Incremental Subdivision for Triangle Meshes

Hamid-Reza Pakdel and Faramarz F. Samavati

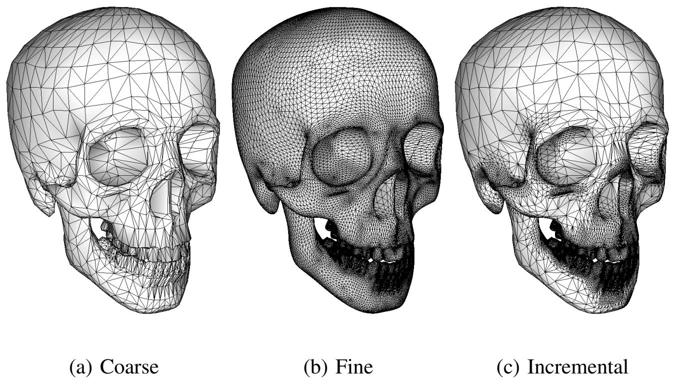


Fig. 1. Subdivision of the coarse mesh produces a smooth surface. Adaptive subdivision of the same mesh produces a smooth surface, but with fewer faces.

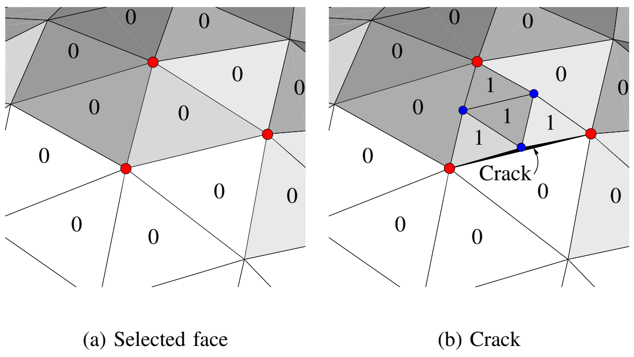


Fig. 2. Subdivision of a single face creates cracks where faces from different subdivision generations meet at an edge. The numbers represent the subdivision level that the faces belong to.

***Abstract*— We introduce incremental subdivision as a new adaptive subdivision method for triangle meshes. While regular (global) subdivisions produce a smooth surface from a given polygon mesh by refining all of its faces, adaptive subdivision produces a surface by refining only some selected areas of the mesh. Consequently, the selected area becomes fine and high resolution while the rest of mesh is coarse. Incremental subdivision produces a surface whose subdivided area is identical to when the entire mesh is subdivided regularly. In addition, as a good effect, the resolution of the produced surface gradually increases from coarse to fine. The incremental subdivision method expands the specified area to create a buffer region that is subdivided along with it. This method is efficient and easy to implement. We apply the incremental method to Loop and Butterfly subdivision schemes, and we compare it with other adaptive subdivision methods. We discuss some applications of incremental subdivision.**

***Index Terms*— Subdivision, adaptive, incremental, triangle mesh, Loop, Butterfly, anti-aliasing**

I. INTRODUCTION

A. Subdivision Surfaces

SUBDIVISION surfaces are increasingly used in computer modeling and animation packages. Subdivision is defined by simple operations that are uniformly applied to a given control mesh. Repeated application of these operations produces a sequence of meshes that converge to a surface that is smooth everywhere. Special subdivision rules enable construction of surfaces with varying smoothness. The subdivision operations are efficient and allow for intuitive modelling tools. In addition, they can be applied to arbitrary topology polygon meshes. Consequently, subdivision is replacing traditional modeling tools, such as NURBS and tensor product patches, in free-form solid and surface modeling applications [1].

In recent years, many subdivision schemes have been developed, including Doo-Sabin [2], Catmull-Clark [3], Loop [4] and Butterfly [5]. We have focused on Loop and Butterfly subdivision algorithms in this work. These subdivision schemes operate on triangle meshes, and are commonly used in modeling, rendering and animation systems.

B. Motivation

Starting with the input polygon mesh, each subdivision step refines its faces and repositions its vertices to produce a smooth surface at the limit. Repeated subdivisions increase the number of faces exponentially, which quickly leads to heavy computation load. In some applications it is not necessary to refine the entire mesh. For example, visualization of 3D models requires refinement of only the viewable and detailed regions of the mesh. Designers sometimes need to focus on an area of the model they are working on. Adding features to a mesh requires an increase in the level of detail where the feature is to be added. In these cases, subdivision of the entire mesh would be inefficient because the large number of subdivided faces taxes the CPU, system bus and the graphics subsystem.

Adaptive subdivision is the refinement of a subset of the faces of the control mesh. An example of adaptive subdivision is given in Fig. 1. The area to be subdivided is determined either by the user or by the application. For example, only high curvature regions of the mesh are subdivided for fast rendering of subdivision surfaces. In modeling applications, the areas selected by the designer are adaptively subdivided for finer editing.

Hamid-Reza Pakdel, University of Calgary
Faramarz F. Samavati, University of Calgary

C. Problem Statement

Though the idea of adaptive subdivision is simple, care must be taken to handle connectivity and geometrical inconsistencies that arise when a subset of the faces are subdivided. As shown in Fig. 2, a single adaptive subdivision step creates cracks in the mesh where two triangles from different subdivision generations meet at an edge. A 2-manifold mesh is conforming if the intersection of any two faces is a line segment or the empty set [6]. A mesh with cracks is non-conforming. Cracks must be removed for proper rendering, editing, animation and subdivision of the mesh.

Generally, the faces containing the cracks are triangulated to produce a conforming mesh. The triangulation changes the connectivity of the vertices inside the selected subdivision area. Because the positions of the vertices are computed as a weighted sum of their neighbours, a change in the connectivity of the vertices within the selected subdivision area affects the subdivision process, and leads to geometrical inconsistency in the limit surface. The goal of adaptive subdivision is to produce, from the selected region of the mesh, a limit surface that is exactly the same as when the entire mesh is subdivided. Therefore, all vertices within the selected subdivision area must have the same connectivity as when the entire mesh is subdivided.

Another effect of adaptive subdivision is that repeated refinements of a region exponentially increases the relative resolution of that area with respect to the rest of the mesh. The sudden change of resolution is similar to aliasing in images and creates artifacts when the model is rendered. It also makes editing the model difficult. It is desired that the adaptively subdivided mesh is balanced. In a balanced mesh adjacent triangles differ by up to one subdivision generation [7]. The result is a gradual increase in the resolution from coarse to fine regions of the mesh.

Although there are existing algorithms that deal with some of the issues noted above, they either provide a partial solution or are inefficient. We introduce *incremental subdivision* as an efficient and comprehensive solution to these problems. In particular, incremental subdivision has been designed to produce adaptive subdivision surfaces with the following properties:

- consistent connectivity,
- consistent geometry,
- gradual change of resolution throughout the surface.

Incremental subdivision is fast and does not require a complicated data structure. It is also intuitive and simple to implement. The properties of the algorithm make it suitable for modeling applications, and the produced surfaces can be used in high quality renderings and animations.

The rest of this paper is organized as follows. In Section II we give an overview of Loop and Butterfly subdivision schemes and briefly discuss other relevant triangle subdivisions. Section III provides a background of existing adaptive subdivision algorithms. Incremental subdivision is described in detail in Section IV, and is compared to other adaptive subdivision methods. Results and some applications of incremental subdivision are presented in Section V.

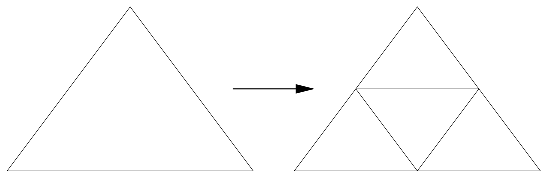
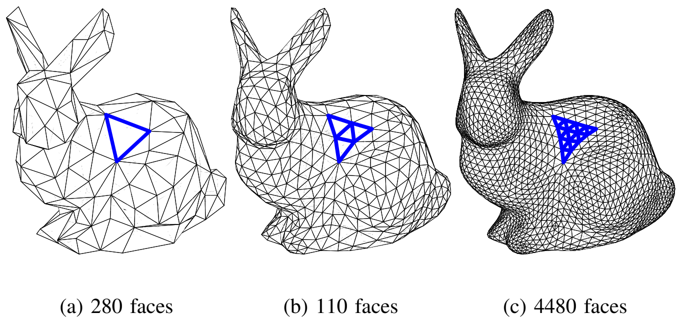


Fig. 3. One-to-four refinement in Loop and Butterfly subdivisions



(a) 280 faces (b) 110 faces (c) 4480 faces

Fig. 4. Three levels of Loop subdivision of a coarse mesh. The highlighted edges show the 1-to-4 refinement.

II. BACKGROUND

Subdivision is the combination of refinement and averaging operations that are recursively applied to a control mesh, producing a smooth surface at the limit. In this section, we give an overview of Loop and Butterfly subdivision schemes. We chose these schemes because they are commonly used refinement algorithms. These subdivision algorithms share the same refinement operation, but the averaging operators that define new vertex positions are different. Loop subdivision produces a surface that is an approximation of the control mesh. The surface created by Butterfly subdivision interpolates the control mesh. Fig. 3 shows the refinement operation in Loop and Butterfly subdivision. At each refinement step, each face is split into four new faces.

