Adaptive Subdivision of Cubic Subdivision Surfaces

Category: Research

Abstract

Catmull-Clark subdivision scheme provides a powerful method for building smooth and complex surfaces. But the number of faces in the uniformly refined meshes increases sharply with respect to subdivision depth. This paper presents adaptive subdivision techniques as a solution to this problem. The adaptive subdivision process is driven by labels of mesh vertices or error between the mesh faces and the limit surface. A mesh face is subdivided only to meet precision requirement or to avoid crack. New vertices are produced by the same Catmull-Clark scheme to ensure that the limit surface of the adaptively refined meshes is the same as the original limit surface. The resulting meshes are crack-free, and all the faces are quadrilaterals. Test results show that the number of faces in the resulting meshes is significantly reduced. The proposed techniques can be used for cubic Doo-Sabin subdivision surfaces, non-uniform cubic subdivision surfaces and combined subdivision surfaces as well.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling –Curve, surface, solid, and object representations.

Additional Keywords: Adaptive refinement, Mesh Generation, Subdivision surfaces

1 Introduction

Subdivision surfaces have become popular recently in graphical modeling, animation and CAD/CAM [6, 17] because of their stability in numerical computation, simplicity in coding and, most importantly, their capability in modeling/representing complex shape of arbitrary topology. Research work for subdivision surfaces has been carried out in several important areas such as surface evaluation [20], surface trimming [17], boolean operations [2], and mesh editing [21]. However, the work is far from being complete yet; for instance, research work is still needed for surface tessellation and shape design. The purpose of this paper is to study subdivision surface tessellation problem. New techniques that will significantly reduce the number of faces in the resulting meshes will be presented. These techniques would improve the efficiency of subsequent data communication, surface operations (trimming/intersection) and downstream applications (e.g., finite element analysis) for subdivision surfaces significantly.

Research work for reducing the number of faces in a mesh can be classified into three directions. *Mesh simplification* [1, 8, 9, 10, 11, 16, et al] is the most popular among the three directions over the past decade. The aim is to remove over-sampled vertices and produce approximate meshes with various levels of detail. The second direction focuses on approximating the limit surface by surfaces that we know of, such as a displaced subdivision surface [14] or Nurbs patches [18]. The last one is to apply adaptive refinement schemes to subdivision surfaces. Kobbelt has presented methods for adaptively refining triangle meshes for $\sqrt{3}$-subdivision surfaces [13], and balanced nets for interpolatory subdivision surfaces [12].

The techniques presented in this paper belong to the third direction. Two adaptive subdivision techniques: label-driven approach and error-driven approach, are presented here. These techniques will be presented for the Catmull-Clark subdivision process [3]. But they can be used for cubic Doo-Sabin [7], non-uniform cubic [19], and combined [15] subdivision surfaces as well. One only needs to replace the Catmull-Clark vertex-computing formulas with the corresponding vertex-computing formulas of these schemes. The reason that we use the Catmull-Clark subdivision scheme here is because it is a frequently used subdivision surface generation technique and it is simple for the presentation so we can focus on the adaptive subdivision process itself rather than the tedious vertex computation process during the mesh generation process.

Our work is inspired by [4] and [12] which use unbalanced subdivision and "Y"-element to avoid crack, respectively. The idea of [4] is followed in the label-driven approach here, that is, labels are assigned to the faces and vertices of an initial mesh, and used to control the subdivision process. While [4] works on an $m \times n$ rectangular mesh or its subset, the techniques presented in this paper can work for a mesh of arbitrary topology. This is achieved by using a greedy algorithm to eliminate illegal vertex labels in the initial mesh. Building smooth transitions between regions with different subdivision levels is through the unbalanced subdivision process. The concept is similar to the unbalanced subdivision presented in [4], but the details are quite different; an auxiliary structure has to be introduced so that the vertex-computing formulas can still be used in the subsequent subdivision steps. In the error-driven approach, instead of assigning a label to each face and each vertex at the outset, and then using the labels of the vertices to drive the adaptive subdivision process, subdivision is performed for a face only when the error condition is not satisfied or to avoid crack. The testing of the error condition is performed before each adaptive subdivision iteration. By dynamically testing if a subdivision step is needed before each iteration, one can avoid unnecessary subdivision steps that would not be possible for the label-driven approach and, consequently, further reduce the number of faces generated in the resulting mesh.

2 Label-Driven Adaptive Subdivision

Given a control mesh of arbitrary topology, the goal here is to construct a sequence of adaptively refined meshes that would converge to the same Catmull-Clark subdivision surface, but with much fewer vertices and faces than one would get in the traditional Catmull-Clark subdivision process. The mesh refining process will be driven by labels of mesh vertices. We need a few definitions first.

