Computer-Aided Design and Applications

http://www.cadanda.com

Adaptive Refinement for Unified Subdivisions with Sharp Features

Guiqing Li[1,2] and Weiyin Ma[1]

[1]City University of Hong Kong, mewma@cityu.edu.hk
[2]South China University of Technology, ligq@scut.edu.cn

ABSTRACT

This paper proposes a conceptually simple strategy for adaptive subdivision of unified subdivision surfaces with sharp features. A key feature of the proposed adaptive subdivision algorithms is that it follows exactly the same rule of the respective subdivision, which ensures correct evaluation of both refined control vertices and the corresponding limit positions. Another important feature is that it provides a universal API for unified subdivision surfaces with sharp features.

Keywords: subdivision surfaces, unified subdivisions, adaptive refinement.
DOI: 10.3722/cadaps.2009.851-864

1. INTRODUCTION

Subdivision surfaces are capable of defining a complex engineering object as a single surface using a control mesh or arbitrary topology. Crease features, such as sharp edges, corners and darts, can also be easily embedded on the surface. They have been widely used in the entertainment industry and graphics applications for multi-resolution modeling and animation [6]. Their applications to CAD have also been investigated systematically by researchers [19]. As face number increases exponentially, however, it is pertinent to explore adaptive refinement strategies for efficient tessellation and rendering of subdivision surfaces [11-12],[14],[29].

For proper implementation, several challenging issues must be addressed for adaptive subdivision. One of the important issues is then an appropriate criterion in order to determine regions for further refinement. As two faces sharing an edge may have a different subdivision level in adaptive subdivision, cracks will usually appear along mesh boundaries of different levels if there is no special treatment. Furthermore, extra data structure is needed in order to correctly evaluate vertices of a refined mesh in adaptive subdivision.

We can roughly classify the error control criteria into two classes. One is based on the geometry of control meshes and subdivision surfaces [3],[15],[17],[20],[23], [27],[30],[32] and the other relates to non-geometric properties, such as viewpoint, projected size, spatial region and so on [7],[10],[13],[21], [24-25],[31]. In literature, there are also adaptive strategies based on non-geometric metrics such as view dependent tessellation [22], ratio of curvature to the silhouette dot production [1], interest-value driven adaptive subdivision [10], directional light function error [13] and even combination of the above metrics [21],[24-25]. In addition, hardware support is also exploited for adaptive tessellation on the fly [2],[24-25].

Methods for correctly evaluating updated old vertices and newly inserted vertices can be put into three categories. Carefully exploiting the relationship between positions of a vertex in different subdivision levels, Müller and Jaeschke devised their adaptive scheme calculating vertices using the information of the adaptively refined mesh itself [20]. The second kind of approaches usually records enough neighboring information at the same subdivision level for areas to be modified [16],[24-25]. The last

category evaluates vertices using corresponding parametric representations of the subdivision surfaces [4],[15],[30],[32].
Cracks are often removed using local triangulation of the critical regions where faces of different levels meet together. A direct approach is to view the crack, a polygonal hole, as a new face [14]. Most of the methods annihilate the cracks by splitting the faces with lower levels. Dyken et al. introduced an elegant snap function to maintain consistency between regions with different resolution level [8]. Boubekeur and Schlick provided a generic adaptive refinement kernel for all kinds of representations such as B-spline surfaces, Berzier patches and subdivision surfaces [5]. These GPU-oriented approaches generally only perform semi-uniform adaptive strategy, namely, the depth may be different for initial polygonal facets but refinement should be uniform inside the facets.
Though many papers have been devoted to adaptive subdivisions, most of them aim at a specific subdivision scheme. Zorin and Schröder designed an adaptive strategy for their unified subdivision in [8], it is not clear whether the approach was applied to all schemes of the framework. Recently, Sovakar and Kobbelt [26] described an API design method for adaptive refinement of unified subdivisions based on a $\sqrt{3}$ splitting operator. Sharp-features were however not addressed in [5]. The vertex-centered adaptive subdivision of Smith [27] contributed to the treatment of sharp and semi-sharp features for specific subdivisions. The approach results in an undesirable feature ending up with arbitrary polygonal meshes for all schemes implemented so far. Moreover, only a simple example was presented in [27].
This paper presents a strategy for adaptive subdivision with implementation details. It aims at providing a general API for unified subdivisions with sharp features and semi-sharp features. Theoretically, all aforementioned criteria can be reformulated to our strategy. We adapt the normal angle and vertex distance error metrics and devise an edge error metric to test our approach in implementation. In addition, an ordinary face splitting style previously used in [3] is employed to split critical mesh faces for preventing cracks.

2. UNIFIED SUBDIVISION ON TRIANGLE MESHES

In connection with unified subdivision, Zorin and Schröder [33] and Stam [28] independently proposed a unified subdivision scheme generalizing tensor-product B-spline surfaces of higher degree. This section briefly describes the unified subdivision as an extension of Loop subdivision to which our adaptive strategy will be applied.
For a given mesh, *a linear 1-4 splitting operator* firstly inserts a vertex (*E-vertex*) into each edge, and then splits each face into four (see Fig. 1(a)). A refined mesh is produced after one round of splitting and repeated averaging operations. A *VF-type averaging operator* produces the centroid for each face from the corresponding corner vertices (see Fig. 1(b)), while *a FV-type averaging operator* produces a new position for each mesh vertex from the centroids of its neighboring faces (see Fig. 1(c)). Similarly, *a VV-type averaging operator* updates the position of a vertex as the weighted average of its adjacent vertices as well as itself (see Fig. 1(d)).