A. Loop Subdivision

Fig. 4 shows three levels of Loop subdivision. Starting from the input mesh M^0, mesh M^{i+1} is obtained from splitting the faces of mesh M^i and repositioning the resulting vertices. Mesh M^i is said to be at subdivision level or depth i. The resolution of the mesh and the subdivision depth of faces are directly related. As the subdivision depth increases, faces become finer and the resolution of the mesh increases. The geometric operators that define the position of the new vertices are represented by masks. Existing vertices v^i of mesh M^i are repositioned as a linear combination of their neighbours v^i_j.

$$v^{i+1} = \beta v^i + \alpha \sum_{j=0}^{n-1} v^i_j \, , \tag{1}$$

where n is the valence of vertex v^i and

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

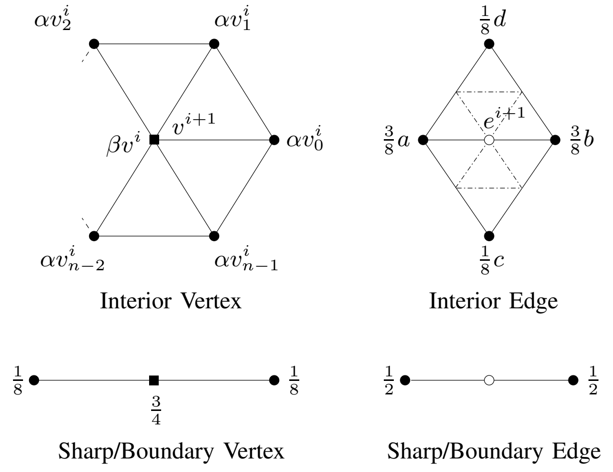


Fig. 5. Even and odd vertices in Loop subdivision. ● denotes existing vertex of mesh M^i, ■ denotes even vertex of mesh M^{i+1}, and ○ denotes odd vertex of mesh M^{i+1}. Subdivision masks of sharp or boundary edges are also shown.

and

$$\beta = 1 - n\alpha\,. \tag{3}$$

The vertices v^{i+1} are called *even* vertices of mesh M^{i+1}. As an example, in a regular mesh, where all vertices have valence six, $\alpha = \frac{1}{16}$ and $\beta = \frac{5}{8}$. The splitting operation introduces *odd* vertices e^{i+1} on the edges

$$e^{i+1} = \frac{3}{8}a + \frac{3}{8}b + \frac{1}{8}c + \frac{1}{8}d \tag{4}$$

where a, b, c, d are defined as in Fig. 5.

To model piecewise smooth surfaces, the subdivision rules are modified to break tangent continuity along the surface [8], [9]. A crease is a tangent line smooth curve along which the surface is $\mathbf{C}^0$. Edges along the crease are tagged as *sharp* and subdivided using the modified subdivision rules. Boundary edges are sharp edges with only one incident face. Fig. 5 shows the masks of odd and even vertices on sharp and boundary edges. A corner is a vertex where three or more sharp edges intersect. Corners are not repositioned during subdivision.

We can conclude from (1) and (4) that in order to correctly compute odd and even vertices of mesh M^{i+1}, all vertices of mesh M^i must be at the same subdivision depth. Since in adaptive subdivision only some faces are refined, adjacent vertices of M^i may have different subdivision depths. As a result, care must be taken to choose proper neighbours when computing the new vertex positions.

Loop subdivision surfaces are C^2 almost everywhere except at extraordinary vertices where they are G^1. In a triangle mesh, vertices with valence six are ordinary, otherwise they are extraordinary. In Loop subdivision, odd vertices are always ordinary while even vertices keep their valence. From the Euler-Poincaré formula it can be deduced that in Loop subdivision most vertices are ordinary, so the limit surface is C^2 almost everywhere. In Section III, we show that triangulating faces containing cracks changes the connectivity of odd and even vertices. This modifies the shape of the limit surface.

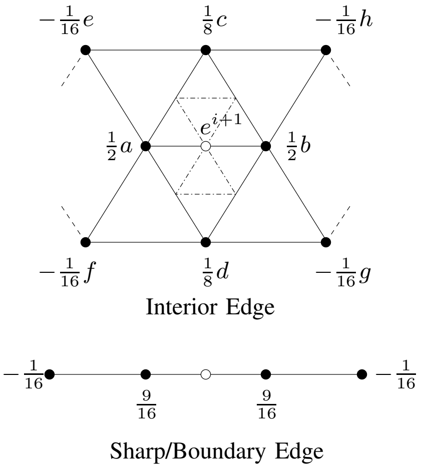


Fig. 6. Mask of odd vertex in Butterfly subdivision. ● denotes existing vertex of mesh M^i, and ○ denotes odd vertex of mesh M^{i+1}. Vertices a and b are ordinary vertices.

B. Butterfly Subdivision

Butterfly subdivision was first proposed by Dyn, Levin and Gregory [5]. It is an interpolating scheme, meaning that the even vertices are not repositioned during subdivision. Fig. 6 depicts the masks of odd vertices in Butterfly subdivision. In the regular case where a and b are ordinary vertices, the odd vertex e^{i+1} on edge ab is determined by

$$\begin{aligned} e^{i+1} \;=\;& \frac{1}{2}a + \frac{1}{2}b + \frac{1}{8}c + \frac{1}{8}d \\ & - \frac{1}{16}e - \frac{1}{16}f - \frac{1}{16}g - \frac{1}{16}h\,, \end{aligned} \tag{5}$$

Again, for correct results, all vertices involved in computing e^{i+1} must be at the same subdivision depth.

Subdivision surfaces produced by the original Butterfly scheme are C^0 at extraordinary vertices and C^1 away from them. The subdivision mask for odd vertex e^{i+1} was later modified so that the surface is also C^1 at the extraordinary vertices [10]. The position of odd vertex e^{i+1} on edge $v^i v_j^i$ depends on whether one or both edge end-points v^i and v_j^i are extraordinary. In the case where v^i is extraordinary and v_j^i is ordinary, the mask of e^{i+1} is determined according to the valence n of v^i. For valence three

$$e^{i+1} = \frac{9}{12}v^i + \frac{5}{12}v_j^i - \frac{1}{12}v_{j+1}^i - \frac{1}{12}v_{j-1}^i\,, \tag{6}$$

for valence four

$$e^{i+1} = \frac{6}{8}v^i + \frac{3}{8}v_j^i - \frac{1}{8}v_{j+2}^i\,, \tag{7}$$

and for all other valences of extraordinary vertex v^i

$$e^{i+1} = \beta v^i + \sum_{j=0}^{n-1} \alpha_j v_j^i\,, \tag{8}$$

where

$$\alpha_j = \frac{1}{n}\left(\frac{1}{4} + cos\frac{2j\pi}{n} + \frac{1}{2}cos\frac{4j\pi}{n}\right), \tag{9}$$

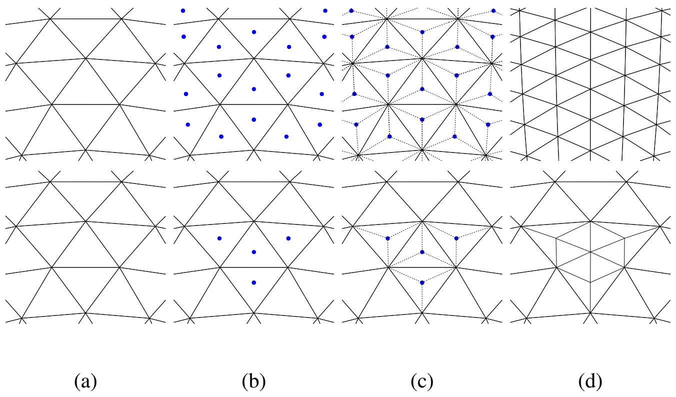


Fig. 7. $\sqrt{3}$-subdivision: starting from the coarse mesh (a), new vertices are inserted at the centroid of the faces (b), and connected to the vertices of the mesh (c). An edge flip completes the refinement (d). The bottom row shows adaptive $\sqrt{3}$-subdivision. The mesh is conforming even when a subset of the faces are refined.

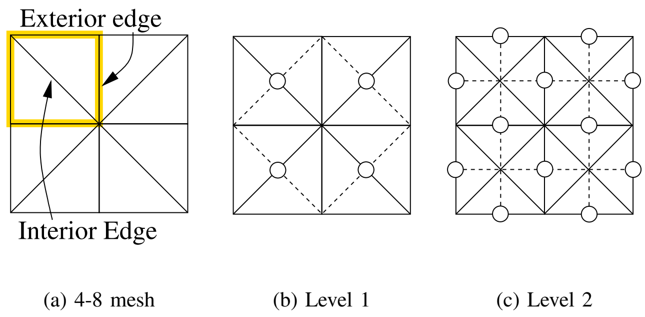


Fig. 8. 4-8 Subdivision. Vertices are inserted on interior edges at each subdivision step.

and

$$\beta = 1 - \sum_{j=0}^{n-1} \alpha_j \, . \tag{10}$$

If both vertices v_i and v_j^i are extraordinary, then the above is repeated for each vertex and the result is averaged.

C. Other Subdivisions

Some subdivision algorithms are designed such that their adaptive refinement is a natural extension of the subdivision process. The $\sqrt{3}$-subdivision by Kobbelt [7] and the 4-8 subdivision by Luiz Velho and Denis Zorin [11] are such schemes.