The given control mesh will be referred to as $\mathbf{M}_0$ if all of its faces are quadrilaterals. Otherwise, $\mathbf{M}_0$ refers to the result of applying the Catmull-Clark subdivision one time. In either case, all faces of $\mathbf{M}_0$ are quadrilaterals. For each positive integer k, $\mathbf{M}_k$ refers to the result of applying the Catmull-Clark subdivision k times on $\mathbf{M}_0$. $\mathbf{M}_k$, $k \geq 0$, and the limit surface $\mathbf{F}$ are parametrized using the techniques presented in [20]. Each face in these meshes is considered as a bilinear polynomial surface. The error of a face $\mathbf{f}$ in $\mathbf{M}_k$, $k \geq 0$, is defined

by

$$\xi_{\mathbf{f}} = max_{u \in \mathbf{f}} \left\| \mathbf{M}_k(u) - \mathbf{F}(u) \right\|_2 . \tag{1}$$

The initial *label* of a face $\mathbf{f}$ in $\mathbf{M}_0$, denoted $L_f(\mathbf{f})$, is set to k if all the faces descended from $\mathbf{f}$ after k Catmull-Clark subdivision steps satisfy the given error tolerance ϵ, i.e. errors of these faces are smaller than ϵ. Consequently, *labels* of the faces in $\mathbf{M}_l$, $0 < l < k$, who are descendants of $\mathbf{f}$ would be $k - l$. The *label* of a vertex $\mathbf{v}$ in $\mathbf{M}_0$ is defined by

$$L_v(\mathbf{v}) = max \left\{ L_f(\mathbf{f}) \,|\mathbf{f} \in \mathbf{M}_0 \text{ and } \mathbf{v} \text{ is a vertex of } \mathbf{f} \right\}. \tag{2}$$

The label of a face $\mathbf{f}$ can be determined as follows. If $\mathbf{f}$ is the center piece of a 4×4 rectangular grid in $\mathbf{M}_0$, the label of $\mathbf{f}$ can be computed using the technique presented in [5] directly. Otherwise, keep performing Catmull-Clark subdivision on $\mathbf{M}_0$ until each descendent face of $\mathbf{f}$ is the center piece of a 4×4 rectangular grid and a label is computed using [5] for that descendent face. If the descendent face is generated by applying the Catmull-Clark subdivision i times, the label of the descendent face is added to i to get an adjusted label for the descendent face. The label of $\mathbf{f}$ is then the maximum of the adjusted labels of all the descendent faces. If a vertex of the face is an *extraordinary point* of the limit surface, one needs to estimate the label of a small neighborhood (face) of this point using convex hall property and the chordal deviation computation technique.

The adaptive refinement procedure requires the vertex labels of $\mathbf{M}_0$ to satisfy the *consistent condition*. A face of $\mathbf{M}_0$ is said to be an *illegal face* if it satisfies the following conditions:

1. two vertices of the face have non-zero labels, and

2. these vertices are on the same edge.

The vertex labels of $\mathbf{M}_0$ are said to satisfy the *consistent condition* if $\mathbf{M}_0$ contains no illegal faces. The consistent condition ensures that the adaptively refined meshes are crack free [4]. Usually $L_v(\mathbf{v})$ does not satisfy the consistent condition. The easiest way to make the vertex labels to satisfy the consistent condition is to set all the zero labels to 1. But this would unnecessarily increase the number of faces generated in the resulting meshes since the number of faces in the refined meshes is determined by the labels of the vertices. A better way is to construct an extension function $E_v(\mathbf{v})$ of $L_v(\mathbf{v})$,

$$E_v(\mathbf{v}) = \begin{cases} L_v(\mathbf{v}), & \text{if } L_v(\mathbf{v}) > 0; \\ 0 \text{ or } 1, & \text{if } L_v(\mathbf{v}) = 0, \end{cases} \tag{3}$$

which satisfies the consistent condition but with as many zero labels as possible.

A greedy algorithm for the construction of $E_v(\mathbf{v})$ via a *connection supporting graph* $\mathbf{G_b}$ is presented here. The vertices of $\mathbf{G_b}$ are those of the illegal faces whose labels are zero. The edges of $\mathbf{G_b}$ are those of $\mathbf{M}_0$ that connect vertices of $\mathbf{G_b}$. The extension function $E_v(\mathbf{v})$ is constructed by repeatedly selecting a vertex from $\mathbf{G_b}$, changing its label to 1 and then updating $\mathbf{G_b}$ accordingly. This process continues until $\mathbf{G_b}$ is empty. The complexity of this process is

that changing the label of a vertex from 0 to 1 would change the status of adjacent faces: some illegal faces would become legal and some legal faces would become illegal. Therefore, one needs to remove some old vertices and edges from $\mathbf{G_b}$ as well as to introduce some new vertices and edges into $\mathbf{G_b}$. Obviously, the greedy algorithm should remove as many old vertices from $\mathbf{G_b}$ and introduce as few new vertices into $\mathbf{G_b}$ as possible during each cycle. This is achieved by using the following rule in selecting a vertex from $\mathbf{G_b}$ to change label. Let $D(\mathbf{v})$ denote the degree of $\mathbf{v}$ in $\mathbf{G_b}$ and let $N(\mathbf{v})$ be the number of new vertices introduced into $\mathbf{G_b}$ if the label of $\mathbf{v}$ is changed from 0 to 1. If the number of $D(\mathbf{v}) = 1$ vertices is not zero then, in the pool of vertices which are adjacent to a $D(\mathbf{v}) = 1$ vertex, select any one with a minimum $N(\mathbf{v})$ among those with maximum $D(\mathbf{v})$. Otherwise, select any vertex with a minimum $N(\mathbf{v})$ among the vertices of $\mathbf{G_b}$ with maximum $D(\mathbf{v})$.

