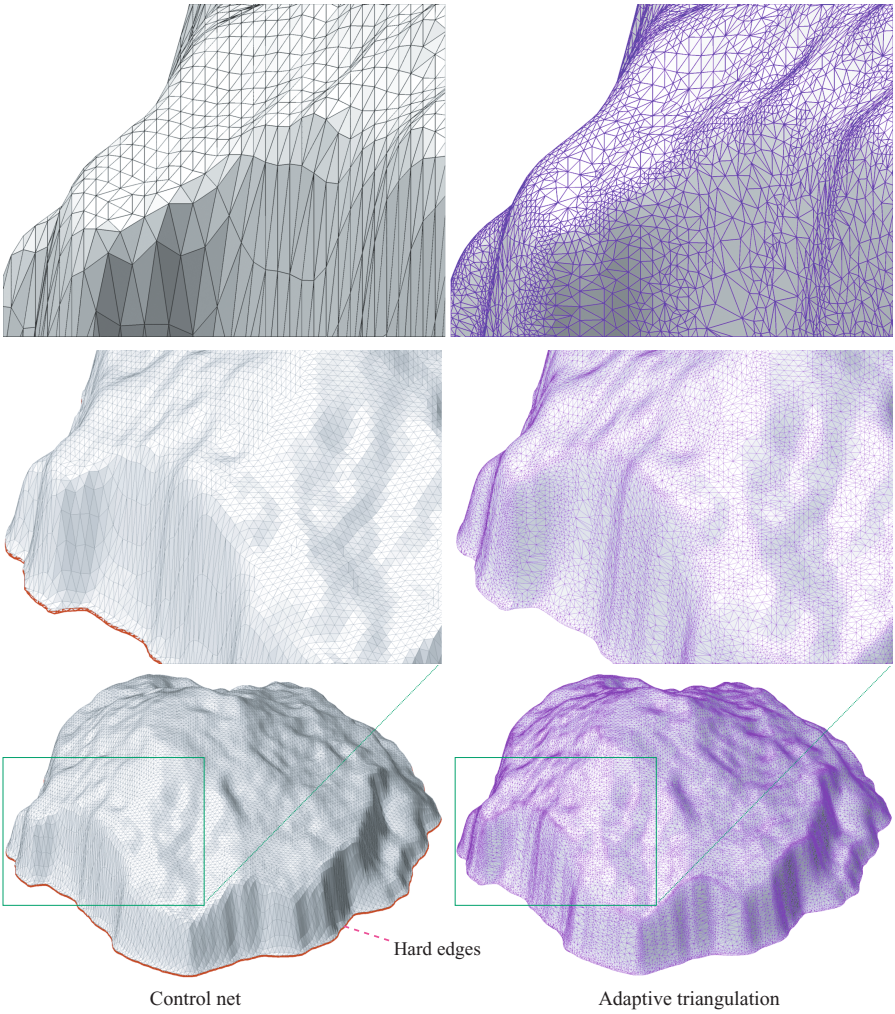

Fig. 12. Control-net (45k triangles) and curvature-adaptive triangulation (105k triangles) for irregularly shaped foam piece from high-resolution 3D scan. Inset frames show details of control-net and surface triangulation.

STL triangulation file and this triangulation (without decimation) was used as the control-net for the subdivision surface. The simple adaptive mesher used in the previous example was then run using this surface as an underlying geometry. While the original scan is at quite high resolution, the enlargements of the control-net shown in Fig. 12 (inset top-left) exhibit substantial faceting due to the small characteristic feature size on the irregular surface. The adaptive triangulation shown at the right of Fig. 12 offers improved resolution of regions with high curvature. Hard edges are indicated on the figure and form a single closed loop around the lower perimeter of the piece.

