Journal for Geometry and Graphics
Volume 8 (2004), No. 1, 69–80.

An Adaptive Scheme for Subdivision Surfaces based on Triangular Meshes

Weizhong Liu, Kunio Kondo

Department of Information and Computer Sciences
Saitama University, 338-8570, Japan
email: {wzliu,kondo}@ke.ics.saitama-u.ac.jp

Abstract. One problem in subdivision surfaces is that the number of meshes grows quickly after every subdivision step. The number of meshes of the subdivision surface is usually huge and the scheme is difficult to manipulate. Subdivision schemes are cost intensive at higher levels of subdivision. In this paper, we introduce an adaptive subdivision scheme for subdivision surfaces based on triangular meshes. This scheme works with the new subdivision rules and the biggest angle between the normal vectors of adjacent faces of a vertex is considered as error estimation and termed *CA*. The regular subdivision process is modified to stop at the flat areas, so we can represent surfaces with lower cost when compared with those obtained by regular subdivision schemes. In our scheme, we take care of the T-junction (cracking) problem and propose our solution. We compare our methods for various triangular meshes and present our results.

Key Words: subdivision surfaces, adaptive scheme, triangular mesh
MSC 2000: 68U05

1. Introduction

As polyhedral subdivision process provides a simple and efficient way to generate surfaces over polyhedral networks, it has become one of the basic tools in Computer Aided Geometric Design (CAGD) for modelling complex surfaces. Many approaches were made since two basic subdivision surfaces were proposed by Catmull-Clark [2] and Doo-Sabin [3] in 1978. Nasri [13] extended Doo-Sabin surfaces to interpolate vertices of an original polyhedron and B-spline curves on the subdivision surface. N. Dyn et al. [4] proposed a butterfly subdivision scheme, it is an interpolating scheme and extended by Zorin et al. [19]. Schemes based on triangles were discussed by Farin [5] and Loop [10]. M. Halstead et al. [6] proposed an interpolation method with a Catmull-Clark surface. And a Non-Uniform Recursive Subdivision Surface (NURSS) was proposed by Sederberg et al. [15] recently.

Generally, in subdivision surfaces, the subdivision process is over a simple polyhedron, the whole polyhedral meshes are refined globally at a level of mesh density. After several subdivision steps, the generated subdivision surface will be smooth enough to represent a fine shape. But even after a few subdivision steps, the number of generated meshes will become huge. For example, at Loop surfaces, the number of meshes after one refinement step is about four times that of original meshes. It is difficult to deal with these data. But usually after several steps of iterations, most areas of subdivision surfaces are smooth enough to give fine schemes, only some regions where curvatures change significantly are still coarse, and need to be refined. It therefore is not ideal to have a global subdivision scheme being applied at every level. Adaptive Subdivision aims at providing a local subdivision rule that governs whether or not a given face in a mesh needs to be subdivided at the next level of subdivision.

2. Related works

We will overview research works related to adaptive subdivision schemes and refinements.

Mueller [12] proposed an adaptive process for Catmull-Clark and Doo-Sabin subdivision schemes. In his method, adaptation is controlled by an error measure which indicates for the vertices of a mesh whether the approximation is sufficient. The error estimation is measured as the distance between a original vertex of the mesh and its limit point. All the vertices that lie in the error range are labelled differently and special rules are applied for subdividing a polygon when it contains one or more of these labelled vertices.

Xu and Kondo [18] devised an adaptive subdivision scheme based on the Doo-Sabin scheme. In their method the adaptive refinement is controlled by the faces of the original mesh. Faces are labelled as *alive* or *dead* if they have to be subdivided or not. The labelling is based on the angle between the normal vectors of adjacent faces and a tolerance limit for this angle is set. If a face satisfies the set tolerance then it is labelled as dead and further refinements are stopped for that face.

Kobbelt proposed an adaptive refinement method for both his Kobbelt scheme and newly introduced $\sqrt{3}$ subdivision [9]. His refinement strategy is also centered around the faces. In both the schemes adaptive refinement presents a face cracking problem. His solution is to use a combination of mesh balancing and the Y-technique for his Kobbelt scheme. For his $\sqrt{3}$ subdivision he uses a combination of dyadic refinement, mesh balancing and gap fixing by temporary triangle fans. This process is well known in the finite element community under the name red-green triangulation [17].