The adaptive subdivision process is driven by vertex labels and is performed on individual mesh faces independently. After each subdivision step, labels should be assigned to the newly generated vertices so they can drive the next subdivision step. The resulting meshes are guaranteed to be crack free. We shall assume that labels of the vertices of $\mathbf{M}_0$ are defined by an extension function $E_{\mathbf{v}}$. In the following, $\mathbf{M}_k$, $k = 1, 2, \cdots$, stand for the meshes generated by the adaptive refinement process. Also, variables without a bar refer to those of $\mathbf{M}_{k-1}$, and variables with a bar refer to those of $\mathbf{M}_k$.

The adaptive subdivision of $\mathbf{M}_{k-1}$, $k \geq 1$, is performed as follows. If a face has two or more nonzero vertex labels, a *balanced Catmull-Clark subdivision* is performed on that face (see Figure 1). A balanced Catmull-Clark subdivision is a standard Catmull-Clark subdivision. However, coordinates of the new vertices will not be computed yet. The new vertices will be marked "UPDATE" though. Labels of the new vertices are defined as follows. For each new vertex point, $\overline{E}_{\mathbf{v}}(\overline{\mathbf{v}}_i) = max\left\{0, E_v(\mathbf{v}_i) - 1\right\}, i = 1, 2, 3, 4$. For each new edge point, $\overline{E}_{\mathbf{v}}(\overline{\mathbf{v}}_i)$ is the minimum of labels of the new vertex points adjacent to $\overline{\mathbf{v}}_i$, $i = 5, 6, 7, 8$. For the face point,

$$\overline{E}_v(\overline{\mathbf{v}}_9) = \begin{cases} 0, & \text{if } \overline{E}_v(\overline{\mathbf{v}}_5) = \overline{E}_v(\overline{\mathbf{v}}_6) = \overline{E}_v(\overline{\mathbf{v}}_7) = \overline{E}_v(\overline{\mathbf{v}}_8) = 0; \\ 1, & \text{if some but not all of } \{\ \overline{E}_v(\overline{\mathbf{v}}_5), \overline{E}_v(\overline{\mathbf{v}}_6), \overline{E}_v(\overline{\mathbf{v}}_7), \overline{E}_v(\overline{\mathbf{v}}_8)\ \} \text{ are zero;} \\ min\left\{\ \overline{E}_{\mathbf{v}}(\overline{\mathbf{v}})\ \mid \overline{\mathbf{v}} \in \{\overline{\mathbf{v}}_5, \overline{\mathbf{v}}_6, \overline{\mathbf{v}}_7, \overline{\mathbf{v}}_8, \}\ \right\}, & \text{otherwise.} \end{cases}$$

If a face has only one nonzero vertex label, an *unbalanced Catmull-Clark subdivision* with respect to the vertex with nonzero label is performed (see Figure 2). An unbalanced Catmull-Clark subdivision generates three new faces only, as shown in Figure 2(c). But $\overline{\mathbf{v}}_8$, $\overline{\mathbf{v}}_9$ and the auxiliary structure shown in Figure 2(b) will still be computed and recorded; they are needed in the computation of the vertices of $\mathbf{M}_{k+1}$. Again, coordinates of the new vertices are not computed at this moment. These vertices, except $\overline{\mathbf{v}}_3$, are marked with an "UPDATE" to indicate that they will be evaluated later. The labels of all the new points are set to zero except $\overline{E}_v(\overline{\mathbf{v}}_1)$ which is defined as $E_v(\mathbf{v}_1) - 1$. Those faces without non-zero vertex labels will not be adaptively subdivided, but will be inherited topologically.

After all the faces of $\mathbf{M}_{k-1}$ are processed, vertices marked with an "UPDATE" in $\mathbf{M}_k$ are computed using the Catmull-Clark subdivision scheme to find their new coordinates in $\mathbf{M}_k$. Note that the vertices of $\mathbf{M}_{k-1}$ required in the computation process for the new vertices are available because they are stored with the auxiliary structure (see Figure 2(b)) even though

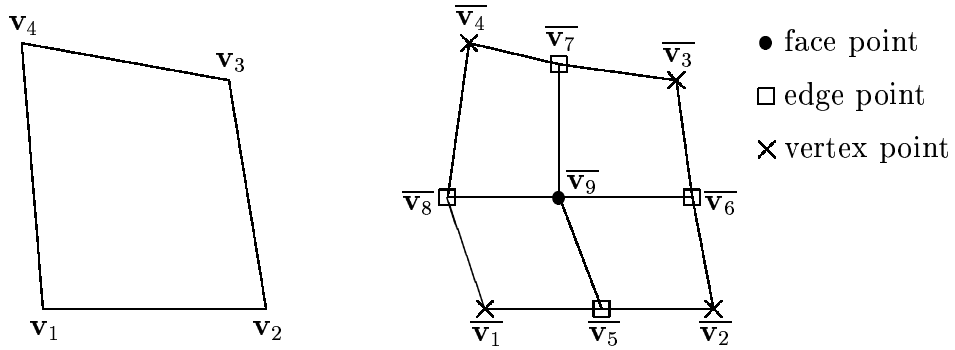


Figure 1: Balanced Catmull-Clark subdivision **(a)** before; **(b)** after.