Figure 13 shows a final example with the control-net and adaptive triangulation generated on a model of a pig. The control-net in this case is quite coarse with only about 7000 triangles. Despite this, the model is remarkably detailed and includes all major anatomical features as well as details like eyelids, hoofs, nostrils, and jowls. These features are even more apparent in the adapted triangulation shown in the center and two inset frames of Fig. 13. This adapted triangulation was built using the simple mesher described earlier with curvature-sensitive refinement and includes 244k triangles. In addition to the obvious features, refinement reveals more subtle detail captured by the control-net. The refined triangulation reveals the pig's shoulder blades, pelvis, hamstring definition, and additional facial detail. As with the earlier cases, the goal of this example is not to show the "best" triangulation for this geometry, but to illustrate the use of this very coarse legacy triangulation as a control-net definition and to mesh using the Loop subdivision surface defined by that control-net.

5 Conclusions

This work outlined a method for using existing surface triangulations as underlying geometric models for the construction of more highly refined models complete with local surface derivatives and curvature information. The method is based upon the construction of a Loop subdivision surface using the legacy triangulation at full resolution. The presentation outlined a simplified surface evaluation procedure based on Stam's exact method for Loop surfaces. The presentation of this simplified approach highlighted the construction of surface coordinates, derivatives, and curvatures for evaluations at arbitrary locations on the surface. The implementation also provides the ability to tag *hard*-edges (and therefore *hard*-vertices) in the imported geometry to preserve creases and points using one-dimensional cubic-spline scheme which preserves C^2 continuity along hard-edges. Away from hard-edges and vertices, the Loop surface is C^2 continuous everywhere except right at irregular vertices in the control-net, where it is still C^1.

The modeler was demonstrated using a simplistic meshing application which queries arbitrary locations on the surface to support either uniform or curvature-adaptive triangle refinement. Curvature information for adaptation parameter was obtained through direct evaluation on the subdivision surface using the algorithm presented in section 3. Examples were presented using control-nets built from a variety of legacy triangulations with widely varying complexity. The ability to query the surface at arbitrary locations and quickly find surface derivatives and curvature information makes this modeler very attractive for users of finite element analysis methods that require this information to achieve higher-order accuracy.

The simple evaluation rules for surface coordinates and derivatives make the modeler extremely fast and robust. Even with no special effort to optimize