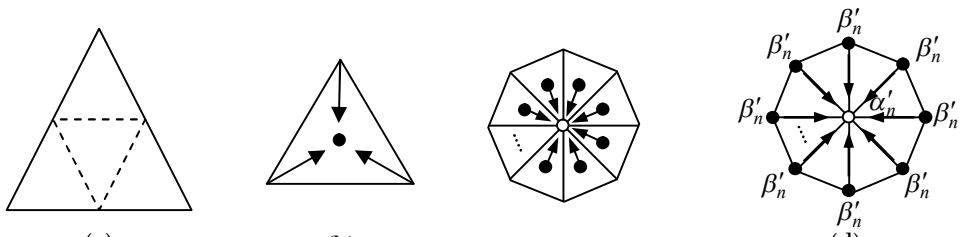


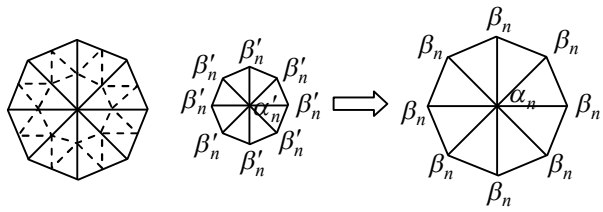
Fig. 1: Key operators for unified subdivision: (a) Face linear splitting; (b) VF-type averaging operation; (c) FV-type averaging operation; and (d) VV-type operation with weights.

2.1 Stam's VV-type Operation

Stam generalized three-direction box splines to arbitrary triangular meshes using a VV-type geometric operator (see Fig. 1(d)). The weights of the VV-type averaging operator are defined such that one linear

refinement and one step of VV-type operator (with d=1) is identical to Loop scheme (see Fig. 2). It results in [28]

$$\alpha_n' = \alpha_n - n\beta_n,\ \beta_n' = 2\beta_n,\ \ \beta_n = \frac{1}{n}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n}\right)^2\right)\quad \alpha_n = 1 - n\beta_n \tag{2.1}$$



(a) (b) (c)

Fig. 2: Unified subdivision with d=1 is equivalent to the Loop subdivision scheme: (a) linear refinement; (b) VV-type averaging operator on the refined mesh; and (c) V-vertex mask for Loop subdivision.

2.2 Stam's Unified Subdivision

Algorithm 1 in Tab. 1 presents one step of uniform refinement for the unified subdivision with d steps of the VV-type averaging operator. In the algorithm, function LinearSplit(Mesh) is the 1-4 linear splitting operator described above, the refined mesh is still stored in Mesh. Let $|V|$, $|E|$ and $|F|$ be the vertex number, edge number and face number of Mesh, respectively. As a new vertex is inserted into each edge, the total vertex number of the refined mesh should be $|V|+|E|$. In addition, the face number of the refined mesh increases up to $4|F|$. Give face i, suppose that $v_i[0]$, $v_i[1]$ and $v_i[2]$ are three vertices of i while and $e_i[0]$, $e_i[1]$ and $e_i[2]$ are indices of its three edges ($v_i[0], v_i[1]$), ($v_i[1], v_i[2]$) and ($v_i[2]$, $v_i[0]$), respectively. It is then partitioned into four new triangles (v[0], $|V|$+ e[0], $|V|$+ e[2]), (v[1], $|V|$+ e[1], $|V|$+ e[0]), (v[2], $|V|$+ e[2], $|V|$+ e[1]) and ($|V|$+ e[0] , $|V|$+ e[1], $|V|$+ e[2]) after a step of 1-4 splitting. Algorithm 2 in Tab. 1 shows the linear splitting operator.

Algorithm 1 UniformUnifiedSubdivision
Input: Initial mesh "Mesh"; and number of averaging operations d
Output: Refined mesh also stored in "Mesh".

```
1  LinearSplit(Mesh);              // Producing a linearly refined mesh.
2  for (i = 0; i < d; i++)         // Performing averaging operation
3      VVOperator(Mesh);           // on refined mesh "Mesh".
```

Algorithm 2 LinearSplitting
Input: Initial mesh "Mesh"; and number of averaging operations d
Output: A refined mesh through the linear splitting operator.

```
1  for (i = 0; i < |E|; i++)
2      Append the midpoint of edge i to the vertex array;
3  for (i = 0; i < |F|; i++) {
4      Append (v[0], |V|+ e[0], |V|+ e[2]) to the face array;
5      Append (v[1], |V|+ e[1], |V|+ e[0]) to the face array;
6      Append (v[2], |V|+ e[2], |V|+ e[1]) to the face array;
7      Append (|V|+ e[0] , |V|+ e[1], |V|+ e[2]) to the face array;
8  }
```

Tab. 1: Uniform refinement for unified subdivision.

2.3 Boundaries and Sharp Features Modeling

We follow the idea of [9] to model sharp features for the above unified subdivision. An edge of the given control mesh is called a *crease edge* if it corresponds to a segment of curve on which the subdivision surface is only C^0 continuous. Otherwise the edge is called a *smooth edge.* The control vertices are put into four categories: *smooth vertices*, *darts*, *crease vertices* and *corners*. A dart belongs to only one crease edge; a crease vertex is adjacent to two crease edges while a corner is the common vertex of at least three crease edges. All other vertices are smooth vertices. To ensure that the limit surface is also smooth at darts, special care is needed when computing E-vertices of crease edges near darts. Initially, smooth vertices, darts, crease vertices and corners are tagged with types 0, 1, 2 and 3 respectively. Let d be the number of averaging operations. We first replace the type of crease vertices and corners with $d+2$ and $d+3$ respectively. For each dart, we then update the tags of crease vertices whose distance to the dart via crease edges is d or less as the distance plus 1. Notice that if a corner is encountered, the procedure is then terminated. Fig. 3 depicts how the tag change for $d=2$. In mesh splitting, linear refinement is applied to both crease and smooth edges while all old vertices are interpolated. The type of crease vertices and newly inserted vertices on crease edges changes as follows. Let $t\left(v^L\right)$ be the type of a crease v^L. After a step of refinement, $t\left(v^L\right)$ is replaced by

$$t\left(v^{L+1}\right)=\min\left\{d+2,\ 2t\left(v^L\right)-1\right\}. \tag{2.2}$$

As for E-vertex v_E^L of a crease edge (v_0^L, v_1^L), it is tagged with type

$$t\left(v_E^{L+1}\right)=\min\left\{d+2,\ 2t\left(v_0^L\right),\ 2t\left(v_1^L\right)\right\}. \tag{2.3}$$

Fig. 3(d) shows the updated crease type of the crease vertices of Fig. 3(c) and crease type of newly inserted crease vertices when refining the crease edges of Fig. 3(c).