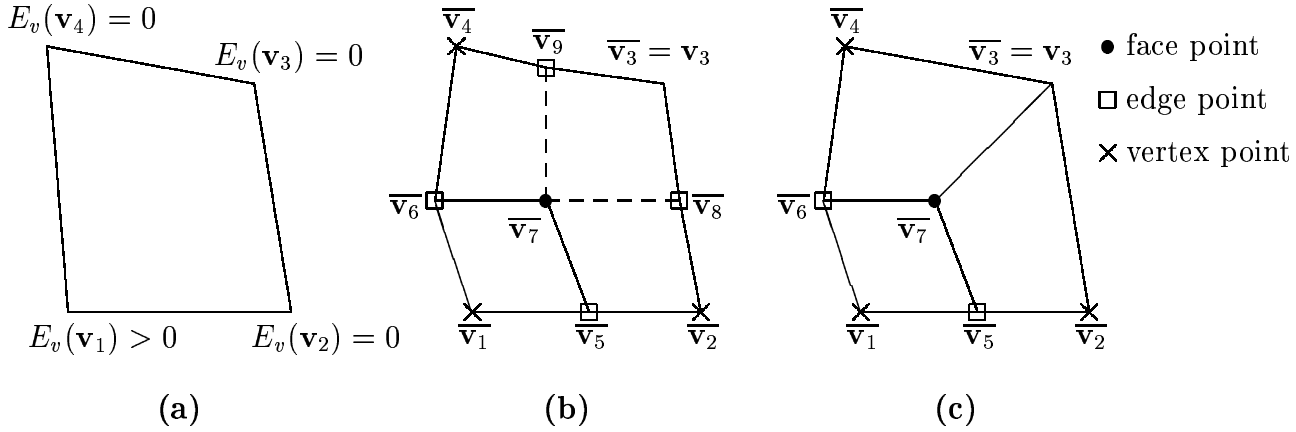


Figure 2: Unbalanced Catmull-Clark subdivision with respect to $\mathbf{v}_1$ **(a)** before subdivision; **(b)** auxiliary structure stored after subdivision; **(c)** structure output after subdivision

not output. Other vertices (vertices not marked with an "UPDATE") of $\mathbf{M}_k$ will be inherited from $\mathbf{M}_{k-1}$ directly. Setting up the status of a vertex with the mark "UPDATE" is necessary because whether a vertex should be inherited or updated depends on all its adjacent faces.

3 Error-Driven Adaptive Subdivision

The idea here is to avoid as many unnecessary subdivision steps as possible by examining the error between the current mesh faces and the limit surface more thoroughly. A face $\mathbf{f} \in \mathbf{M}_{k-1}$ is subdivided only if it does not satisfy the error condition or to avoid crack. The procedure is as follows. The notations used here follow those of Section 2.

Step 1: $k = 1$; /* k is the subdivision step */
Step 2: for each face $\mathbf{f} \in \mathbf{M}_{k-1}$, define:
$L_f(\mathbf{f}) = 1$ if $\mathbf{f}$ does not satisfy the error condition;
otherwise, $L_f(\mathbf{f}) = 0$.
Step 3: if no face satisfies the condition $L_f(\mathbf{f}) = 1$, output $\mathbf{M}_{k-1}$, and exit this procedure;
Step 4: calculate $L_v(\mathbf{v})$, following Equation (2);
Step 5: calculate $E_v(\mathbf{v})$;
Step 6: adaptively subdivide $\mathbf{M}_{k-1}$, based on $E_v(\mathbf{v})$, to get $\mathbf{M}_k$;
Step 7: k=k+1, and go to Step 2; /* start a new subdivision step */

The error computation process of Step 2 is performed as follows. If $\mathbf{f} \in \mathbf{M}_{k-1}$ is a face descended from a face $\mathbf{g} \in \mathbf{M}_{k-2}$ and $L_f(\mathbf{g}) = 0$ in $\mathbf{M}_{k-2}$, then $L_f(\mathbf{f})$ is automatically set to zero. This is based on the assumption that new faces generated by a subdivision step would be closer to the limit surface than the parent face. For other faces, the error is computed using Equation (1).

The construction of the extension function $E_v(\mathbf{v})$ in Step 5, when $k = 1$, is the same as the greedy algorithm in the previous section. However, when $k > 1$, the construction of $E_v(\mathbf{v})$ is different from the greedy algorithm on the selection of a vertex to change label from 0 to 1. The vertex selection part of the construction process in this case depends on a vertex group called *forbidden vertex set* $\mathbf{V_b}$ as well. $\mathbf{V_b}$ consists of all the vertices that are not marked "UPDATE" in the $(k-1)st$ subdivision step. For an unbalanced subdivision in step $k-1$, as shown in **Figure 2**, $\overline{\mathbf{v}_i} \in \mathbf{V_b}$, for $i = 2, 3, \cdots, 7$. If $\mathbf{v} \in \mathbf{V_b}$ is a vertex point, and is marked "UPDATE" in subdivision step $k-1$, then all the edge points, which are adjacent to $\mathbf{v}$ and having zero vertex label in subdivision step k, belong to $\mathbf{V_b}$ as well. The rule for vertex selection when $k > 1$ is as follows. If $\mathbf{f}$ is an illegal face with $\mathbf{v}_1$ and $\mathbf{v}_2$ being its zero label vertices (i.e., $\mathbf{v}_1, \mathbf{v}_2 \in \mathbf{G_b}$) and $\mathbf{v}_2 \in \mathbf{V_b}$, then $\mathbf{v}_1$ is selected to change its label from 0 to 1 (note that $\mathbf{v}_1$ and $\mathbf{v}_2$ can never be in $\mathbf{V_b}$ at the same time). Otherwise, the selection process is the same as the greedy algorithm in the previous section.