Shown in Fig. 7, the $\sqrt{3}$-subdivision inserts new vertices at the centroid of the faces. The new vertices are then connected to the existing vertices of the mesh within the faces. The refinement concludes with flipping all the edges of the resulting mesh. The existing vertices of the mesh are repositioned by a linear combination of their neighbours, which ensures a smooth surface at the limit [7].

Because the new vertices are inserted on faces, the mesh is conforming when it is adaptively subdivided.. The bottom row of Fig. 7 demonstrates one level of adaptive $\sqrt{3}$-subdivision. When a single face is refined, its adjacent neighbours are included in the subdivision.

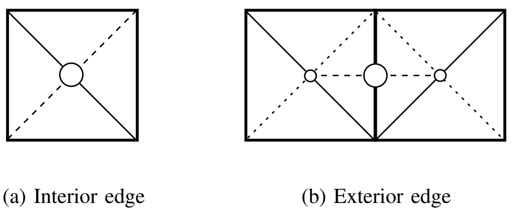


Fig. 9. Adaptive 4-8 Subdivision. A new vertex is inserted on an interior edge on the left mesh. It bisects the pair of faces incident to the edge. If the new vertex is inserted on the exterior edge, the two faces adjacent to it need to be refined first as in the right mesh.

Similarly, the adaptive refinement of 4-8 subdivision does not require post-processing to remove cracks. The 4-8 subdivision operates on 4-8 triangle meshes, shown in Fig. 8, that consist of vertices with valences four and eight. A pair of triangles, forming a square block divided along one of its diagonals, is called a basic block. An interior edge is the common edge to the two triangles forming a basic block. All other edges are exterior edges. A new vertex is inserted on the interior edge at the barycenter of the basic block. The vertex is connected to the vertices of the basic block, bisecting its faces. The new position of an existing vertex is computed as an average of its old position and barycenter of the vertices sharing an exterior block edge.

In adaptive 4-8 subdivision, the mesh is always conforming because both faces incident to the interior edges are bisected. If a new vertex is inserted on an exterior edge, then an extra step of subdivision is required. Fig. 9 shows both cases of inserting a new vertex on interior and exterior edges.

Our incremental approach can be applied to the above triangle subdivisions, and it can produce better triangle distributions. However, since they are not commonly used in practical applications and because they have a reasonable adaptive scheme, we do not see a strong motivation to discuss the incremental approach for them.

III. Adaptive Subdivision

In this section we review the adaptive subdivision schemes that have the 1-to-4 triangle split refinement. First, some methods of determining which regions of the mesh to subdivide are discussed. Then, a simple algorithm for removing cracks is introduced. This method has two drawbacks that can be avoided with more complicated algorithms, which we discuss here.

A. Selection Criteria

In adaptive subdivision, one must decide on which areas of the control mesh to subdivide. This decision is either determined by the application or is specified directly by the user.

To render or visualize subdivision surfaces, the control mesh is subdivided until it is a sufficiently good approximation of the limit surface. Subdividing the entire mesh grows the number of faces exponentially. To reduce the complexity of the model in real-time applications, an approximation of the limit surface

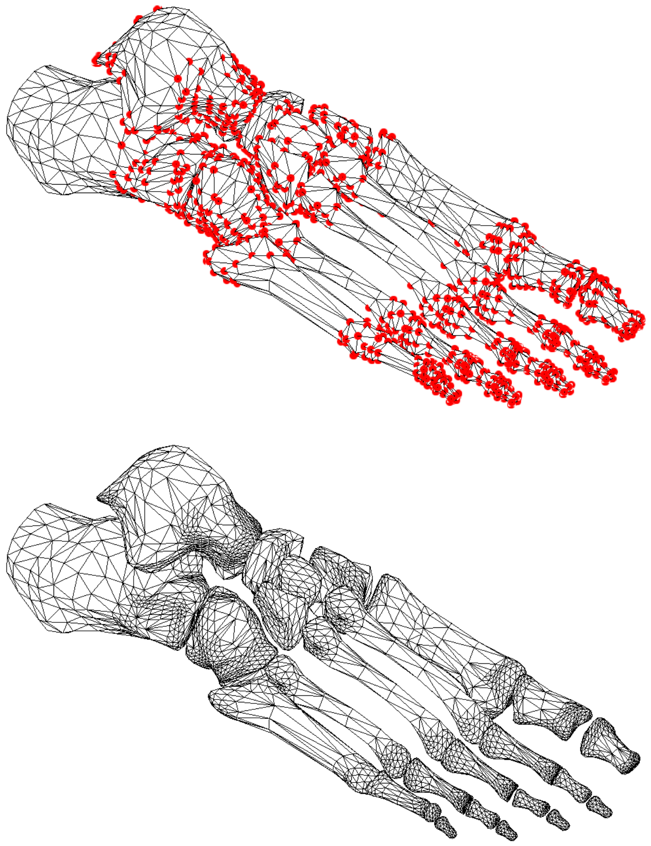


Fig. 10. Adaptive subdivision based on Gaussian curvature. Vertices with computed high curvature are selected for refinement.

is computed by adaptively subdividing the mesh. There are a number of methods for computing an approximation of the limit surface. One is to calculate the error of the mesh by comparing it to its limit surface. In Loop subdivision, the position of vertex v^i at the limit is

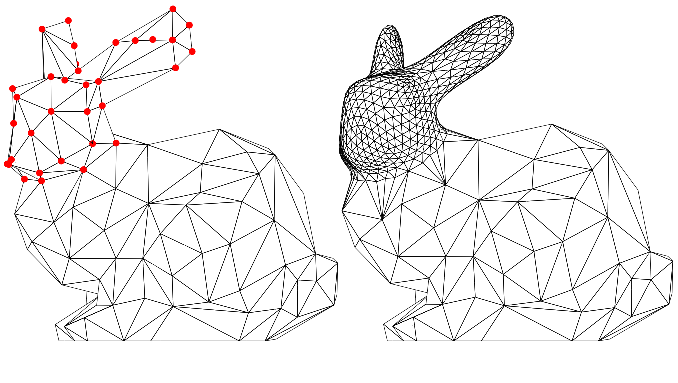
$$v^{i\to\infty} = \chi v^i + \alpha \sum_{j=0}^{n-1} v^i_j\,, \tag{11}$$

where

$$\chi = \frac{1}{\frac{3}{8}\beta + n}\,, \tag{12}$$

and α and β are defined in (2) and (3)[4]. If the approximation error $\epsilon^i = \left|v^i - v^{i\to\infty}\right|$ is larger than a predefined threshold, then the vertex v^i and its neighbours v^i_j are selected for adaptive subdivision.

Another criterion for deciding which areas to subdivide is the curvature of the surface. Higher curvature regions of the mesh M^i require more refinement than flat areas. Generally, high curvature areas contain more details. The local curvature of the mesh at vertex v^i can be computed using discrete Gaussian curvature analysis [12]. Fig. 10 shows adaptive subdivision of a control mesh based on discrete Gaussian curvature. Dihedral angle, the angle between the normals of adjacent faces, is also used as a simple approximation of surface curvature.



(a) User selected region (b) Adaptive subdivision

Fig. 11. Adaptive subdivision of user selected area

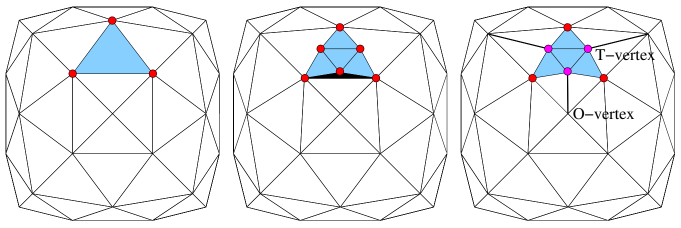
In real-time applications, other factors such as visibility, distance to the viewer, and the pixel area of faces are used to tune the selection algorithm. Isenberg, Hartmann and Knig present a general framework for the selection algorithm [13]. The adaptive subdivision area is defined as a set of faces that satisfy a Degree of Interest (DoI) function which may or may not be based on the geometric properties of the model. For example, to generate smooth silhouettes, the DoI can be set to take the normal of each face and the view vector into consideration, and subdivide all the faces that share edges on the silhouette boundary. In non-photorealistic rendering methods that are based on edge size [14], the selection area can be determined by setting the DoI to the size of the coarse edges.

In modeling applications, users may need control over the level of detail of the model. Artists may want to emphasize part of a scene by increasing the detail of that area. Adding features to the mesh generally requires an increase in the level of detail where the features are being added. In these cases the user selects areas of the mesh that are then subdivided. Fig. 11 shows adaptive Loop subdivision of a user defined area.

The target subdivision depth is either controlled by the user or by the application during run-time. In some adaptive subdivisions, the depth at which the vertices best approximate the limit surface are precomputed and then the mesh is subdivided to the preset depth [15]. In our case, the error in the mesh is computed at each step and the subdivision is stopped once this error reaches a threshold.