In each smoothing step, Stam's VV-type averaging operator as shown in Fig. 2(b) is applied to vertices whose type is not greater than d. Crease vertices of type $d+1$ are calculated using the curve averaging operator shown in Fig. 4 while corners remain unchanged. We have mentioned that the unified subdivision reduces to Loop scheme for d =1. However, a slightly different rule should be devised for internal edges incident to boundary or crease vertices for Loop scheme in order to sustain C^1 continuity of the subdivision surface on boundaries or creases [31].

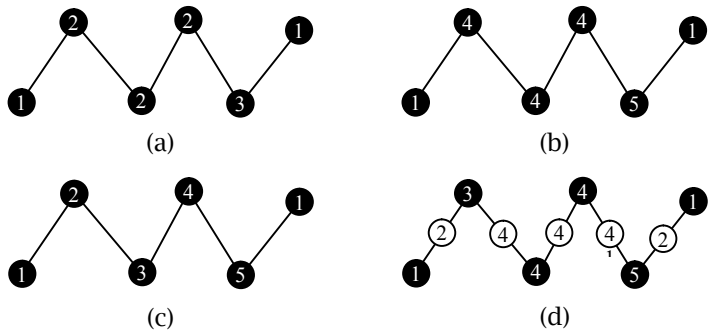


Fig. 3: Modification of vertex labels/types for d = 2: (a) the original labels; (b) all crease vertices are tagged with type d+2 = 4 and corners are tagged with d+3 = 5; (c) crease vertices whose distance to a dart is not greater than d are tagged with the distance plus 1; and (d) labels of crease vertices after a step of linear subdivision for the crease of (c).

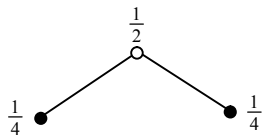
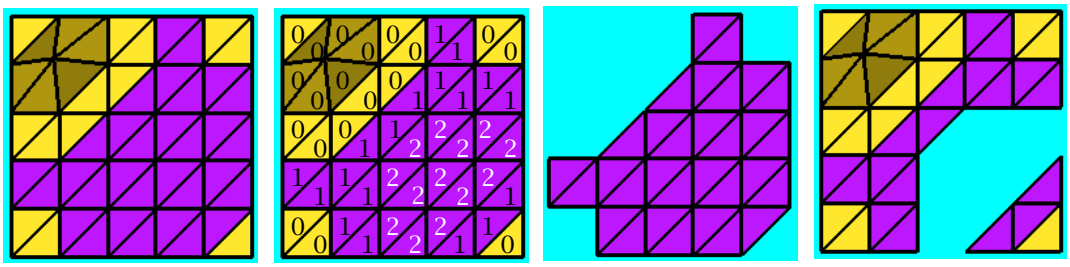


Fig. 4: Averaging operator for boundary and crease vertices.

3. THE PROPOSED ADAPTIVE SUBDIVISION STRATEGY

Our approach is a generalization of the strategy described in [16]. After each level of adaptive subdivision, we partition the mesh faces into two sets, i.e. divisible and indivisible faces. Divisible faces should be further refined. As all faces in the divisible mesh have the same subdivision level and, hence, only the ordinary uniform subdivision is performed to refine them further. However, in order to correctly perform refinement near boundaries, we need to maintain a number of extra layers of faces around the boundaries of the divisible mesh (see section 4).

Fig. 5 illustrates the process of partitioning an initial mesh into divisible and indivisible meshes for further adaptive subdivision. Fig. 5(a) shows all divisible faces (yellow) and indivisible faces (purple) after evaluation. In Fig. 5(b), all divisible faces are marked with label 0, while all indivisible faces are marked with label 1 if they fall within the layer of immediate neighbors of the divisible faces, or label 2 otherwise (for the case of $d = 1$, with d being number of averaging of the unified subdivision). Vertices can also be marked in a similar way for $d > 1$. All new indivisible faces are further added to the indivisible mesh as shown in Fig. 5(c) (the initial set of indivisible faces is a zero set in this case). Indivisible faces with a label 2 (d+1) are removed from the divisible mesh and the resulting updated divisible mesh is shown in Fig. 5(d) for further adaptive subdivision.



(a) (b) (c) (d)

Fig. 5: Mesh splitting: (a) faces are tagged with divisible (yellow) and indivisible (purple); (b) all divisible faces are marked with label 0, while indivisible faces are marked with positive labels; (c) all newly found indivisible faces are added to the indivisible mesh (purple), while (d) all divisible faces (yellow) and a portion of indivisible faces (purple) remain in the divisible mesh.

4. INDIVISIBLE FACE DICISION

We consider two types of criteria to determine whether a face needs to split further. The first criterion is to test the planarity of the mesh face, while the second criterion is used to verify whether the position of a vertex meets a pre-specified termination error.

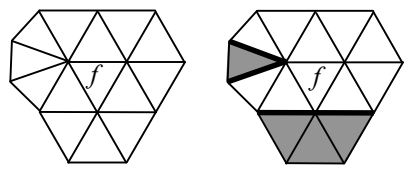
4.1 Planarity of Faces

We slightly make a modification to the approach of Amresh and Farin [8] by extending the set of neighboring faces. Let $N_F(f)$ be the 1-ring face neighborhood of f, namely the set consisting of all faces sharing at least one vertex with f as well as f itself, as shown in Fig. 6. Denote the unit normal of f by $\mathbf{n}(f)$. The planarity of f is defined as

$$C_A(f) = \max_{g \in N_F(f)-\{f\}} \{1 - \mathbf{n}(g)\cdot\mathbf{n}(f)\}. \quad (4.1)$$

For a given threshold $\varepsilon > 0$, we say f is indivisible if $C_A(f) < \varepsilon$, and divisible otherwise. If a threshold is specified by the normal angle θ_T between two faces, it can be converted into the ε − threshold as follows:

$$\varepsilon = 1 - \cos\theta_T. \quad (4.2)$$



(a) (b)

Fig. 6: Criteria for planarity: (a) the 1-ring neighborhood of face f; (b) the 1-ring neighborhood of face f in the presence of crease features. Only white faces / triangles are used for computing the planarity $C_A(f)$ of f having crease vertices.