Amresh et al. [1] proposed an adaptive subdivision scheme for the Loop scheme. Their first method uses the angles between normals of a face with adjoining face normals to determine if the face needs to be subdivided or not. Their second method is based on user interaction, where the user can select areas on the mesh where refinements are desired. They perform an identification process called watershed segmentation to identify the regions in a mesh that need to be subdivided. But they also note that when the mesh is highly undulating, then using a curvature based user selected process like watershed segmentation will not result in considerable savings in mesh size. In their schemes, they mention the cracking problem and introduce a method called triangle fans to solve it.

Zorin et al. have developed adaptive refinement strategies in [20], where they have additional constraints that require a certain number of vertices in the neighborhood of those vertices calculated by adaptive subdivision to be present. Their methods have been implemented on the Loop scheme.

Generally, the adaptive strategies can be developed in two ways, one by classifying which vertices need to be subdivided (vertex split operation at the next level) or two by identifying those faces that should be subdivided (face split at the next level).

3. The Loop subdivision scheme

The Loop scheme is a simple approximating face-split scheme for triangular meshes proposed by Charles Loop [10]. The scheme is based on the triangular splines [16], which produces C^2-continuous surfaces over regular meshes. A regular mesh is a mesh which has no extraordinary vertices, i.e., vertices whose number of neighbors do not equal six. It also means that the vertex has a valance of six. The Loop scheme produces surfaces that are C^2-continuous everywhere except at extraordinary vertices, where they are C^1-continuous. A boundary vertex is regular or even if it has a valance of three and is extraordinary for any other valance. The masks for the Loop scheme are shown in Fig. 1. For boundaries special rules are used. These rules produce a cubic spline curve along the boundary. The curve only depends on control points on the boundary. The scheme works as follows:

- For every original vertex a new vertex (odd vertex) is calculated by calculating β from eq. (1), where n is the number of adjacent vertices for the vertex, and finding the suitable coefficients for the adjacent control points as shown in Fig. 1.
- For every edge in the original mesh a new vertex (even vertex) is calculated by using the mask shown in Fig. 1.
- Every triangle in the original mesh gives rise to six new vertices, three from original vertices and three from original edges, these six vertices are joined to give four new triangles.

In Fig. 1 n is the number of adjacent vertices for a given vertex and β can be chosen as

$$\beta = \frac{1}{n}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n}\right)^2\right) \tag{1}$$

The value for β [8] was found such that the resulting surface is C^1-continuous at the extraordinary points. For regular vertices the coefficients for calculating the new vertices are obtained by substituting n as six in the mask for even vertices shown in Fig. 1.

4. Our proposed method

We now discuss the method we have developed for adaptively subdividing meshes. Our method is based on identifying which vertex of a face is *"dead"* (see Section 4.3) and proposing suitable mesh refinements based on the properties of its three vertices.

We firstly analyze the planar areas created in the subdivision surface, and then we give some definitions and present our new subdivision rules. We take care of the T-junction (cracking) problem in our scheme and propose our solution which is called *LMR*. Finally, we analyze the results.

4.1. Analysis of the planar areas created in subdivision surfaces

Fig. 2 shows the planar area in subdivision surfaces. For a face $f_0 = \{v_0, v_1, v_2\}$, the limit surface is the shaded area in Fig. 2(b). c_0, c_1, c_2 are the limit points of the vertices v_0, v_1, v_2, respectively. The boundary curves of the limit surface are b_0, b_1, b_2. Now assume the neighbor