B. Handling Inconsistencies

Simple Triangulation: As mentioned earlier, subdividing a subset of the faces creates cracks on the mesh. These cracks are due to odd vertices on the edges shared by faces at different subdivision levels. The neighbourhood of the vertices is incomplete and when they are repositioned a crack is created. Amresh, Farin and Razdan propose a simple triangulation method that splits the neighbouring faces into two, three or four faces depending on the number of odd vertices [16]. As shown in Fig. 12, for each odd vertex, a bisection of the face with lower subdivision depth removes the crack. We call to



(a) Selected face (b) Cracks (c) Edge insertion

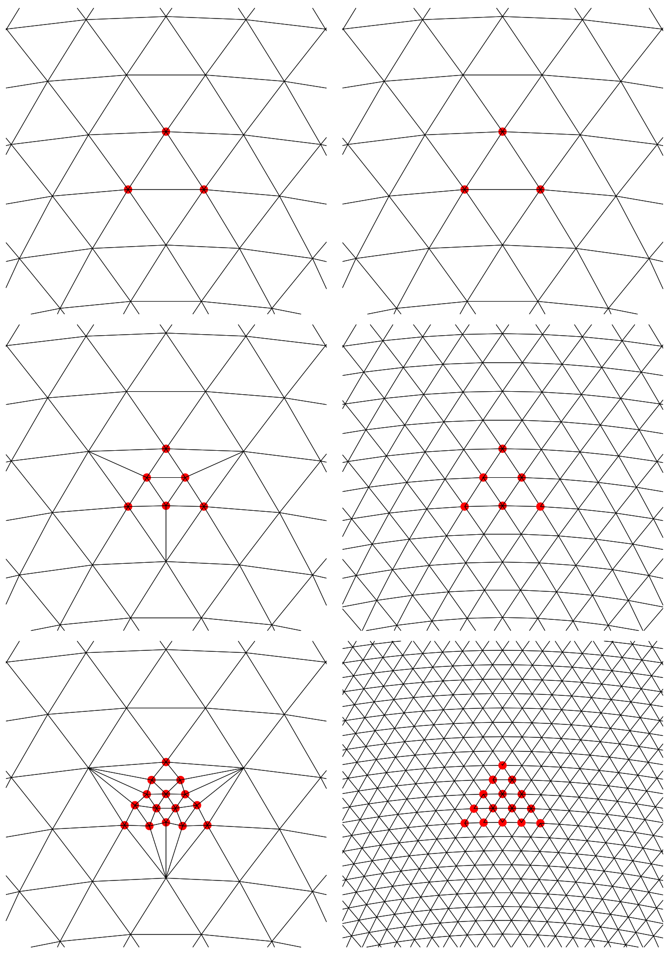
Fig. 12. Adaptive subdivision of one triangle and bisection of its neighbours to remove cracks

this odd vertex a *T-vertex*. The opposite vertex that the T-vertex connects to is an *O-vertex*.

While this simple triangulation method removes cracks efficiently, it has some undesired side-effects. First, it changes the connectivity and valence of odd vertices. This not only alters the limit subdivision surface, but also reduces its smoothness to G^1. Second, O-vertices have a different subdivision depth than the even vertices of the selected area. Therefore, as shown in Fig. 13, the even vertices are not repositioned properly if the selected area is subdivided again. Last, ignoring the irregular connectivity of T-vertices, repeated subdivision and simple triangulation of the selected area produces high valence O-vertices. High valence vertices lead to long faces which create ripple effects on the subdivision surface as show in Fig. 14.

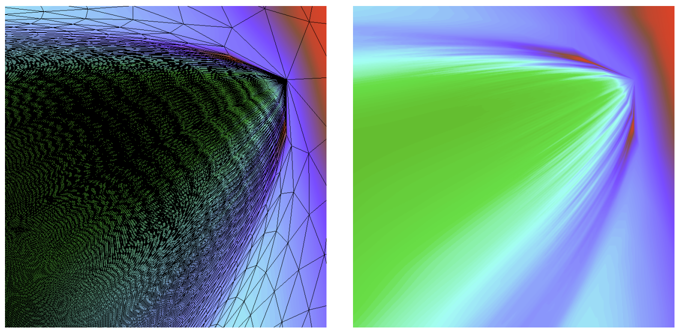
Red-Green Triangulation: Another method of removing cracks by inserting edges into the mesh is the red-green triangulation method of Bank, Sherman and Weiser [17]. This method was developed to allow adaptive refinement of meshes in finite element analysis. A mesh is used for approximating systems of equations in numerical analysis applications. When better approximations are required, the mesh is refined by splitting each edge to half and connecting the new vertices. This refinement operation is the same 1-to-4 refinement of Fig. 3. The exponential growth in the number of elements of the mesh leads to high computation load. Adaptive refinement is used to obtain good approximations while maintaining a reasonable number of faces. The adaptive subdivision leads to cracks and non-conforming mesh. If the mesh is non-conforming complex matrix assembly is required, so it is preferable to have a conforming mesh [6].

The red-green algorithm is as follows. Faces with one crack are bisected (green triangulation), and faces with more than one crack per edge are split into four (red triangulation). In other words, T-vertices, O-vertices, and the edges connecting them are temporary. Therefore, the connectivity of the selected region is unaffected. In addition, high valence extraordinary vertices are avoided. Fig. 15 shows red-green triangulation step by step. The consequence of red triangulations is that the subdivision depth difference of triangles incident to an edge is never larger than one. Therefore, the produced mesh is balanced, but these properties are not sufficient in the context of adaptive subdivision. In Loop subdivision, the new position of even vertices is computed as a weighted average of its neighbours as in (4), and all these vertices must be at the



(a) Adaptive (b) Regular

Fig. 13. Adaptive subdivision with simple triangulation on the left compared to subdividing the complete mesh on the right



(a) High valence vertex (b) Ripple effects

Fig. 14. Ripple effects appear on Loop subdivision surfaces due to high valence vertices

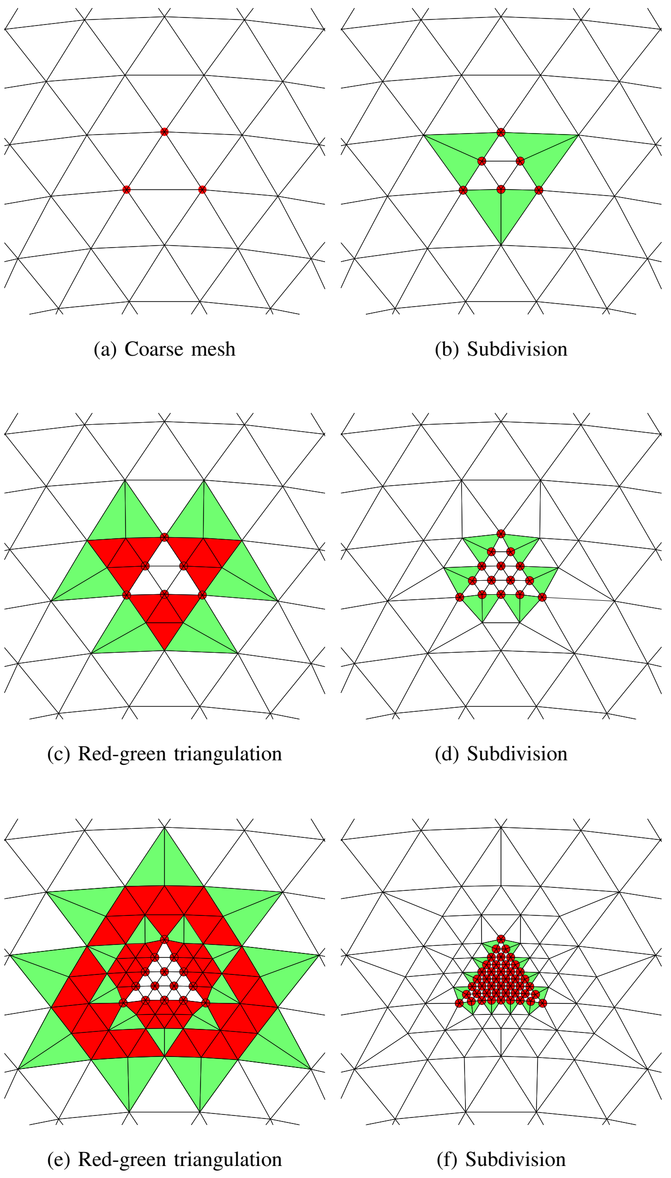


Fig. 15. Red-green triangulation: the green triangulation step removes cracks and the red triangulation refines faces with more than one crack per edge

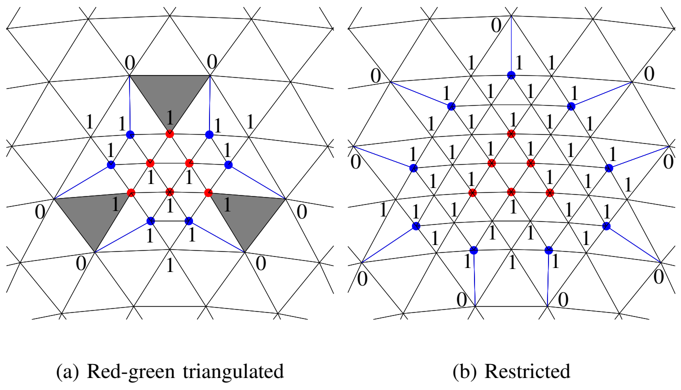


Fig. 16. Subdivision of highlighted faces to satisfy the mesh restriction criterion. The numbers represent subdivision depth of vertices.