However, further exploration is necessary for those f containing a non-smooth vertex. To clearly describe this case, we firstly define a face path on $N_F(f)$ (Fig. 6). Let $f_0, f_1, \cdots, f_{n-1}$ be a sequence of n faces such that f_i and f_{i+1} share an edge for $i = 0,1,\cdots,n-2$, we call the sequence a face path from f_0 to f_{n-1}. If there is no crease edge shared by these faces, we say that the path is a crease-free face path. On the submesh consisting of faces of $N_F(f)$, we introduce a notation $g \triangleright f$ which denotes the case that there is a crease-free face path from g to f. The planarity of f is adapted to

$$C_A(f) = \max_{g \in N_F(f)-\{f\} \wedge g \triangleright f} \{1 - \mathbf{n}(g) \cdot \mathbf{n}(f)\}. \tag{4.3}$$

The divisibility of a face having crease or boundary edges depends on not only the planarity of the face itself, but also the divisibility of the associated crease or boundary edges. Suppose that v is a crease vertex and (v', v) and (v'', v) are two crease edge incident to it. Let us further introduce the following parameter

$$C_A(v) = 1 - \frac{v - v'}{\|v - v'\|} \cdot \frac{v'' - v}{\|v'' - v\|}. \tag{4.4}$$

In case that v is not a crease vertex, we set $C_A(v) = 0$. The planarity of a crease edge (v_0, v_1) is then defined as

$$C_A(e) = \max\{C_A(v_0), C_A(v_1)\}. \tag{4.5}$$

The planarity of e can be defined similarly if e is a boundary edge. Also, introduce $P(e) = 0$ for an internal smooth edge e. It finally results in the following extended planarity of face f.

$$C_A(f) = \max\left\{ \max_{g \in N_F(f)-\{f\} \wedge g \triangleright f} \{1 - \mathbf{n}(g) \cdot \mathbf{n}(f)\}, \max_{e \in E\{f\}} \{C_A(e)\} \right\}, \tag{4.6}$$

where $E(f)$ is the edge set of f.

We have pointed out that the normal angle criterion is a C^1 condition. However, there exists the possibility in practice that some refined areas may violate the C^1 continuity constraint. To make the algorithm robust, we further introduce an auxiliary criterion to terminate the face splitting. A face is considered indivisible if the length of all edges of a face is less than the threshold ε. Specifically, for a given face $f^{(l)} = (v_0^{(l)}, v_1^{(l)}, v_2^{(l)})$ generated after l steps of subdivision, this auxiliary criterion for determining whether the face is indivisible is defined as

$$\max_{i=0,1,2} \left\{ \left\| v^{(l)}_{\mathrm{mod}(i+1,3)} - v_i^{(l)} \right\| \right\} < \varepsilon. \tag{4.7}$$

If a face satisfies Eq. (4.7), it is regarded as indivisible and not needed to further test the normal angle between the face and its neighbors. Note that the criterion can be disabled by setting $\varepsilon = 0$.

Also, Eq. (4.7) is a scale-dependent constraint. Therefore, in order to make the strategy comparable for models with different scales, we introduce a so-called relative error r to replace ε. Let $B_l = (x_{\min}, y_{\min}, z_{\min})$ and $B_u = (x_{\max}, y_{\max}, z_{\max})$ be the two extreme corner points of the bounding box of the control mesh with minimum and maximum coordinates, respectively. Error terms r and ε have the following relationship

$$\varepsilon = r\|B_u - B_l\|. \tag{4.8}$$

4.2 Vertex Error Metric

With the increase of averaging times, the termination condition described in section 4.1 becomes inaccurate due to the fact that the underlying subdivision masks are much larger than the 1-ring neighborhood of a mesh face. Here we describe another criterion based on vertex distance. We then adapt the criterion to general subdivisions for which the explicit vertex limit is unavailable.

Given face $f^{(l)}=(v_0^{(l)},v_1^{(l)},v_2^{(l)})$ generated after l steps of subdivision, assume that $v_0^{(\infty)},v_1^{(\infty)},v_2^{(\infty)}$ are the limit positions of the three V-vertices of the face. Denote a vertex error metric as

$$D_V\left(f\right)=\max_{i=0,1,2}\left\{\left\|v_i^{(\infty)}-v_i^{(l)}\right\|\right\}. \tag{4.9}$$

Obviously, $D_V\left(v^{(l)}\right)$ converges to zero when $l\to\infty$. When $D_V\left(v^{(l)}\right)$ becomes small enough, we may assume that the refined mesh has closely approximated to its limit surface. Therefore it can be employed to determine the divisibility of mesh faces. Given a threshold ε, f is indivisible if $D(f)<\varepsilon$, and divisible otherwise. Obviously, error ε can be replaced by the relative error defined in Eq. (4.8) as well.

The unified scheme discussed in this paper reduces to Loop subdivision for averaging time $d=1$. The limit behavior has been addressed by Loop for the original Loop scheme [18] and by Hoppe et al. for the extension version with sharp features [9]. For completeness, we list the results here. Fig. 7(a) and (b) are the masks for computing the limit position of smooth/dart vertices and crease/boundary vertices, respectively, in which

$$\overline{\alpha}_n=\frac{3}{3+8\alpha_n}\ \text{and}\ \overline{\beta}_n=\frac{8\alpha_n}{n\left(3+8\alpha_n\right)}. \tag{4.10}$$

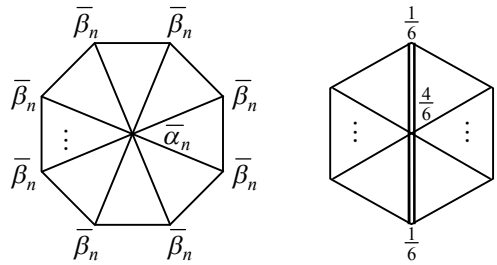


Fig. 7: Limit masks for Loop subdivision: (a) the limit mask for internal vertices; (b) the limit mask for crease and boundary vertices.

Though the limit position of a control vertex can be computed analytically or numerically for a specific subdivision scheme it is not available for general unified subdivisions. We introduce the following metric as an alternative criterion in this case

$$D_V\left(f^{(l)}\right)=\max_{i=0,1,2}\left\{\left\|v_i^{(l+1)}-v_i^{(l)}\right\|\right\}, \tag{4.11}$$

where $v_0^{(l+1)},v_1^{(l+1)}$ and $v_2^{(l+1)}$ are the three V-vertices corresponding to $v_0^{(l)},v_1^{(l)}$ and $v_2^{(l)}$.