The adaptive subdivision process performed in Step 6 is the same as that in the previous section.

4 Implementation and Results

The above adaptive refinement techniques have been implemented on SUN Ultra 10 workstations using OpenGL as the graphics system. Figures 3-4 show some of the test results. In these figures (and Table 1 as well), "CC" stands for the Catmull-Clark scheme, "ACC1" (adaptive Catmull-Clark 1) stands for the label-driven approach, and "ACC2" (adaptive Catmull-Clark 2) stands for the error-driven approach, respectively. The tested cases shown in Figure 3 are a heart, a cup, a ball and a hollow pyramid. The tested cases shown in Figure 4 are the

Stanford bunny[1,3] and the Powerflip cow[2,3]. As one can see in these figures, the unbalanced subdivision provides a way to build a smooth, crack-free transition between faces with different subdivision levels, therefore, avoid the need of performing the same level of subdivision on all the faces.

Comparison on the numbers of faces generated by different approaches is given in Table 1(a~f), where ϵ is the given error tolerance, Ms is the maximum subdivision level of vertices in the resulting mesh, "S_X" is the number of faces generated in the resulting mesh by scheme "X", and "R_X" is the ratio of the number of faces in the resulting mesh produced by adaptive subdivision approach "X" to the number of faces in the mesh produced by the corresponding standard subdivision scheme. For instance, the second line in Table 1(a) should be interpreted as follows: if 0.7 is the given error tolerance, then 2 is the maximum subdivision level for all the vertices in the resulting mesh to satisfy the error requirement. In this case, the number of faces in the resulting mesh generated by the standard CC scheme is 698, the numbers of faces in the resulting meshes generated by ACC1 and ACC2 are 284 and 212, respectively, and the ratio of ACC1 to CC is 0.47 and the ratio of ACC2 to CC is 0.35. As can be seen from the table, both adaptive subdivision techniques reduce the numbers of faces in the resulting meshes significantly. However, the error-driven approach produces much fewer faces in the resulting meshes if ϵ is small. This is because smaller ϵ requires larger subdivision levels to reach the required precision. Therefore, by dynamically testing if a subdivision step is needed before each iteration, one can avoid more unnecessary subdivision steps in such a case and, consequently, further reduce the number of faces generated in the resulting mesh.

The presented techniques can be used for cubic Doo-Sabin subdivision surfaces (Figure 5(a~d)), non-uniform cubic subdivision surfaces (Figure 5(e~h)) and combined subdivision surfaces (not implemented in this paper) as well. In Figure 5, "DS" and "NU' stand for Doo-Sabin and non-uniform subdivision schemes, respectively. "ADS1" (adaptive Doo-Sabin 1) and "ADS2" (adaptive Doo-Sabin 2) stand for the label-driven approach and the error-driven approach, respectively. "ANU1" and "ANU2" are defined similarly. The input mesh is the control mesh for the hollow pyramid shown in Figure 3, where "0.3", "0.9" and "0.3" are knot spacings, and the other knot spacings, not shown here, are 1. The performance of the adaptive approaches for these two cases is shown in Table 1(g) and (h). The results are similar to those of the Catmull-Clark surfaces.