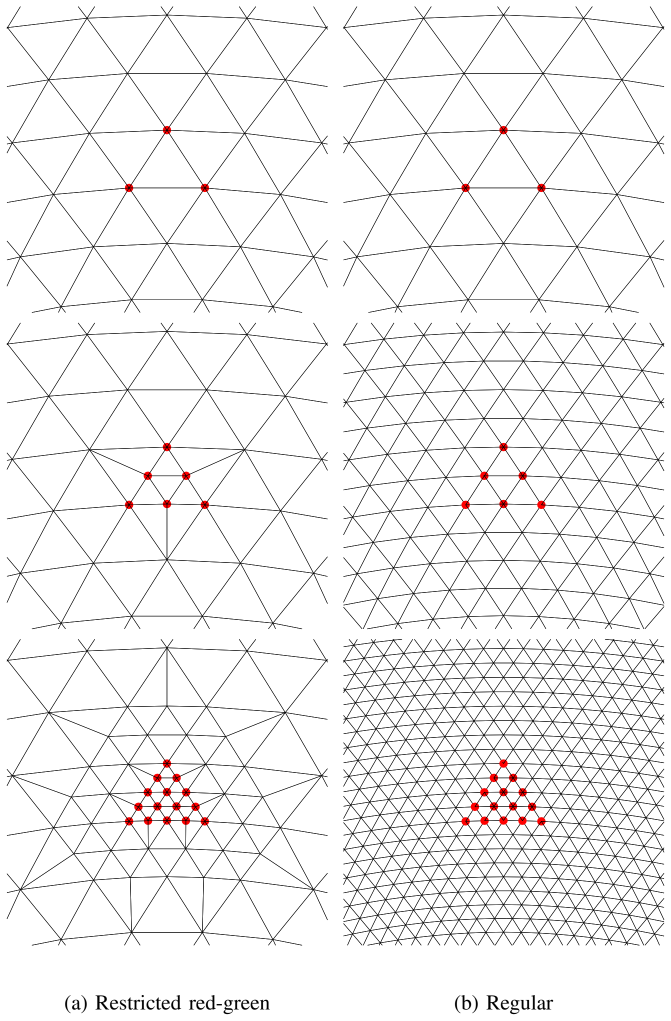


Fig. 17. Red-green triangulation with mesh restriction compared to when the entire mesh is subdivided

same subdivision depth. The even vertices have neighbours from different subdivision levels in red-green triangulation.

Restricted Mesh: As we discussed in Section II, to avoid any changes to the adaptive subdivision surface, odd and even vertices must be at the same subdivision depth as their neighbours. Zorin, Schröder and Sweldens call a mesh that satisfies this criterion a restricted mesh [18]. To obtain a restricted mesh, faces are constructed so vertices of the selected area and their neighbours have the same subdivision depth. Fig. 16 demonstrates a case where red-green triangulation is used to avoid cracks. While T-vertices are removed from the subdivision area, even vertices have neighbouring vertices from different subdivision depths. A hierarchical data structure is used for reconstruction of faces at different subdivision depths [18].

A Combined Algorithm: While red-green triangulation handles the connectivity issues of adaptive subdivision, mesh restriction addresses the correct computation of odd and even vertices. These two methods can be combined to obtain proper adaptive subdivision. Before subdividing the mesh, bisected faces are checked to see whether they need to be split into four. After red-green triangulation, the mesh is checked to satisfy the restriction criterion and if needed, some faces are subdivided (see Fig. 16). The selected area is subdivided and cracks are removed using green triangulation. Fig. 17 shows two levels of adaptive subdivision with red-green triangulation and mesh restriction.

This algorithm achieves the goals that we set in the introduction of this paper but it is not efficient nor simple. For each subdivision, three passes over the selected area must be made. One pass performs the red-green triangulation. New T-vertices may be created during this phase, therefore it must be recursively repeated until no new T-vertices are created. A second pass ensures mesh restriction, and finally the selected area is subdivided. If a large area is selected, this process becomes inefficient as some faces are bisected and then split into four. Incremental subdivision also achieves these goals but it is efficient and simple to implement.

IV. INCREMENTAL SUBDIVISION

A. Formal Description

We now describe our incremental method of adaptive subdivision for triangle meshes. This method efficiently generates, from the selected regions of the mesh, adaptive subdivision surfaces that are the same as when the entire mesh is subdivided. The resulting surface changes gradually in resolution from coarse to the fine areas.

Let $V = \{v_0, v_1, \ldots, v_{m-1}\}$ be the vertex set of the current mesh. Let S be a subset of V. We wish to adaptively subdivide S such that the limit surface generated from S is exactly the same as when V is subdivided. To do this, we expand the selected set S to a new larger set of vertices and then we subdivide this larger set.

More formally, at each subdivision level, expand S to $E^r(S)$ by including the vertices of V that are inside the r-ring neighbourhood of at least one vertex of S

$$E^r(S) = \bigcup_{v \in S} N^r(v), r > 0, \tag{13}$$

where $N^r(v)$ denotes the r-ring neighbourhood of v. Therefore, $w \in V$ is in $N^r(v)$ if and only if there is a path from v to w with maximum r edges. In graph theory terms, the distance of v and w must be smaller or equal than r. Fig. 18 illustrates $E^1(S)$ and $E^2(S)$ expansion of a selection region of the mesh. Next, subdivide $E^r(S)$ and use simple triangulation to remove cracks. Let $S^{'}$ be the new selected area that is the result of subdividing S. Fig. 19 illustrates two steps of incremental subdivision using $E^1(S)$ expansion. Fig. 20 shows incremental Loop and Butterfly subdivisions of a model.

Adaptive subdivision of $E^r(S)$ produces a limit surface from S that is exactly the same as when the entire mesh is subdivided. The reason for this is that T-vertices and O-vertices lie outside $E^r(S)$, so the connectivity of vertices within S remains unchanged. In addition, vertices of $S^{'}$ and

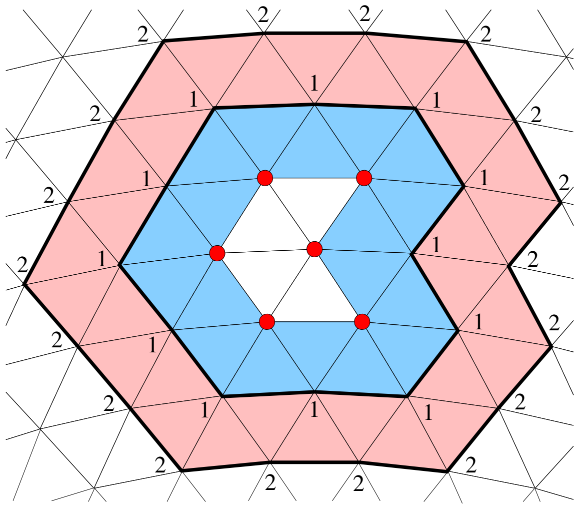


Fig. 18. One and two ring neighbours of selected vertices. Each number shows the distance of the vertex to the selected region.

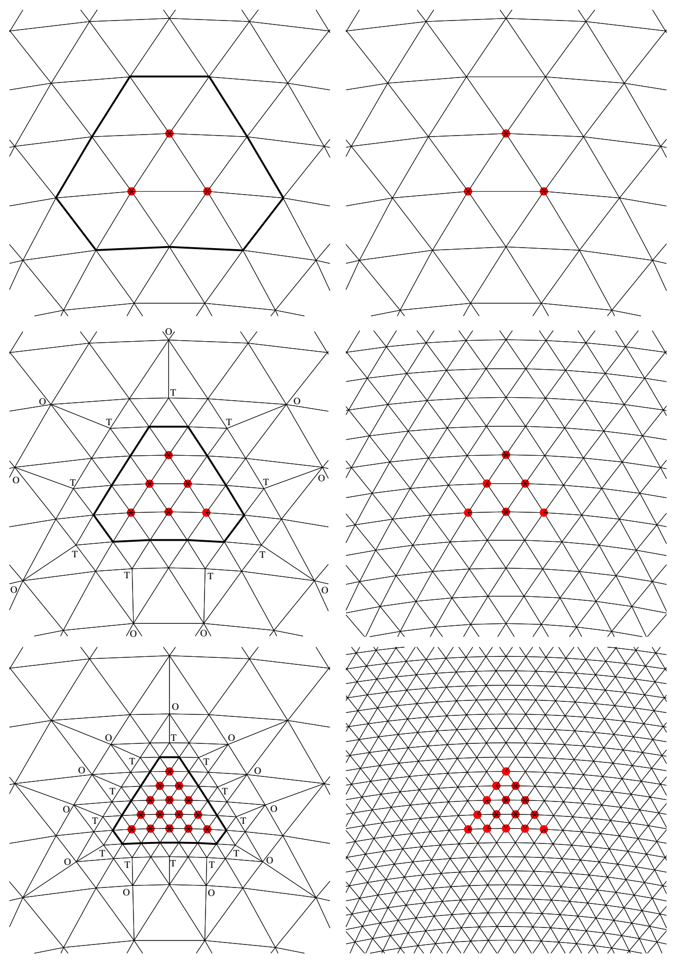
their neighbours are at the same subdivision depth because $E^r(S)$ includes the r-ring neighbours of S in the subdivision process. The limit surface of S is not changed due to the adaptive subdivision.