4.3 Edge Error Metric

Noting that the vertex error metric does not work for interpolatory subdivisions for which control vertices are on the limit surface. We explore another error metric, the so-called edge error metric such that the strategy is capable of handling both approximatory and interpolatory subdivisions. Also, assume that $v_0^{(\infty)},v_1^{(\infty)},v_2^{(\infty)}$ are the limit positions of the three V-vertices of a given face $f^{(l)}=\left(v_0^{(l)},v_1^{(l)},v_2^{(l)}\right)$. Furthermore, denote $\overline{v}_0^{(\infty)},\overline{v}_1^{(\infty)},\overline{v}_2^{(\infty)}$ the limit positions of the three E-vertices corresponding to edges $e_i^{(l)}=\left(v_i^{(l)},v_{\mathrm{mod}(i+1,3)}^{(l)}\right)$ ($i=0,1,2$), respectively. The direction of line segment $e_i^{(\infty)}=\left(v_i^{(\infty)},v_{\mathrm{mod}(i+1,3)}^{(\infty)}\right)$ can be evaluated by

$$\mathbf{e}_{i,\mathrm{mod}(i+1,3)}=\frac{v_{\mathrm{mod}(i+1,3)}^{(\infty)}-v_i^{(\infty)}}{\left|v_{\mathrm{mod}(i+1,3)}^{(\infty)}-v_i^{(\infty)}\right|}. \tag{4.12}$$

The edge error metric is then defined as the maximum of the orthogonal distances from limit position $\overline{v}_i^{(\infty)}$ of $e_i^{(l)}$ to line segment $e_i^{(\infty)}$, $i=0,1,2$:

$$D_E\left(f\right)=\max_{i=0,1,2}\left\{\left\|\left(\overline{v}_i^{(\infty)}-v_i^{(\infty)}\right)-\left(\left(\overline{v}_i^{(\infty)}-v_i^{(\infty)}\right)\cdot\mathbf{e}_{i,\mathrm{mod}(i+1,3)}\right)\cdot\mathbf{e}_{i,\mathrm{mod}(i+1,3)}\right\|\right\}. \tag{4.13}$$

For given threshold ε, f is indivisible if $D_E(f) < \varepsilon$ and divisible otherwise. In this case, the relative error given in Eq. (4.8) also works. Certainly, for subdivisions whose limit behavior is unknown, Eq. (4.13) can be modified to

$$D_E(f) = \max_{i=0,1,2} \left\{ \left\| \overline{v}_i^{(l+1)} - \tfrac{1}{2}(v_i^{(l)} + v_{i+1}^{(l)}) \right\| \right\}, \tag{4.14}$$

where $v_i^{(l+1)}$ is the same as that in Eq. (4.11), while $\overline{v}_i^{(l+1)}$ is the E-vertex of edge $\left(v_i^{(l)}, v_{\mathrm{mod}(i+1,3)}^{(l)}\right)$.

4.4 Classification of Faces and Vertices

As described in section 3, we do not remove all indivisible faces from the divisible mesh after its faces being classified into divisible and indivisible ones in order to correctly subdividing the divisible faces in the next step of refinement. Let d be the number of averaging operations involved. Then d layers of faces around a face f are required in order to achieve this goal. In other words, those indivisible faces located within the d-ring neighborhood of f should be kept in the divisible mesh. Therefore we assign a label, which ranges from 0 to d, to each vertex and each face of the divisible mesh firstly. Initially the label of f is assigned as follows

$$L(f,0) = \begin{cases} 0 & f \text{ is divisible} \\ 1 & f \text{ is indivisible} \end{cases}. \tag{4.15}$$

And the label of vertex v of a face is given by

$$L(v,0) = \min_{f \in N_F(v)} \{L(f,0)\}. \tag{4.16}$$

Now assume that $L(f,j)$ and $L(v,j)$ have been calculated for all faces and vertices. We update labels according to the following rules:

$$L(f,j+1) = \min_{v \in V(f)} \{L(v,j)\}, \quad L(v,j+1) = \min_{f \in N_F(v)} \{L(f,j+1)\}. \tag{4.17}$$

$L(f,d)$ and $L(v,d)$ are then the appropriate labels of faces and vertices of the divisible mesh.

5. CRACKS PREVENTION

If a face in the indivisible mesh shares an edge with a divisible face in the divisible mesh, cracks appear after the divisible face is refined in the next round of subdivision. Some indivisible faces of the indivisible mesh are also refined according to the number of the divisible faces sharing the same edge with the indivisible face in order to annihilate the cracks. The treatment is initiated after the divisible mesh is subdivided because newly inserted vertices into the shared edges are produced at this stage. In general, each indivisible face has four possible splitting patterns which respectively correspond to the number of neighboring divisible faces sharing an edge with the indivisible face being 0, 1, 2 and 3 as shown in Fig. 8.

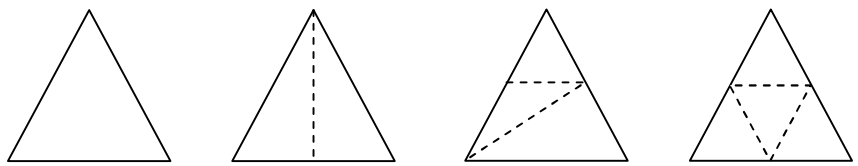


Fig. 8: Splitting patterns for boundary indivisible faces when shared with (a) 0, (b) 1, (c) 2 and (d) 3 edges with neighboring divisible faces.

6. SEMI-SHARP CREASES HANDLING

In uniform subdivisions, semi-sharp creases are produced by refining the mesh through applying geometric rules for generating sharp features to regions tagged with sharp features in initial several steps of subdivision and then performing smooth rules in the rest of subdivision. Hence, in adaptive refinement, we always refine those faces adjacent to sharp feature vertices in the initial several steps

of subdivision, namely, tag these faces as divisible regardless their planarity. We denote the number of the initial steps by τ. Fig. 13 illustrates an example with $\tau = 2$.