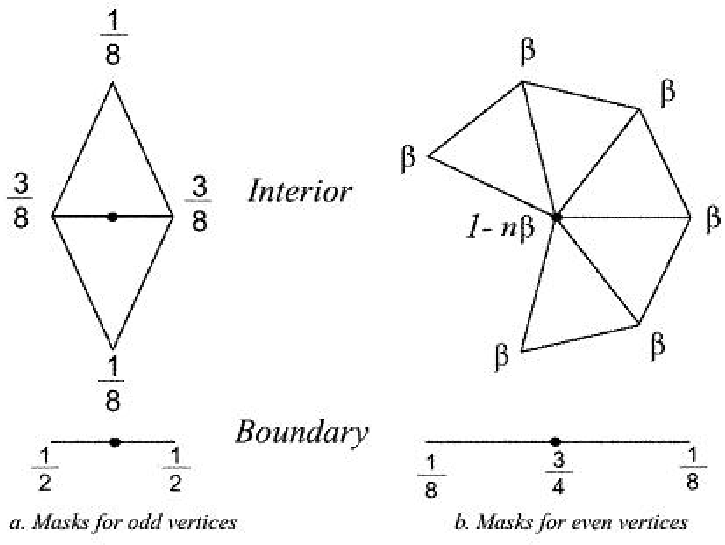


Figure 1: Masks for LOOP scheme

faces of vertices v_0, v_1, v_2 are on a plane, so the limit subdivision surface is also on that plane, just like shown in Fig. 2(b). For each face of initial polyhedral meshes, there is a limit surface that is decided by the faces meeting at its three vertices. If these faces are on a plane, then the limit face is also on that plane. Actually, the limit surface is an n-sided patch. What we should do is to get the boundary curves of the planar limit face. Usually, in subdivision surfaces, the boundary curves can be gotten by subdividing the faces that adjoin the planar limit face, the unnecessary subdivision process in the planar limit face area can be avoided and the continuity property of subdivision surface can also be kept.

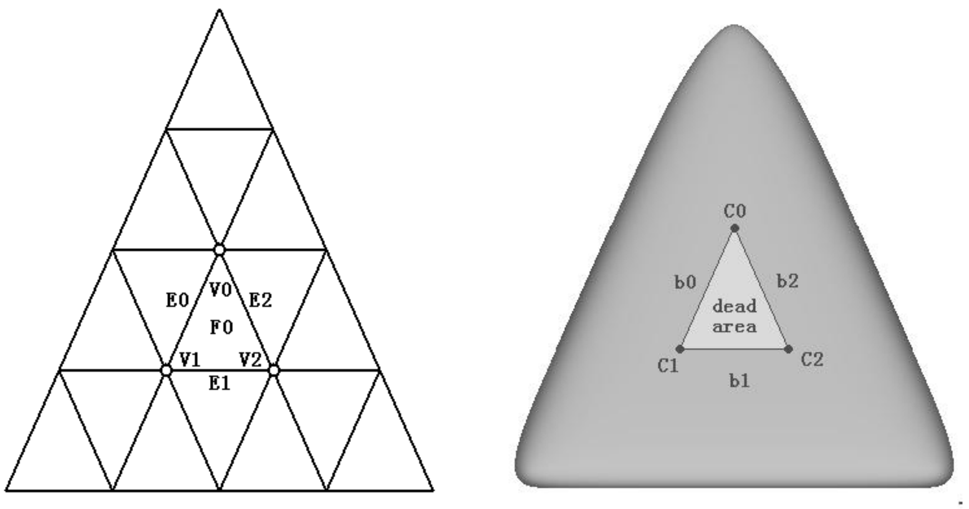


Figure 2: The planar area in LOOP surfaces

4.2. Definition of the conical angle

The allowable tolerance is used here to decide whether the surfaces meeting at a vertex give a sufficient approximation plane. In this paper, the biggest angle between the normal vectors of each pair of adjacent faces that meet at a common vertex v_i is used as allowable tolerance, and called *conical angle* (*CA*), referring to Fig. 3. The user can select a suitable *CA* to control the smoothness of the surfaces of the final shapes.

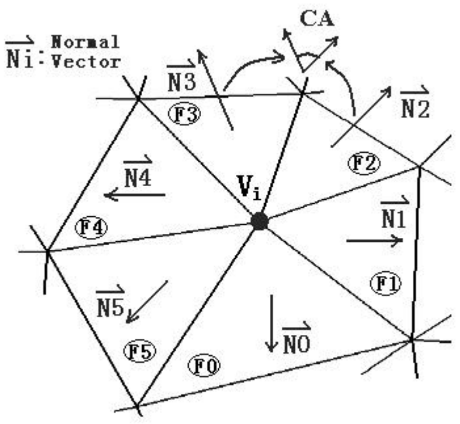


Figure 3: Definition of the *conical angle (CA)*

4.3. Definition of *dead* vertices, edges and faces

Firstly, we introduce some items for describing our process easily.