Incremental subdivision spreads O-vertices outside the selected subdivision area. Since odd vertices correspond to edges, an r-ring neighbourhood before subdivision corresponds to $2r$-ring neighbourhood after subdivision. T-vertices from adaptive subdivision of $E^r(S)$ are $2r$ units away from $S^{'}$. O-vertices are $3r$ units away. As shown in Fig. 19, O-vertices created from incremental subdivision of $S^{'}$ are different from O-vertices generated from incremental subdivision of S. This prevents the incremental subdivision algorithm from producing high valence O-vertices.

In addition, $3r$-ring neighbours of $S^{'}$ are always one level of subdivision lower than its $2r$-ring neighbours. The adaptively subdivided mesh is balanced, and a surface is created that gradually increases in subdivision depth from the coarse to the incrementally subdivided areas. Fig. 21 demonstrates this anti-aliasing effect.

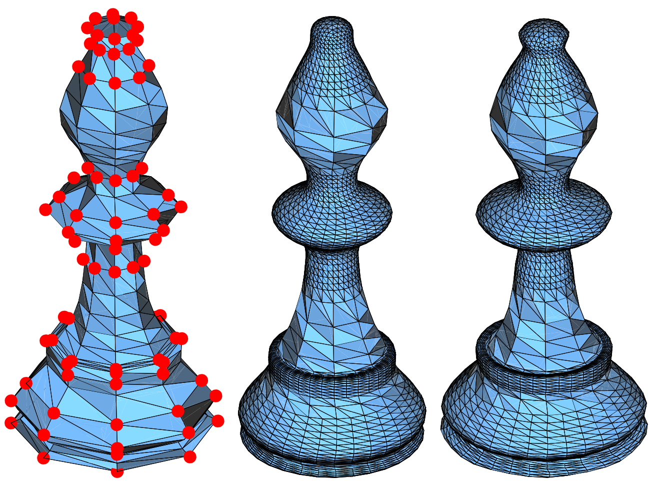
A larger r value includes a bigger neighbourhood of the selected area, and leads to a smoother transition from the coarse to the fine regions of the mesh. Fig. 22 demonstrates incremental subdivision with $r = 1$ and $r = 2$ expansions. The increased transition area from coarse to fine is beneficial for the rare occasion that a selected region of a coarse mesh is subdivided many times. In practice, a few refinements are sufficient to obtain a smooth surface, so $E^1(S)$ expansion is used in most cases.

The same $E^r(S)$ expansion for the interior of the mesh is be used to subdivide a selected region near or on the boundary of the mesh. The expansion includes r-ring neighbours of the selected region, but up to the boundary of the mesh. Therefore, the incremental method can easily be applied to open meshes. Fig. 23 illustrates incremental subdivision of a region on the boundary of the mesh.



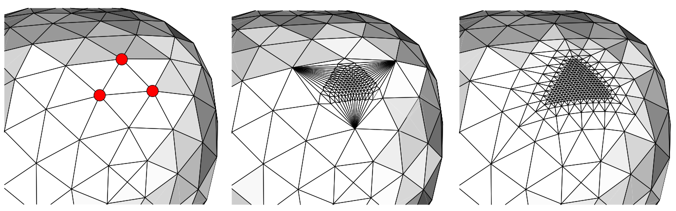
(a) Incremental (b) Regular

Fig. 19. Incremental subdivision on the left. The dots indicate selected vertices for subdivision. The thick edges indicates the boundary of $E^1(S)$, and the letters show T- and O-vertices.



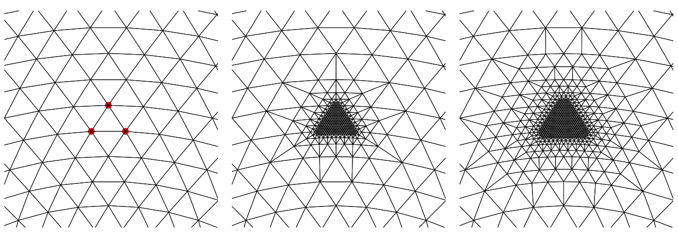
(a) Control mesh (b) Loop (c) Butterfly

Fig. 20. Incremental Loop and Butterfly subdivisions of a coarse model. The selected area is determined by surface curvature.



(a) Selected face (b) Adaptive (c) Incremental

Fig. 21. Smooth transition from coarse to fine areas of incrementally subdivided mesh



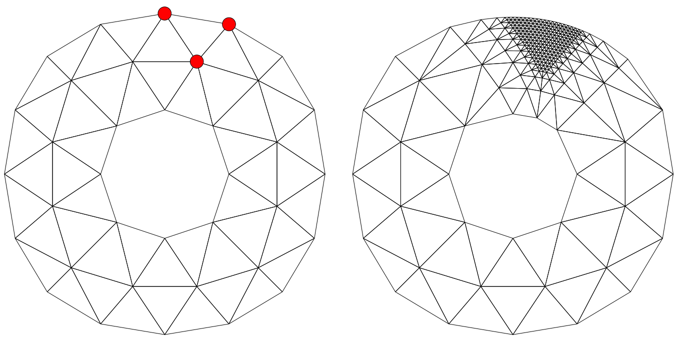
(a) Coarse mesh (b) $r = 1$ (c) $r = 2$

Fig. 22. Four levels of incremental subdivision with two different r-ring expansions

B. Comparison

We will now compare incremental subdivision to red-green triangulation with mesh restriction. Fig. 24 shows these two algorithms side by side. For $r = 1$, red-green triangulation and incremental subdivision are very similar. Red-green triangulation produces fewer faces by carefully triangulating faces containing cracks. In contrast, incremental subdivision always refines neighbouring faces of the selected area, and therefore creates more faces.

In practice, red-green triangulation is effective when small isolated areas of the mesh are selected for subdivision. In these cases the number of green and red triangulations are relatively small. Incremental subdivision is more efficient for large



(a) Selected area (b) Incremental subdivision

Fig. 23. Incremental subdivision of a region near the boundary of a strip mesh

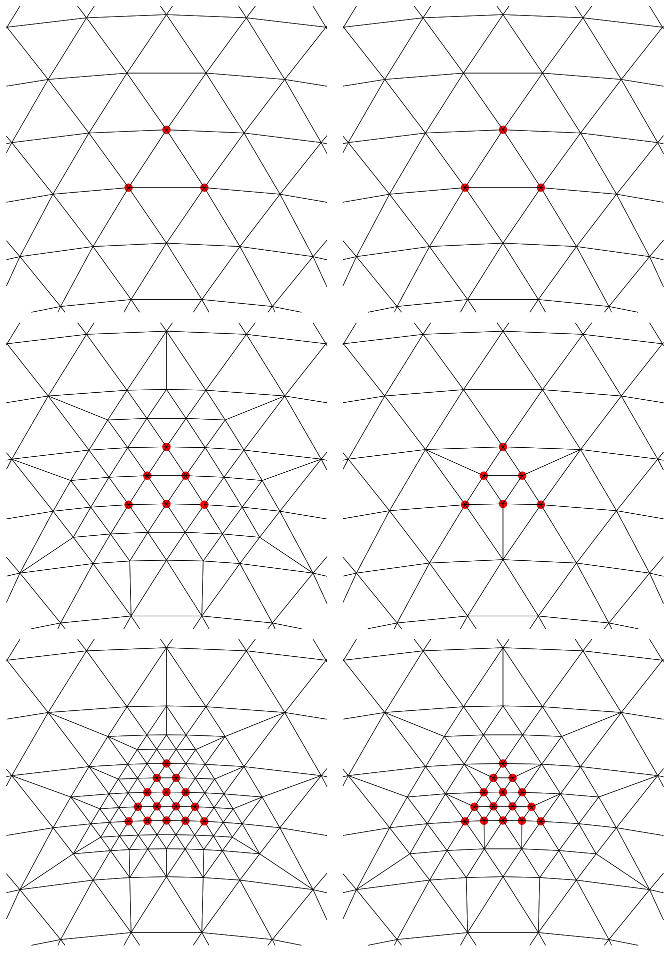


Fig. 24. Incremental subdivision on the left compared to red-green triangulation with mesh restriction on the right

randomly selected areas with overlaps. Adaptive subdivision with red-green triangulation and mesh restriction involves two additional steps compared to incremental subdivision. Before the subdivision step, green triangulated faces must be checked and red triangulated if it is determined that they will have more than one crack after the subdivision. Since a face split may create new T-vertices, this algorithm must recurse until no faces are bisected. Some faces may be triangulated multiple times after one subdivision. After red-green triangulation, the selected area must be analysed and processed to ensure it satisfies the restriction criterion. If faces have to be subdivided, red-green triangulation must be performed. In incremental subdivision, the expansion $E^r(S)$ can be performed quickly if neighbours of vertices can be directly accessed.

Red-green triangulation with mesh restriction also requires a complex data structure [7]. To reverse the green triangulations, the previous state of vertices and faces are needed, so it is implemented using a hierarchical data structure. The incremental subdivision can be implemented with a linear data structure that operates entirely within the graph space of the mesh.