7. RESULTS AND DISCUSSIONS

As an example, we implemented the proposed adaptive refinement of unified subdivisions operating upon triangle meshes. The implementation was realized using a Thinkpad T60p laptop PC with 2.0GHz CPU and 512MB RAM. When the number of averaging operations for each level of refinement equals to one, the unified subdivision on triangle meshes produces Loop subdivision surfaces. In this case, we also tested the strategy for Loop subdivision using the criterion defined by Eqs. (4.8)-(4.9) which prevents a face from further splitting using the distance between vertices as well as midpoints of edges of the face and their limit positions. In the following we demonstrate the versatility of the proposed approach by applying the proposed adaptive subdivision to a variety of models.

Fig. 9 depicts the results of different unified subdivisions with 1, 2, and 3 steps of averaging operations. All results are generated under a normal angle threshold $\theta_T = 5°$. Fig. 10 illustrates an example for adaptively subdividing open models with also $\theta_T = 5°$. Both examples demonstrate that the resolution levels uniformly increase from the area with lower curvature to the area with higher curvature. In fact, the proposed strategy generally produces adaptively refined meshes with less difference of subdivision levels of adjacent faces for models without sharp features. The proposed strategy can produce an adaptive triangle mesh with good aspect ratio for this kind of models.

The situation is different for adaptive refinement of models with sharp features. It is possible that a flat region and a highly curved region are separated by a crease on such models. This leads to the connection of faces with much lower resolution (in a flat region) with faces of higher resolution (in a curved region) through a crease edge. The proposed strategy tends to produce narrow triangles along crease edges in this situation since it has to split the faces with lower resolution multiple times along the crease edge in order to avoid cracks. This can be observed from the examples shown in Figs 11-12. The rendering quality is satisfactory as seen in Fig. 11(c) and Fig. 12(c). Selected experimental data of Figs 9-12 are summarized in Tab. 2. In this table, V num and F num represent the number of vertices and the number of faces of the associated mesh, respectively, θ_T is the normal angle threshold, and d is the steps of averaging operations picked up from the corresponding figures. Also note that a relative error threshold r is also used in these examples, but as a very small threshold value is used, the results are mainly dictated by the normal angle threshold θ_T.

Models	*Initial meshes*		*Adaptive meshes*		θ_T	r	d	*Max. depth*	*Time (s)*
	V num.	*F num.*	*V num.*	*F num.*					
Bunny (Fig. 9)	456	904	68746	137380	5°	4e-6	2	7	8.48
Engine (Fig. 10)	194	332	26108	51703	5°	0	2	8	3.52
Fandisk (Fig. 11)	204	404	38062	76120	2°	0	1	12	14.86
Mechpart (Fig. 12)	229	466	299442	598892	4°	4e-13	1	19	569.50

Tab. 2: Experimental results using the normal angle criterion.

Fig. 13 presents an example for generation of semi-sharp features with an initial mesh of 48 faces and 26 vertices and with $d = 1$ averaging operation. The adaptively refined mesh, which consists of 144712 faces and 72358 vertices, is generated under normal angle threshold $\theta_T = 5°$ only (vertex and edge metrics not used). In the first two rounds of subdivision, faces incident on crease vertices and corners are enforced to be divisible. The maximum depth of subdivision is 8 and a face compression ratio is 4.2% compared with that of uniform subdivision of 8 levels. The face compression ratio refers to the ratio of face number of adaptive subdivision over that of uniform subdivision. Fig. 14 illustrates some examples produced by applying vertex metric only or edge metric only to the above initial models. Related experimental data is listed in Tab. 3. The initial models used in this table are the same as those summarized in Tab. 2.

Regarding the selection of appropriate error metrics, both vertex and edge error metrics can be used to produce an adaptive mesh with a roughly expected relative mesh error, though the relative error threshold r is actually not a direct measure for model accuracy. Experimental results also show that such metrics tend to yield results with small difference in resolution between adjacent faces than that using the normal angle metric. This can be observed by comparing Fig. 9(c), Fig. 10(c), Fig. 11(c) and Fig. 12(c) with Figs. 14(a)-(b), respectively. As the normal angle between adjacent faces reflects the most important visual property, it is generally easier to produce adaptive meshes with good visual appearance using the normal angle metric. If a visually high quality appealing adaptive mesh is needed, one should use the normal angle metric with an appropriate normal angle threshold θ_T for adjusting mesh resolution.

Models	*Adaptive meshes*		*Error metrics*	*Relative error threshold r*	*Steps of averaging operations d*	*Maximum subdivision depth*	*Face compression rate (%)*
	V num.	*F num.*					
Bunny	9580	19132	V	0.001	1	5	2.07
Engine	4955	9666	E	0.001	1	5	2.84
Fandisk	69892	139780	E	0.0001	1	11	0.01
Mechpart	59142	118292	E	0.0001	1	10	0.02

Tab. 3: Experimental results for the vertex / edge error criteria.

8. CONCLUSIONS

We described a universal strategy for adaptively refining general models with sharp features. A unified interface is proposed for the implementation of adaptive subdivisions of unified subdivision schemes based on a splitting operator and repeated averaging operations. Though only subdivisions based on the 1-4 splitting operator for triangular meshes are discussed in the paper, the approach is applicable to any other topological operators and polygonal meshes. As future work, it is useful to conduct a comparative study and further investigate the impact of different error criteria on the complexity and quality of the refined meshes of adaptive subdivision. On the other hand, this paper only considered primal subdivisions, while adaptive dual unified subdivisions remain open due to the difficulty in modeling sharp features with this kind of subdivisions. It is also an interesting topic to apply the proposed adaptive subdivision strategy for the manipulation of subdivision surfaces, such as intersection, trimming, Boolean operations and collision detection of subdivision based 3D models.

9. ACKNOWLEDGEMENT

The work described in this paper is supported by the following sponsors: City University of Hong Kong (SRG grant No. 7001928), Natural Science Foundation of Guangdong (05006540), the Research Grants Council of Hong Kong SAR (CERG grant No: CityU 1188/05E) and Natural Science Foundation of China (60373034).

10. REFERENCES

[1] Amor, M.; Bóo, M.; Doggett, M.: A meshing scheme for efficient hardware implementation of butterfly subdivision using displacement mapping, IEEE Computer Graphics and Applications, 25(2), 2005, 46-59.

[2] Amor, M.; Bóo, M.; Doggett, M.; Hirche, J.; Strasser, W.: Hardware support for adaptive subdivision surface Rendering. Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware, Los Angeles, 2001, USA, 33-40.