1. For vertices: A vertex is called *flat*, if the corresponding *CA* is within some tolerance limit. It is called *dead*, if its neighbor vertices are *flat*. In all other cases the vertex is called *alive*.
2. For edges: An edge is called *dead*, if each of its vertices is *dead*. In all other cases the edge is called *alive*.
3. For faces: A face is called *dead*, if each of its vertices is *dead*. In all other cases the face is called *alive*. Of course, the *dead* face will not be subdivided in the further subdivision process.

4.4. Conical angle (*CA*) method

Our method is called *"conical angle method"* because it uses the *CA* of every vertex of a face to determine if the face needs to be subdivided or not.

From the definition of *CA* it can be seen that the angle of the normal vectors of adjacent faces of a vertex is considered as error estimation. If the angles are within some tolerance limit then we classify the vertex as *flat*. If all the neighbor vertices of a vertex are classified as *flat*, we classify the vertex as *dead*. The new mesh refinements are based on the degree of deadness of a face and are shown in Fig. 4. In the case $n = 2$ in Fig. 4, we select a pair of vertices with lower error by comparing the angle between (v_1, v_5, v_4) and (v_5, v_4, v_2). We introduce an adaptive weight θ to control the tolerance limit. Our scheme works as follows:

- The normal for each face is calculated.
- For every vertex, its conical angle is calculated.
- If the conical angle lies below a certain threshold then the vertex is set to be *flat*.
- For every vertex, if all its neighboring vertices are *flat*, then it is set to be *dead*.

- For every face, a *degree of deadness* which equals the number of its *dead* vertices is defined. The maximum value for this degree can be three and the minimum will be zero. Based on the degree of deadness, refinement is done as shown in Fig. 4.

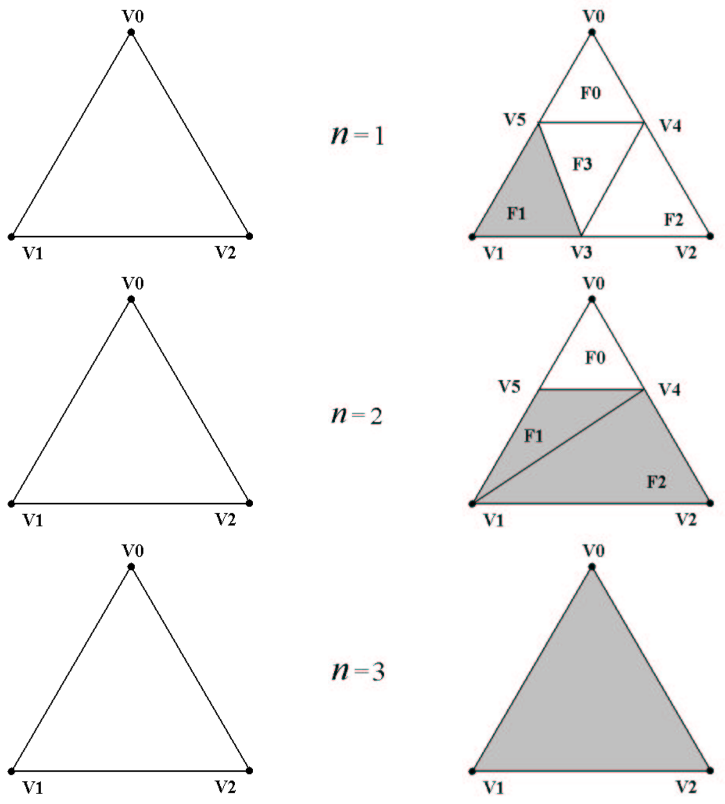


Figure 4: New subdivision rules: The mesh refinement is based on the degree of deadness. n is the number of *dead* vertices of a face

4.5. T-junction (Cracking) problem and its solution

Subdividing a non-flat face that is adjacent to a flat face produces a *T-junction* or *cracking problem*. These T-junctions can generate cracks when we use a non-zero tolerance. We take care of this problem and give our solution which is called *LMR (local mesh realignment)* refinement method.

LMR is an effective refinement to employ during the adaptive subdivision. A brief explanation of *LMR* algorithm is as follows:

- For every level subdivision, mark the newly generated *dead* edges whose two neighbor faces are *alive*.
- For every marked *dead* edge, a local mesh realignment (*LMR*) for this edge is implemented based on the following Rule 1.
- For every level subdivision, mark the newly generated vertices whose *CA* (conical angle) is set to *dead*.
- For every marked *dead* vertex, a local refinement process for this vertex is implemented based on the following Rule 2.