Fig. 25 compares the four methods of subdividing a model: regular, simple adaptive, red-green triangulation with mesh restriction and incremental. Adaptive subdivision with simple triangulation is included as it is used in some cases, even tough it does not produce a correct limit surface. After two subdivision steps, incremental subdivision produces about 4% more faces than red-green triangulation with mesh restriction and about 7% more than simple triangulation. This small increase in the number of faces is outweighed by the gain in efficiency as well as simplicity of incremental subdivision.

To avoid extraordinary vertices when the mesh is adaptively subdivided, the adjacent faces to the subdivided area must also be subdivided [11]. By induction, the whole mesh would have to be subdivided. Therefore extraordinary vertices are unavoidable in adaptive subdivision. Extraordinary vertices affect the shape of the limit surface and reduce its smoothness. In the simple triangulation method, extraordinary vertices affect both the selected region and the area outside of it. Incremental subdivision pushes the extraordinary vertices outside the selected region. If the region outside the selected area is subdivided after incremental subdivision, its limit surface is changed due to the extraordinary vertices. Red-green triangulation avoids extraordinary vertices both inside and outside the adaptively subdivided region.

Both T-vertices and O-vertices are extraordinary vertices. T-vertices always have valence four, but need to be connected to an O-vertex to produce a conforming mesh. Therefore, they generally have valence five. In the simple triangulation method, the same O-vertices may complete the neighbourhood of new T-vertices. The number of extraordinary O-vertices is low, but they can have high valence, which is undesired. In comparison, O-vertices in incremental subdivision are spread outside the selected subdivision area and have low extraordinary valence. If it is required for the area outside the selected region to be unaffected by the adaptive process, then it should be selected for inclusion in the adaptive subdivision. The most common case, where the region outside the adaptively subdivided area is also subdivided, is when the entire mesh is refined after a few adaptive subdivisions. In these cases, we recommend subdividing the entire mesh first and then adaptively refining the regions of interest.

V. Results & Applications

A. Implementation

In order to implement incremental subdivision, we considered the half-edge data structure [19] and the vertex-vertex systems [20] as two powerful mesh representations. Both representations are suitable and efficient for implementation of subdivision algorithms. However, the most significant operation of incremental subdivision is forming the $E^r(S)$ expansion in (13). Efficient access to the neighbourhood of the selected vertices is crucial. The vertex-vertex systems, described below, allows direct access to the neighbourhood of each vertex. Therefore, in this work we used this mesh representation to implement subdivision.

The vertex-vertex systems is a data structure based on graph rotation systems. A graph rotation system associates each vertex of a polygon mesh with an oriented circular list of its neighbouring vertices. The collection of the vertices and their

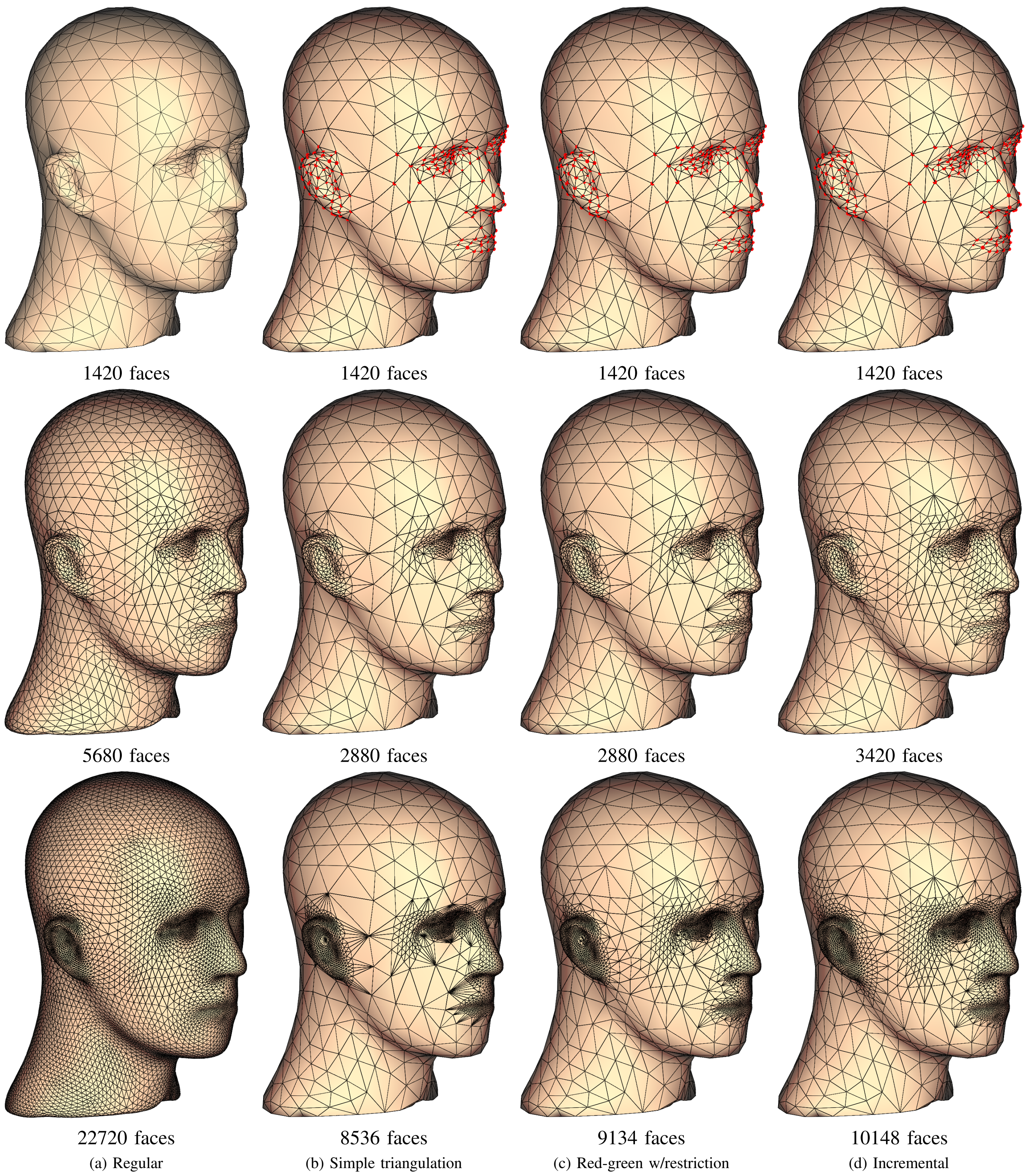
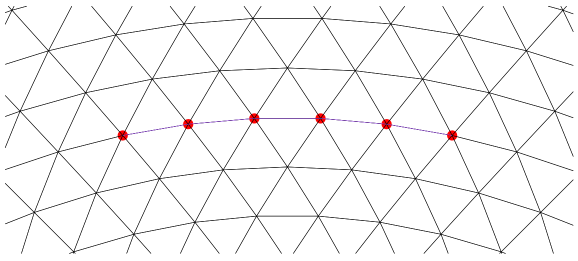
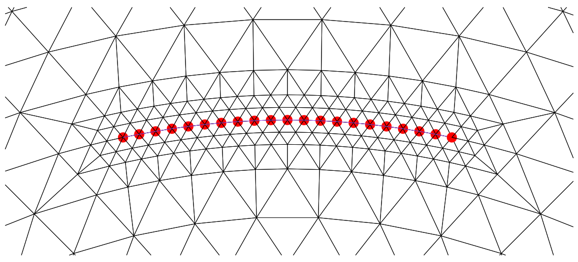


Fig. 25. Comparison of adaptive subdivision schemes. Incremental subdivision produces more faces than simple triangulation and red-green triangulation with mesh restriction, but it is more faster and more efficient.


(a) Selected edges


(b) Incremental subdivision

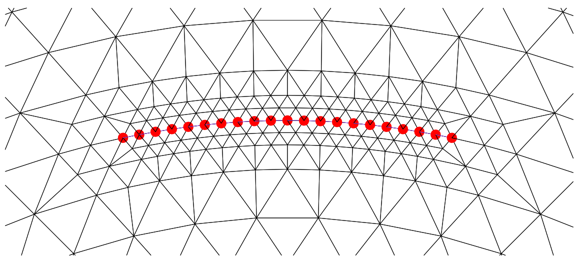
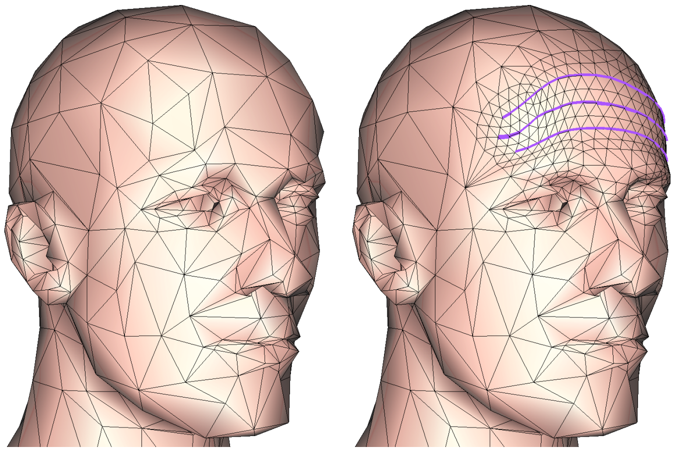

(c) Red-green triangulation with mesh restriction

Fig. 26. The selected edges are tagged as sharp and subdivided using incremental subdivision in the middle row as well as the red-green triangulation method with mesh restriction in the bottom row.