[3] Amresh, A.; Farin, G,: Adaptive subdivision schemes for triangular meshes. In Farin, G.; Hagen, H.; Hamann, B.; (eds): Hierachical and Geometric Methods in Scientific Visualization, Springer-Verlag, 2002, 319-327.

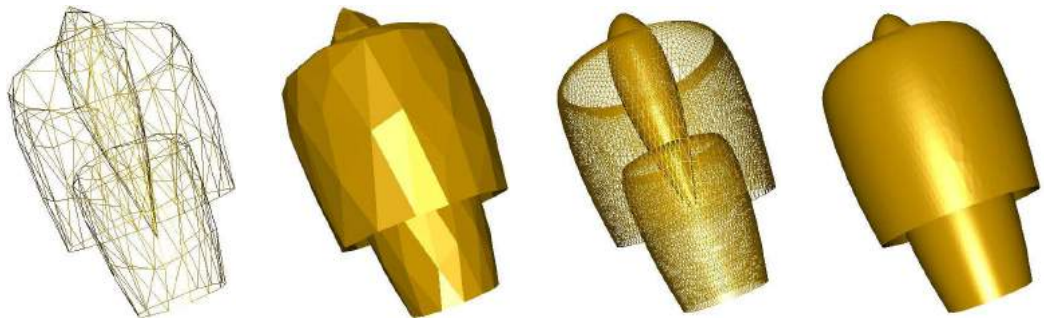
[4] Bolz, J.; Schröder, P.: Rapid evaluation of Catmull-Clark subdivision surfaces, Proceedings of the Seventh International Conference on 3D Web Technology, February 24-28, 2002, Tempe, Arizona, USA, pp. 11-17.

[5] Boubekeur, T.; Schlick, C.: A flexible kernel for adaptive mesh refinement on GPU, Computer Graphics Forum, 27(1), 2008, 102–114.
[6] DeRose, T.; Kass, M.; Truong, T.: Subdivision surfaces in character animation. In Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, 1998, 85-96.
[7] Doggett, M.; Hirche. J.: Adaptive view dependent tessellation of displacement maps, Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware, Interlaken, Switzerland, 2000, 59-66.
[8] Dyken, C.; Reimers, M.; Seland, J.: Semi-uniform adaptive patch tessellation, Computer Graphics Forum, to appear, 2008.
[9] Hoppe, H.; DeRose, T.; Duchamp, T.; Halstead, M.; Jin, H.; McDonald, J.; Schweitzer, J.; Stuetzle, W.: Piecewise smooth surface construction, Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, 1994, 295-302.
[10] Isenberg, T.; Hartmann, K.; König, H.: Interest value driven adaptive subdivision. In Simulation and Visualization, Magdeburg, Germany, SCS European Publishing House, 2003, 139-149.
[11] Kobbelt, L.: Interpolatory Subdivision on open quadrilateral nets with arbitrary topology, Computer Graphics Forum, 15(3) , 1996, C409-420.
[12] Kobbelt, L.: $\sqrt{3}$ -Subdivision, Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, 2000, 103-112.
[13] K□ivánek, J.; Pattanaik, S.; □ára, J.: Adpative mesh subdivision for precomputed radiance transfer, Proceedings of the 20th Spring Conference on Computer Graphics, ACM, 2004, 106-111.
[14] Labisk, U.; Greiner, G.: Interpolatory $\sqrt{3}$ -subdivision, Computer Graphics Forum, 19(3), 2000, 131-138.
[15] Lai, S.; Cheng, F.: Near-optimum adaptive tessellation of general Catmull-Clark subdivision surfaces, Lecture Notes in Computer Science, Vol. 4035, (Volume for CGI 2006: H.-P. Seidel, T. Nishita and Q. Peng (eds.), Advances in Computer Graphics), Springer-Verlag, Berlin / Heidelberg, 2006, pp. 562-569.
[16] Li, G.; Ma, W.: Interpolatory ternary subdivision surfaces, Computer Aided Geometric Design, 23(1), 2006, 45-77.
[17] Liu, W.; Kondo, K.: An adaptive scheme for subdivision surfaces based on triangular meshes, Jorunal for Geometry and Graphics, 8(1), 2004, 69-80.
[18] Loop, C.: Smooth subdivision surfaces based on triangles, Master's Thesis, University of Utah, Department of Mathematics, 1987.
[19] Ma, W.: Subdivision surfaces for CAD—an overview, Computer-Aided Design, 37(13), 2005, 693-709.
[20] Müller, H.; Jaeschke, R.: Adaptive subdivision curves and surfaces, Computer Graphics International, Hannover, Germany, 1998, 48-58.
[21] Müller, K.; Havemann, S.: Subdivision surface tesselation on the fly using a versatile mesh data Structure, Computer Graphics Forum, 19(3), 2000, 151-159.
[22] Müller, K.; Techmann, T.; Fellner, D.: Adaptive ray tracing of subdivision surfaces, Computer Graphics Forum, 22(3), 2003, 553-562.
[23] Pakdel, H. R.; Samavati, F.: Incremental Catmull-Clark subdivision, Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling, IEEE Computer Society Press, Ottawa, Ontario Canada, 2005, 95-102.
[24] Settgast, V.; Müller, K.; Fünfzig, C.; Fellner, D.: Adaptive tessellation of subdivision surfaces in OpenSG, OpenSG Symposium, The Eurographics Association, 2003.
[25] Settgast, V.; Müller, K.; Fünfzig, C.; Fellner, D.: Adaptive tessellation of subdivision surfaces, Computer & Graphics, 28(1), 2004, 73-78.
[26] Sovakar, A.; Kobbelt, L.: API design for adaptive subdivision schemes, Computer & Graphics, 28(1), 2004, 67-72.
[27] Smith, J.: Robust Geometric Methods for Surface Modeling and Manufacturing, Ph.D. Thesis. University of California, Berkley, 2004.
[28] Stam, J.: On subdivision schemes generalizing uniform B-spline surfaces of arbitrary degree, Computer Aided Geometric Design, 18(5), 2001, 383-396.
[29] Velho, L.; Zorin, D.: 4-8 Subdivision, Computer Aided Geometric Design, 18(5), 2001, 397-427.