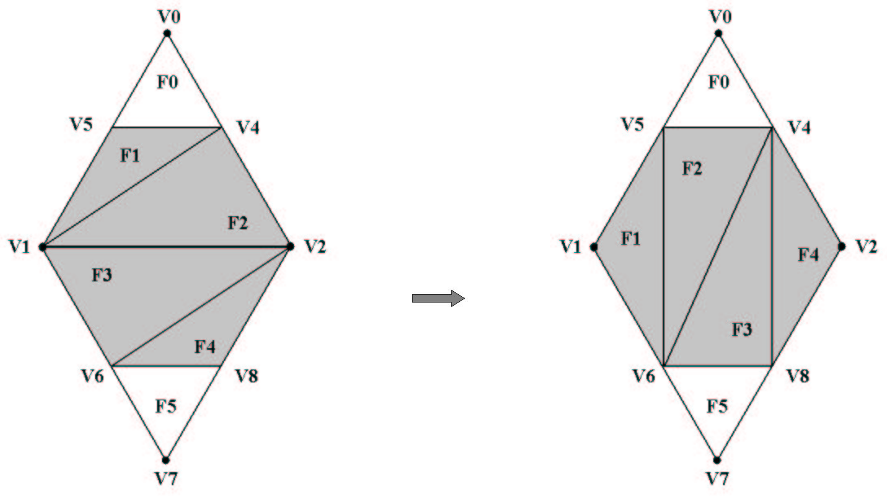
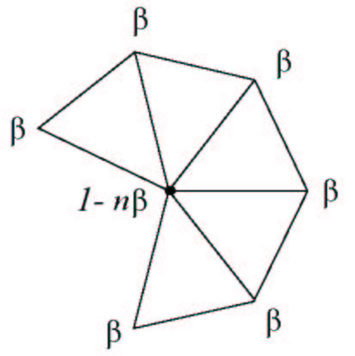


Figure 5: *LMR* Rule 1: Local realignment rule for *dead* edges. The shaded areas are *dead* areas



$$\beta = \frac{1}{n}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n}\right)^2\right) \qquad (1)$$

Figure 6: *LMR* Rule 2: Local refinement rule for *dead* vertices

4.6. Adaptive subdivision refinement process

The procedures of our adaptive subdivision process are described as follows:

1. Giving a value of *CA* between 0° and 180°:
   According to requirements, we choose a suitable value of *CA* (conical angle) which describes the smoothness of the surfaces of final shape. Basically, the value of *CA* must be zero for keeping the continuity property of subdivision surface. But according to requirements, user can select a suitable *CA* to control the smoothness of the surfaces of final shapes.
2. Modification of the subdivision process:
   Assuming that the k-th adaptive process has been implemented, for the coming (k+1)-th step the modified subdivision process can be described as follows:
   (a) Step 1: Implementing a process called *NLFC (next level flags calculation)* process for the mesh created in the k-th level subdivision. This process is to calculate the *dead* information of each *alive* vertex, edge and face and mark them.

(b) Step 2: Generating new vertices.
For *dead* edges, the new vertices will not be generated, for each *alive* edge, a new vertex is generated with the original subdivision method. If one vertex of the *alive* edge is a *dead* vertex, the new vertex is also a *dead* vertex; if two vertices are *flat* vertices, the new vertex is also a *flat* vertex, in other case, it is an *alive* vertex. We mark these information for the new created vertices.

(c) Step 3: Generating new faces.
For *dead* faces, no new faces will be generated, for an *alive* face $f_i = \{v_0, v_1, v_2\}$, there are three cases for f_i to generate new faces based on the degree of *deadness.*

   i. Case 1: One vertex is a *dead* vertex while the other two vertices are *flat.* In this case, a dead face will be generated according to the regular subdivision method. Referring to Fig. 4(a), f_1 is a *dead* face.
   ii. Case 2: Two vertices are *dead* vertices while the other one is *flat.* In this case, three faces will be generated while two faces of them are *dead* faces. Referring to Fig. 4(b), f_1 and f_2 are *dead* faces.
   iii. Case 3: Three vertices are *dead* vertices. In this case only one face will be generated.

(d) Step 4: Implementing the *LMR* process.
The *LMR* process is implemented to deal with the T-junction (cracking) problem (referring to Section 4.5).