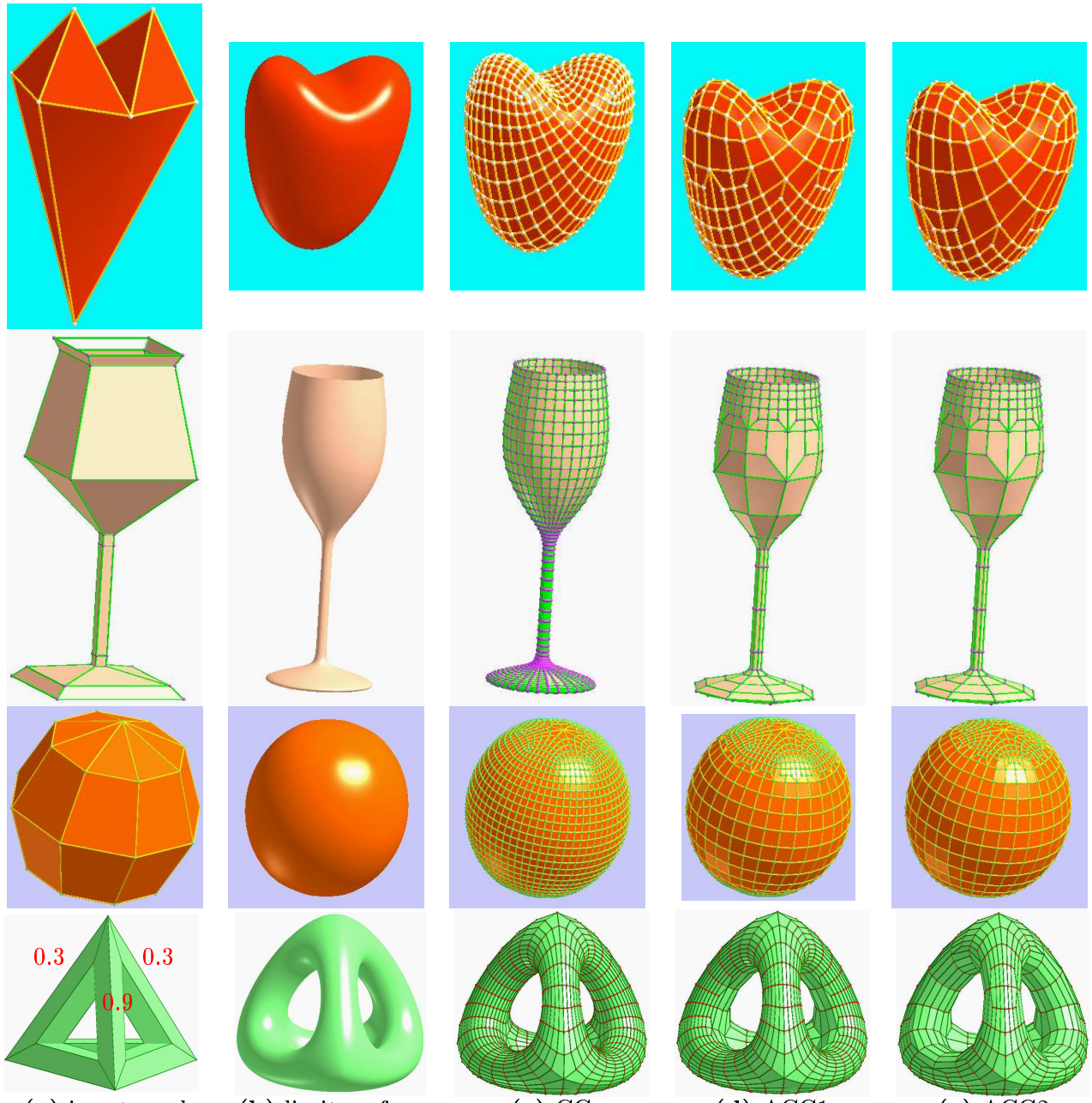
5 Conclusions

This paper presents two adaptive mesh refinement techniques for cubic subdivision surfaces. The techniques are presented for the cubic Catmull-Clark subdivision surfaces. But they can be used for cubic Doo-Sabin subdivision surfaces [7], non-uniform cubic subdivision surfaces [19], and combined subdivision surfaces [15] as well. These techniques avoid uniform subdivi-

[1]Downloaded from "http://graphics.cs.uiuc.edu/~garland/class/model/bunny.mmf".

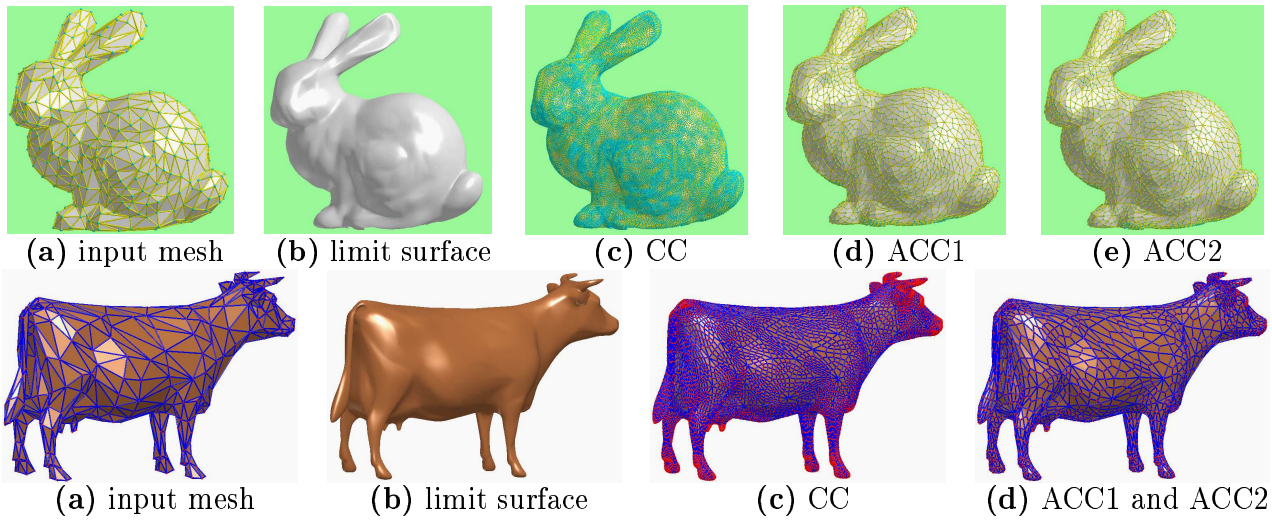
[2]Downloaded from "http://www.watson.org/~tesla/projects/decimate/doc/cow.smf".

[3]The data set is simplified using the program: "http://www.watson.org/~tesla/projects/decimate/doc/SimplifySurface.exe". The method used in this program is presented in [8, 9].