[30] Wu, X.; Peters, J.: Interference detection for subdivision surfaces, Computer Graphics Forum, 23(3), 2004, 577-585.
[31] Wu, X.; Peters, J.: An accurate error measure for adaptive subdivision surfaces, Proceedings of Shape Modeling and Applications, MIT, USA, 2005, 51-56.
[32] Yong, J.-H.; Cheng, F.: Adaptive subdivision of Catmull-Clark subdivision surfaces, Computer-Aided Design & Application, 2(1-4), 2005, 253-261.
[33] Zorin, D.; Schröder, P.: A unified framework for primal/dual quadrilateral subdivision scheme, Computer Aided Geometric Design, 18(5), 2001, 429-454.

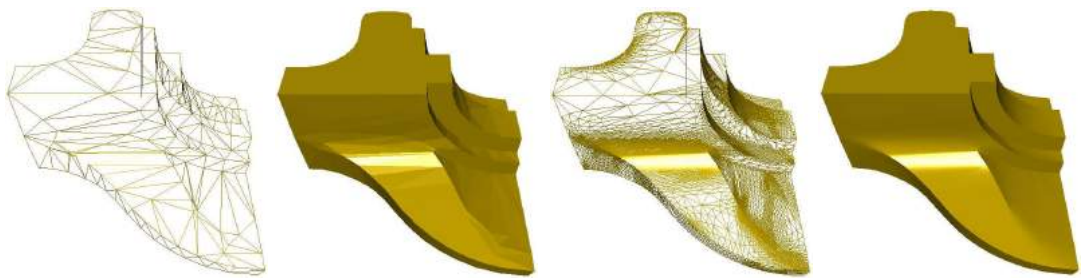
(a) (b) (c) (d)

Fig. 9: Adaptive subdivision of a bunny model: (a) the initial control mesh; (b)-(d) adaptively refined meshes corresponding to subdivisions with d = 1, 2 and 3 step(s) of averaging operation, respectively. All results are produced with a normal angle $\theta_T = 5°$ and relative error r=4e-6.



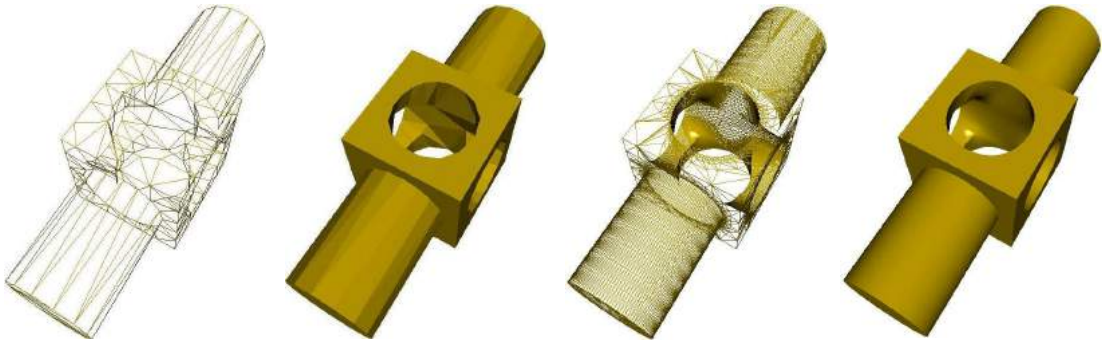
(a) (b) (c) (d)

Fig. 10: Adaptive subdivision of an engine model: (a) and (b) the initial control mesh in wire frame and shading, respectively; (c) and (d) the resulting mesh in wire frame and shading, respectively, generated with d = 2 using a normal angle $\theta_T = 5°$ and relative error r=0.



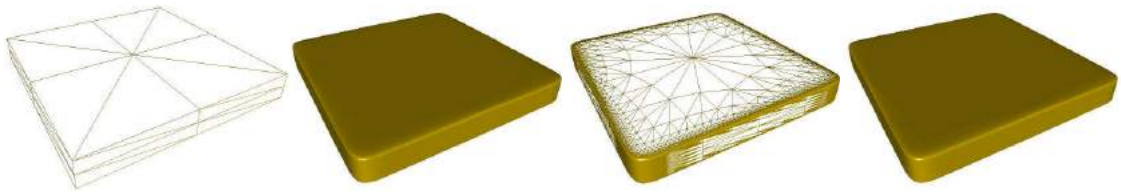
(a) (b) (c) (d)

Fig. 11: Adaptive subdivision with sharp features of a Fandisk model; (a)-(b) the initial control mesh in wireframe and shaded image, respectively, for each of the models; (c)-(d) the adaptively refined Fandisk model with $\theta_T = 2°$ and r=0.



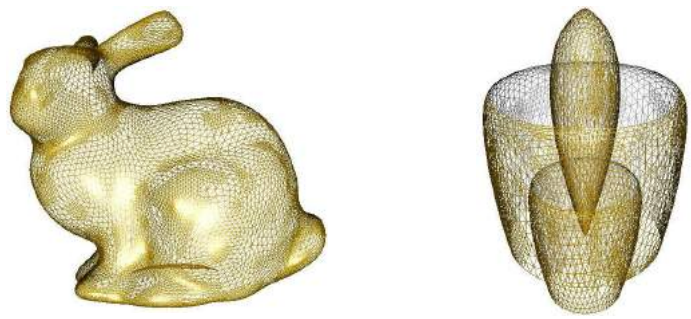
(a) (b) (c) (d)

Fig. 12: Adaptive subdivision with sharp features of a Mechpart model: (a)-(b) the initial control mesh in wireframe and shaded image, respectively, for each of the models; (c)-(d) the adaptively refined Mechpart model with $\theta_T = 4°$ and r=4e-13.



(a) (b) (c) (d)

Fig. 13: Adaptive subdivision with semi-sharp features: (a) the initial control mesh; (b) the subdivision surface generated by two steps of Loop subdivision with sharp features followed three steps of smooth Loop subdivision; (c) the adaptive subdivision result with semi_sharp_level=2 and normal angle $\theta_T = 5°$ only; (d) the shaded image of (c).



(a) (b)

Fig. 14: Adaptive subdivision using two different distance metrics for smooth models: (a) a Bunny model using a vertex error metric only; (b) an Engine model using an edge error metric only.