In the above modified subdivision process, both the number of vertices and the area of the *dead* faces become larger after every subdivision process. The densities of meshes at *dead* areas are kept when the *dead* regions are found.

4.7. Analysis of results

We now take a complicated mesh of a face and compare our adaptive scheme with the normal approximating scheme at three levels of subdivision. Fig. 7 shows the comparison of the normal Loop scheme at three levels (left) and using our adaptive scheme at three levels of subdivision (right). It can be seen that for data that has significant curvature changes and irregularity our method is a better way of adaptive subdivision as the whole mesh is filled with irregularities.

With our method, we can efficiently decrease the number of meshes generated in every subdivision step while keeping the continuity property of subdivision surface. We can also create surfaces that are more densely subdivided in the areas of higher curvature or in the special areas decided by user. The surfaces with different level of density of mesh can be got by mixing our adaptive subdivision scheme and the regular subdivision scheme. Suitable applications could be meshes used in character animation and industrial design prototypes.

We observe that adaptive refinement produce degenerate triangulations compared with the regular subdivision method. A suitable refinement of mesh realignment is suggested in this paper. Of course, realignment introduces some more computation.

5. Computational analysis of results

We compare the costs of our scheme in this section. Some schemes of a Stanford Bunny shape generated from the regular method and our method are illustrated in Fig. 8. The numbers

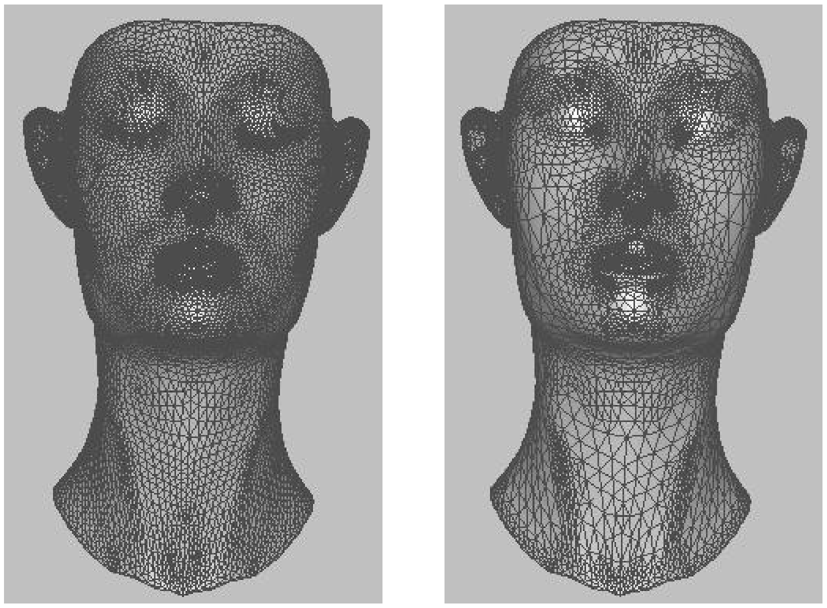


Figure 7: Using conical angle adaptive subdivision on a face model, the image on the left is obtained by using normal subdivision and the one on the right is obtained by our method

of triangles of the closed original polyhedron is 204. The number of triangles of every shape is shown with the scheme. Fig. 8(a),(b) are generated by the regular method. Fig. 8(c),(d) are generated by our method with CA value 22° after the third or fourth level subdivision, respectively. Fig. 8(e),(f) are generated by our method with CA value 11° after the third or fourth level subdivision, respectively. Fig. 8(g),(h) are generated by our method with CA value 7° after the third or fourth level subdivision, respectively.

It can be seen that in the areas with higher curvatures like paw, ears and head of the bunny, the densities of meshes in Fig. 8(b) and Fig. 8(h) are the same. In other areas, as the curvatures do not change quickly, though the densities of meshes generated by our method are lower than original subdivision method, the smoothness of the surface is kept well with fewer meshes.

Table 1 shows the subdivision steps and the number of meshes generated by the Loop subdivision method and by our method with $CA = 7°$, 11° and 22°, respectively. The numbers of are meshes listed in Table 1.

The reduction ratios of the number of meshes in the different cases are also shown. The more the subdivision process is done, the more dead faces are generated, the reduction ratio will increase. The subdivision process is on the coarse areas mainly, the different levels of mesh densities can be viewed clearly.