(a) input mesh **(b)** limit surface **(c)** CC **(d)** ACC1 **(e)** ACC2

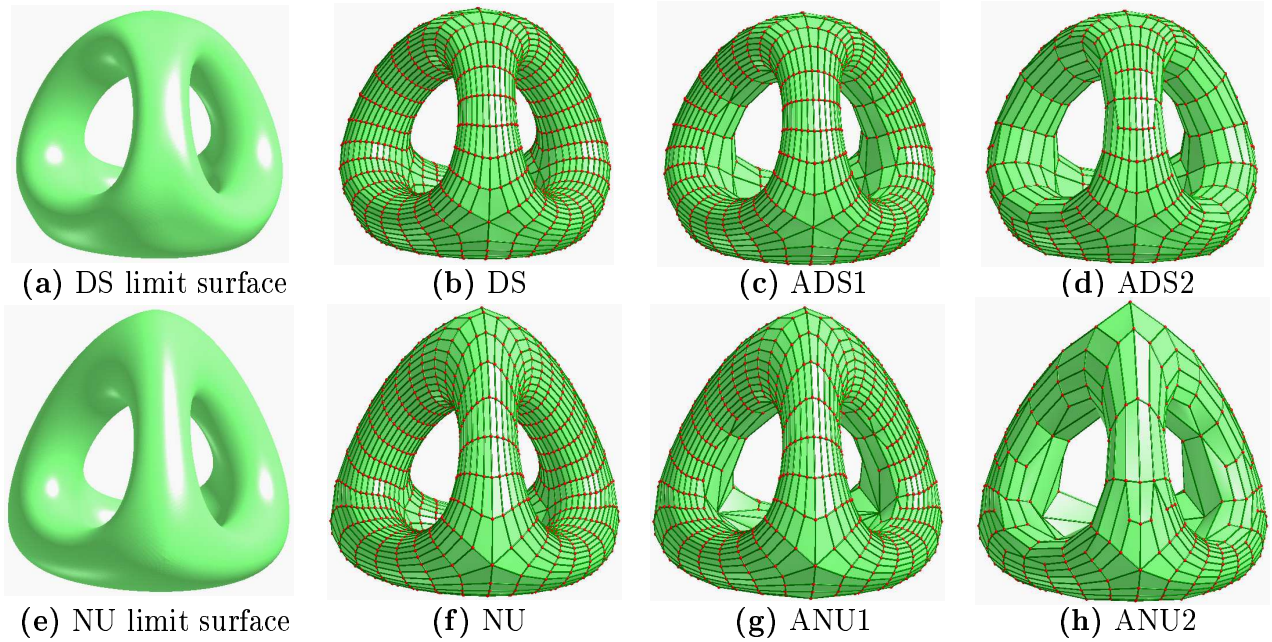
Figure 3: Heart ($\epsilon = 0.7$), cup ($\epsilon = 2$), ball ($\epsilon = 0.6$) and hollow pyramid ($\epsilon = 1$).



(a) input mesh **(b)** limit surface **(c)** CC **(d)** ACC1 **(e)** ACC2

(a) input mesh **(b)** limit surface **(c)** CC **(d)** ACC1 and ACC2

Figure 4: Stanford bunny ($\epsilon = 0.006$) and Powerflip cow ($\epsilon = 0.014$).



(a) DS limit surface **(b)** DS **(c)** ADS1 **(d)** ADS2

(e) NU limit surface **(f)** NU **(g)** ANU1 **(h)** ANU2

Figure 5: Cubic Doo-Sabin surface ($\epsilon = 0.8$) and non-uniform surface ($\epsilon = 3$).

Table 1: Comparison on numbers of faces generated by different approaches

ϵ	Ms	S_{CC}	S_{ACC1}	R_{ACC1}	S_{ACC2}	R_{ACC2}
0.7	2	608	284	0.47	212	0.35
0.068	4	9728	3514	0.36	2232	0.23
0.014	6	155648	39264	0.25	9700	0.062
0.0035	8	2490368	485356	0.19	38567	0.015

(a) heart

ϵ	Ms	S_{CC}	S_{ACC1}	R_{ACC1}	S_{ACC2}	R_{ACC2}
2	3	1888	268	0.14	268	0.14
1	4	7488	704	0.094	656	0.088
0.15	7	475648	4768	0.010	4680	0.0098
0.06	8	1901568	11264	0.0059	10256	0.0054

(b) cup

ϵ	Ms	S_{CC}	S_{ACC1}	R_{ACC1}	S_{ACC2}	R_{ACC2}
0.6	2	1792	960	0.54	688	0.38
0.3	3	7168	2368	0.33	928	0.14
0.1	5	114688	22400	0.20	2816	0.025
0.04	7	1835008	287488	0.16	8384	0.0046

(c) ball

ϵ	Ms	S_{CC}	S_{ACC1}	R_{ACC1}	S_{ACC2}	R_{ACC2}
1	3	1536	1200	0.78	696	0.45
0.5	4	6144	4320	0.70	1656	0.27
0.1	6	98304	63360	0.64	6273	0.064
0.025	8	1572864	990720	0.63	17676	0.011

(d) hollow pyramid

ϵ	Ms	S_{CC}	S_{ACC1}	R_{ACC1}	S_{ACC2}	R_{ACC2}
0.015	1	11863	3020	0.25	3020	0.25
0.006	2	47079	3553	0.075	3524	0.075
0.0027	3	187447	4735	0.025	4673	0.025
0.0015	5	2987863	8871	0.0030	8195	0.0027