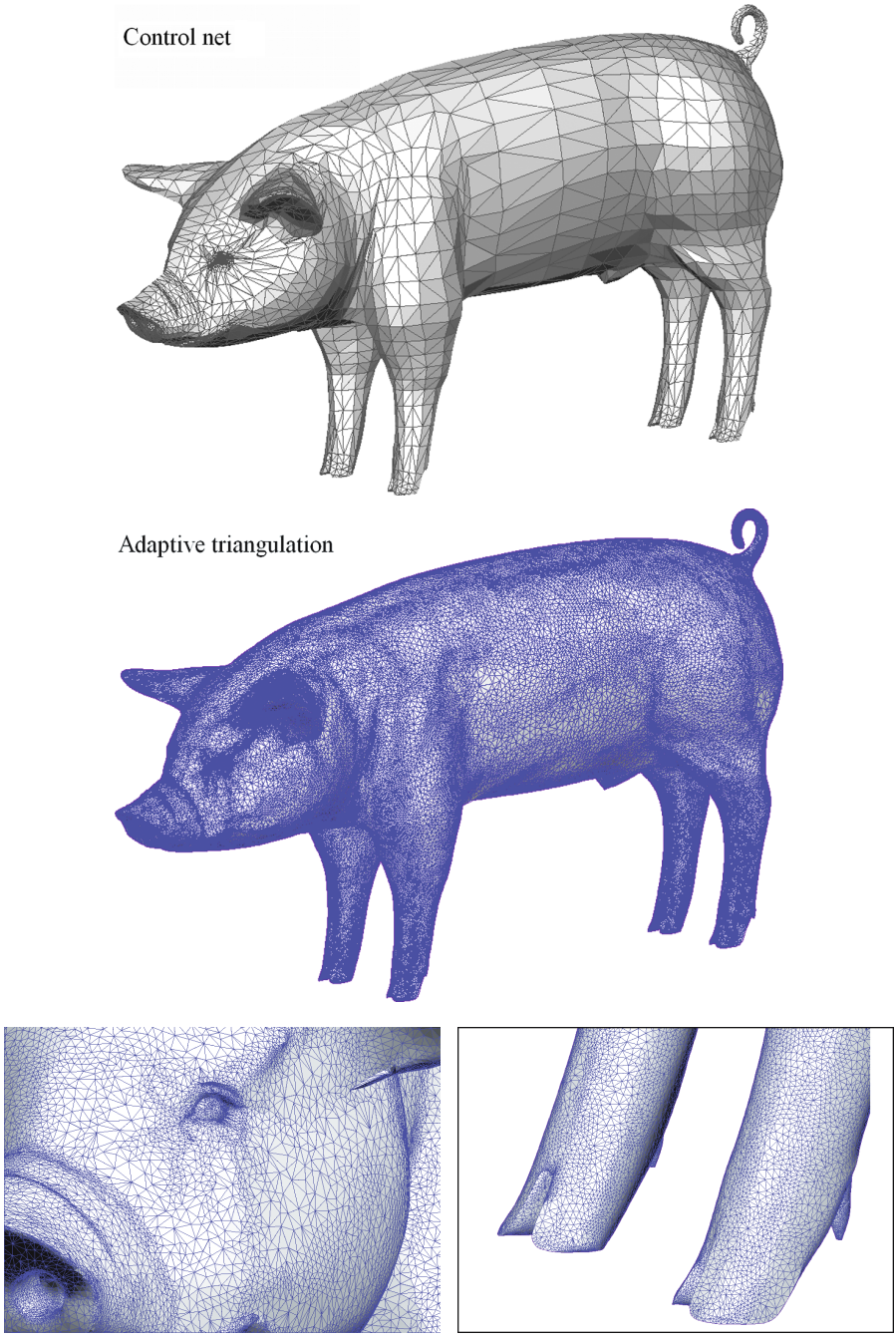

Fig. 13. Control-net and curvature-adaptive triangulation for pig geometry. The control-net contains about 7000 triangles, while the adaptive triangulation contains about 244k. Inset frames at the bottom show details of facial structure and hoofs.

the implementation, surface triangulations with nearly 500k triangles can be generated in under a minute on a single processor desktop machine. Immediate plans will focus on experiments with the construction of the subdivision surface itself. Since the input triangulation becomes the control-net of the subdivision surface, it is not, in general, an interpolant for the input data. However, the deviation is small when the control-net resolves the geometry and hard edges are tagged appropriately. For example, the distances between the control-net nodes and the subdivision surfaces in the three examples in Section 4 are in average less than 0.06% of the geometry size, and the maximum deviations are less than 0.2%. Research examining various approaches for making the surface exactly interpolating is ongoing.

Acknowledgements

The authors wish to thank the Boeing Company (technical monitor Mori Mani) for their support in this effort. This assistance has enabled the generation of a surface modeling geometry software kernel as part of **TURIN** – the **T**etrahedral **U**nstructured **R**emeshing **IN**terface. This software library was used to generate the figures seen in the results section.

References

1. E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.
2. C. de Boor. *A Practical Guide to Splines*. Springer, 2001.
3. C. T. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, Department of Mathematics, University of Utah, August 1987.
4. M. Marinov and L. Kobbelt. Optimization methods for scattered data approximation with subdivision surfaces. *Graphical Models*, 67(5):452–473, 2005.
5. A. Lee H. Moreton and H. Hoppe. Displaced subdivision surfaces. In *ACMSIGGRAPH '2000 CDROM Proceedings*, pages 85–94, 2000.
6. J. Stam. Evaluation of loop subdivision surfaces. In *SIGGRAPH '98 CDROM Proceedings*, 1998.
7. J. Stam. Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In *Computer Graphics Proceedings, Annual Conference Series*, pages 395–404, July 1998.
8. J. Warren. Subdivision methods for geometric design. Unpublished manuscript, November 1995.
9. D. Zorin. *Subdivision and multiresolution surface representations*. PhD thesis, Caltech, Pasadena, California, 1997.
10. D. Zorin and P. Schröder. Subdivision for modeling and animation. SIGGRAPH 2000 Course Notes, 2000.