6. Conclusions

In this paper, we proposed an adaptive scheme for subdivision surfaces based on triangular meshes. Our method can be very well applied to the Loop and Modified Butterfly schemes. The scheme could also be extended to subdivision schemes that work on polygonal meshes like Catmull-Clark or Doo-Sabin. Our method is based on the analysis of the planar area created in subdivision surfaces. We can generate shapes with a smaller number of faces while keeping the same smoothness as the regular subdivision scheme (case $CA = 0°$ in

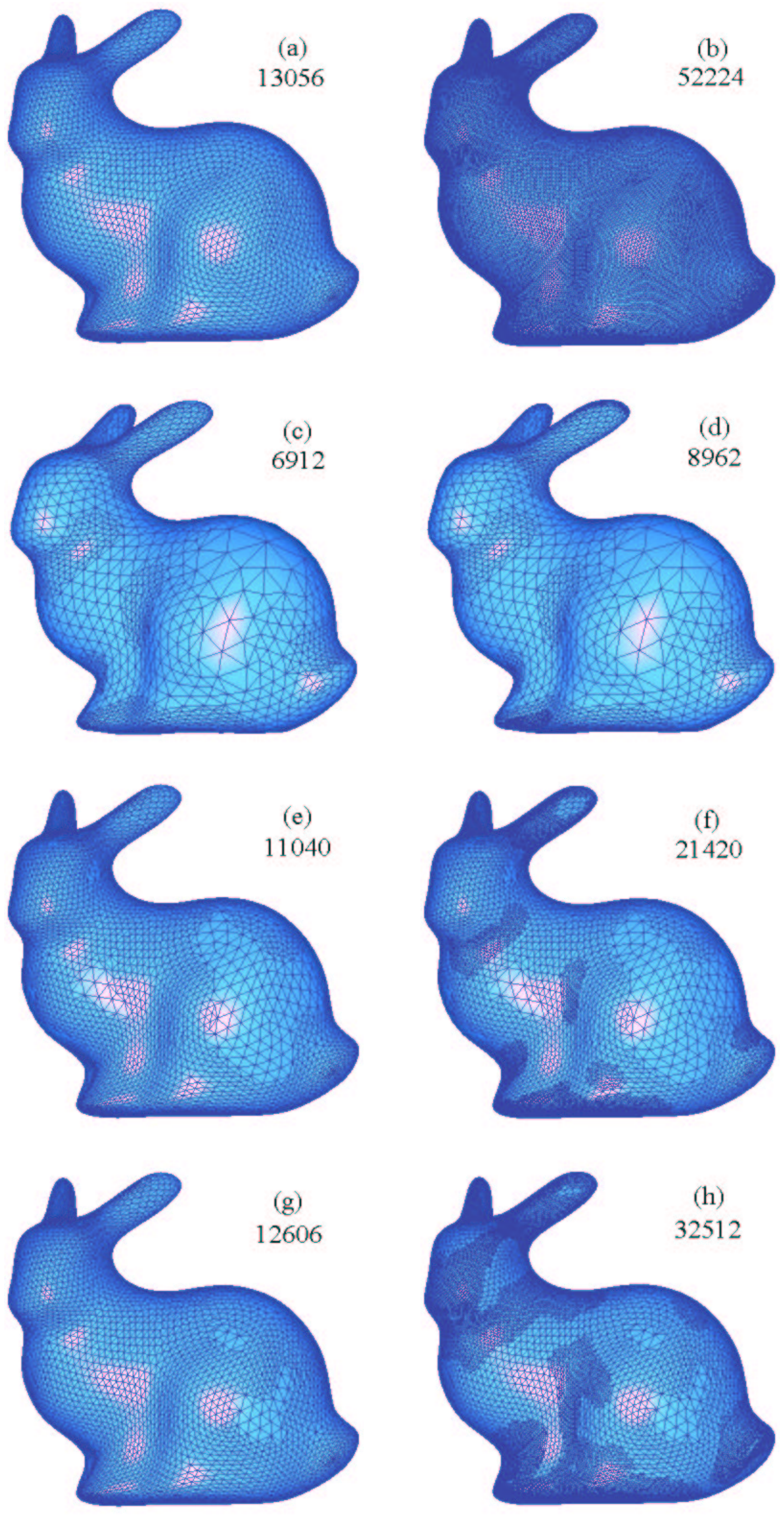


Figure 8: Stanford Bunny generated by the regular method and our method

	after 3 iterations	*after 4 iterations*
Loop	13056	52224
$CA = 7°$ *Reduction ratio*	12606 (3.45 %)	32512 (37.74 %)
$CA = 11°$ *Reduction ratio*	11040 (15.44 %)	21420 (58.98 %)
$CA = 22°$ *Reduction ratio*	6912 (47.06 %)	8962 (82.83 %)