(e) Stanford bunny

ϵ	Ms	S_{CC}	S_{ACC1}	R_{ACC1}	S_{ACC2}	R_{ACC2}
0.014	1	12000	3072	0.26	3072	0.26
0.004	3	192000	19380	0.10	16250	0.085
0.0024	4	768000	51865	0.068	31170	0.041
0.0013	5	3072000	197125	0.064	101062	0.033

(f) Powerflip cow

ϵ	Ms	S_{DS}	S_{ADS1}	R_{ADS1}	S_{ADS2}	R_{ADS2}
0.8	3	1536	1200	0.78	696	0.45
0.3	4	6144	4320	0.70	1896	0.31
0.08	6	98304	63360	0.64	6576	0.067
0.025	8	1572864	990720	0.63	18777	0.012

(g) Cubic Doo-Sabin surface

ϵ	Ms	S_{NU}	S_{ANU1}	R_{ANU1}	S_{ANU2}	R_{ANU2}
3	3	1536	1032	0.67	466	0.30
1	5	24576	12000	0.49	2415	0.098
0.8	6	98304	42210	0.43	3328	0.034
0.4	8	1572864	432568	0.28	10008	0.0064

(h) Non-uniform cubic subdivision surface

sion of all the faces by using labels of the mesh vertices or error between the mesh faces and the limit surface to drive the refinement process.

The strengths of the new techniques include: (1) significantly reduce the number of faces in the refined meshes in just a few (3-5) subdivision steps, (2) conformity (crack-free condition) of the resulting mesh is automatically guaranteed, (3) the refined meshes converge to the same limit surface, and (4) the input mesh could have arbitrary topology.

The limitation of this work is that the proposed adaptive subdivision techniques currently work on subdivision surfaces of the Catmull-Clark style only. A future work is to extend the new techniques to cover other subdivision surfaces such as quadratic Doo-Sabin subdivision surfaces. For such an extension, other shapes, rather than only quadrilaterals, should also be allowed as faces in the resulting meshes.

References

[1] Alliez, P. and Desbrun, M. Progressive Compression for Lossless Transmission of Triangle Meshes. In *Proceedings of SIGGRAPH 2001*, 195-202.

[2] Biermann, H., Kristjansson, D., and Zorin, D. 2001. Approximate Boolean Operations on Free-Form Solids. In *Proceedings of SIGGRAPH 2001*, 185-194.

[3] Catmull, E., and Clark, J. 1978. Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes. *Computer-Aided Design 10*, 6, 350-355.

[4] Cheng, F., Jaromczyk, J.W., Lin, J.-R., Chang, S.-S., and Lu J.-Y. 1989. A Parallel Mesh Generation Algorithm Based on the Vertex Label Assignment Scheme. *International Journal for Numerical Methods in Engineering 28*, 1429-1448.

[5] Cheng, F. 1992. Estimating Subdivision Depths for Rational Curves and Surfaces. *ACM Trans. on Graphics 11*, 2 140-151.

[6] DeRose, T., Kass, M., Truong, T. 1998. Subdivision Surfaces in Character Animation. In *Proceedings of SIGGRAPH 1998*, 85-94.

[7] Doo, D., and Sabin, M. 1978. Behavior of Recursive Division Surfaces near Extraordinary Points. *Computer-Aided Design 10*, 6, 356-360.

[8] Garland, M., and Heckbert, P. 1997. Surface Simplification Using Quadric Error Metrics. In *Proceedings of SIGGRAPH 1997*, 209-216.

[9] Garland, M. 1999. *Quadric-Based Polygonal Surface Simplification*. PhD thesis, Carnegie Mellon University.

[10] Hoppe, H. 1996. Progressive Meshes . In *Proceedings of SIGGRAPH 1996*, 99-108.

[11] Khodakovsky, A., Schröder, P., and Sweldens, W. 2000. Progressive Geometry Compression. In *Proceedings of SIGGRAPH 2000*, 271-278.

[12] Kobbelt, L. 1996. Interpolatory Subdivision on Open Quadrilateral Nets with Arbitrary Topology. *Computer Graphics Forum 15*, 3, 409-420.

[13] Kobbelt, L. 2000. $\sqrt{3}$ Subdivision. In *Proceedings of SIGGRAPH 2000*, 103-112.

[14] Lee, A., Moreton, H., and Hoppe, H. Displaced Subdivision Surfaces. In *Proceedings of SIGGRAPH 2000*, 85-94.

[15] Levin, A. 1999. Interpolating Nets of Curves by Smooth Subdivision Surfaces. In *Proceedings of SIGGRAPH 1999*, 57-64.

[16] Lindstrom, P. 2000. Out-of-Core Simplification of Large Polygonal Models. In *Proceedings of SIGGRAPH 2000*, 259-262.

[17] Litke, N., Levin, A., and Schröder, P. 2001. Trimming for Subdivision Surfaces. *Computer Aided Geometric Design 18*, 5, 463-481.

[18] Peters, J. Patching Catmull-Clark Meshes. In *Proceedings of SIGGRAPH 2000*, 255-258.

[19] Sederberg, T.W. 1998. Non-Uniform Recursive Subdivision Surfaces. In *Proceedings of SIGGRAPH 1999*, 387-394.

[20] Stam, J. 1998. Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values. In *Proceedings of SIGGRAPH 1998*, 395-404.

[21] Zorin, D., Schröder, P., and Sweldens, W. Interactive Multiresolution Mesh Editing. In *Proceedings of SIGGRAPH 1997*, 259-268.