Table 1: The mesh reduction rates of a Stanford Bunny model at different subdivision steps

our scheme). Under a reasonable CA, the obtained results are in accordance to the results obtained by normal subdivision. Local refinements are also possible by selecting areas on the mesh where refinements are desired. In our method, we take good care of the T-junction (cracking) problem and propose our solutions. According to the results of our experiment, the proposed method is certified efficient. The adaptive algorithm proposed here will strengthen the functions of Computer Graphics system which use subdivision surfaces to model surfaces.

References

[1] A. Amresh, G. Farin, A. Razdan: *Adaptive Subdivision Schemes for Triangular Meshes.* Hierarchical and Geometric Methods in Scientific Visualization, 2002.

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

[3] D. Doo, M. Sabin: *Behaviour of recursive division surfaces near extraordinary points.* Computer-Aided Design **10**, 356–360 (1978).

[4] N. Dyn, D. Levin, J.A. Gregory: *A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control.* ACM Transactions on Graphics **9**, 160–169 (1990).

[5] G. Farin: *Designing C1 Surfaces Consisting of Triangular Cubic Patches.* Computer-Aided Design **14**, 253–256 (1982).

[6] M. Halstead, M. Kass, T. DeRose: *Fair Interpolation Using Catmull-Clark Surfaces.* SIGGRAPH'93, 35–44, 1993.

[7] H. Hoppe, T. DeRose, T. Duchamp, et al: *Mesh Optimization.* SIGGRAPH'93, 19–26, 1993.

[8] L. Kobbelt: *Interpolatory Subdivision on Open Quadrilateral Nets with Arbitrary Topology.* Proc. of EuroGraphics'96, 409–420, 1996.

[9] L. Kobbelt: *$\sqrt{3}$ subdivision.* Computer Graphics Proceedings, 103–112 (SIGGRAPH' 2000).

[10] C. Loop: *Smooth Subdivision Surfaces Based on Triangles.* Master Thesis, University of Utah, Dpt. of Mathematics, 1987.

[11] J.M. Lounsbery: *Multiresolution Analysis for Surfaces of Arbitrary Topological Type.* PhD thesis, University of Washington, 1994.

[12] H. Muller, R. Jaeschke: *Adaptive subdivision curves and surfaces.* Proc. of Computer Graphics International, 48–58, 1998.

[13] A.H. Nasri: *Polyhedral Subdivision Methods for Free-Form Surfaces.* ACM Transaction on Graphics **6**, 29–73 (1987).

[14] J. Peters, U. Reif: *The simplest subdivision scheme for smoothing polyhedra.* ACM Trans. Gr. **16**(4), 420–431 (1997).

[15] T.W. Sederberg, J. Zheng, D. Sewell, M. Sabin: *Non-Uniform Recursive Subdivision Surfaces.* SIGGRAPH'98, 387–394, 1998.

[16] H. Seidel: *Polar forms and triangular B-Splines.* Tutorial notes in Pacific Graphics, 61–112, 1997.

[17] M. Vasilescu, D. Terzopoulos: *Adaptive meshes and shells: Irregular triangulation, discontinuities and hierarchical subdivision.* Proc. of the Computer Vision and Pattern Recognition conference, 829–832, 1992.

[18] Z. Xu, K. Kondo: *Adaptive Refinements in Subdivision Surfaces.* EuroGraphics'99, Short paper and demos, 239–242, 1999.

[19] D. Zorin, P. Schroder, et al: *Interpolating Subdivision for Meshes with Arbitrary Topology.* SIGGRAPH'96, 189–192, 1996.

[20] D. Zorin, P. Schroder, W. Sweldens: *Interactive Mulitiresolution Mesh Editing.* SIGGRAPH'97, 259–268, 1997.

Received October 28, 2003; final form May 12, 2004