(a) Coarse mesh (b) Tagged edges

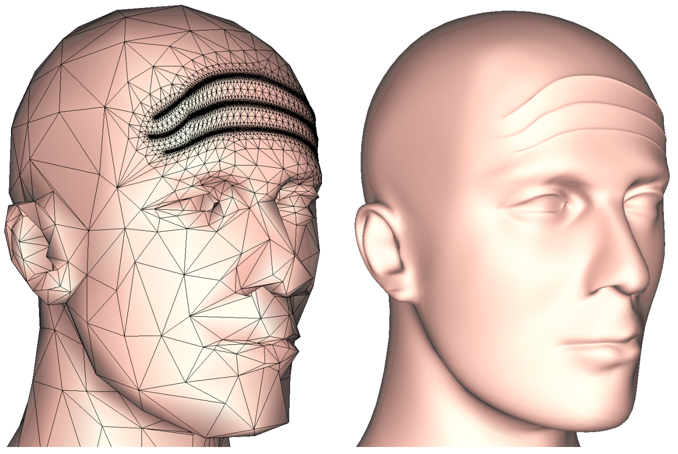

(c) Incremental subdivision (d) Fine mesh

Fig. 27. Creating wrinkles using incremental subdivision of sharp edges: the forehead of the head model (a) is first incrementally subdivided to allow more controlled editing (b). A series of edges are tagged as sharp and incrementally subdivided (c). The entire mesh is then subdivided to produce a high resolution model (d).

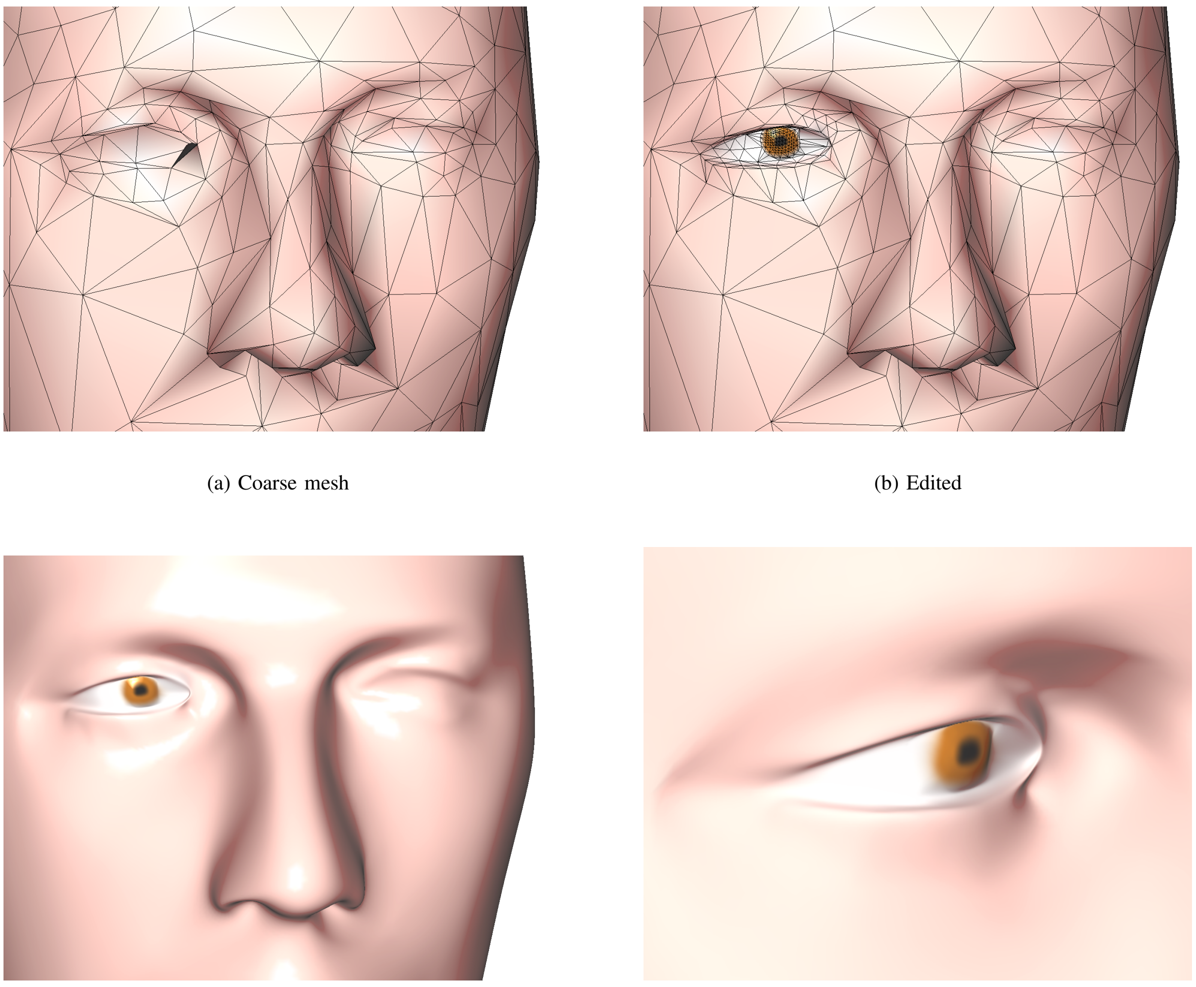
adjacency information completely represents the topology of a 2-manifold mesh.

In order to implement the 1-to-4 refinement of Loop and Butterfly subdivisions, an odd vertex e^{i+1} is inserted on each edge of the vertices of the mesh M^i if the edge is not already split . These edges are then connected to each other (see Fig. 5) to form mesh M^{i+1}. Note that in Loop subdivision, the vertices of mesh M^i must also be repositioned according to the Loop mask. Adaptive refinement is implemented similarly, but edges are split only if both of their vertices are selected. The incremental subdivision is implemented by first forming the $E^r(S)$ expansion. For $r = 1$, forming $E^r(S)$ is straightforward, but for $r > 1$, a depth search or breath search of r neighbours of the selected vertices is needed.

B. Applications

We have developed a number of applications using the incremental method. One application is creating high quality surfaces without exponentially increasing the number of faces. Figures 1, 10, 20, and 25 are examples of adaptive subdivision for efficient surface approximation.

Another application of incremental subdivision is adding features to a model. To add creases some edges are selected, tagged as sharp, and then subdivided. If the entire mesh is subdivided the model becomes too complex. Adaptive subdivision results in a mesh with a manageable number of faces. If the red-green triangulation approach is used, then the faces sharing these edges must be found and subdivided. During red-green triangulation some bisected faces may require regular refinement. Finally, the restriction criterion must be enforced. Incremental subdivision includes adjacent faces to crease edges with the expansion $E^r(S)$ and does not require further post-processing of the mesh. Allowing the user to mark edges provides an intuitive user interface for adding seams and creases to the model. The user selects and tags edges with the mouse or a tablet pen and the incremental algorithm handles the subdivision. As shown in Fig. 26, incremental subdivision produces a mesh almost identical to red-green triangulation with mesh restriction. Fig. 27 shows an example of adding sharp features to a model.



(a) Coarse mesh (b) Edited

(c) Fine mesh (d) Zoomed on the eye

Fig. 28. Adding eye details to the head model. The eye lids are tagged as sharp and incrementally subdivided. Then the pupils are added by incrementally subdividing a circle region inside the eye.

To add details to a coarse model, such as eye details to a human face, the artist can select regions of the mesh and adaptively subdivide them. Incremental subdivision allows the artist to gradually increase the resolution of the selected area. Fig. 28 shows an example of adding an eye to the head model used in the previous figures.

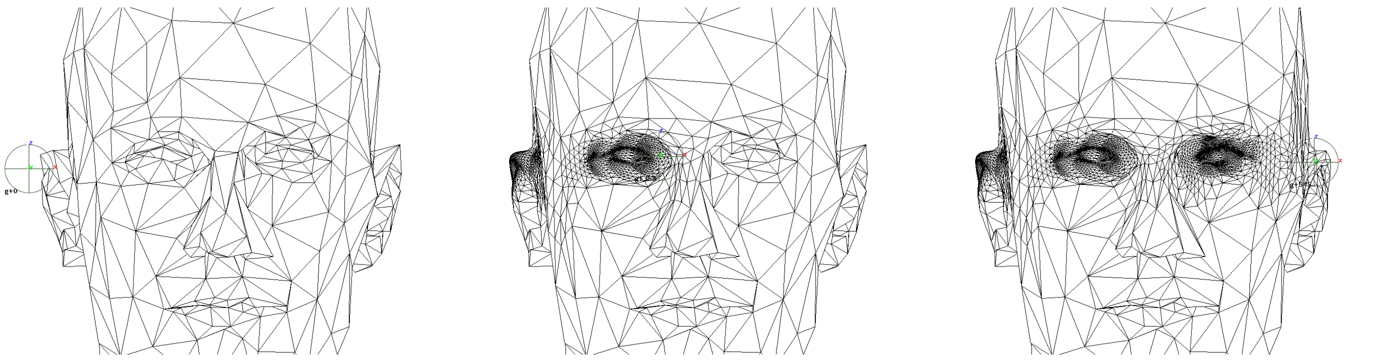
Fig. 29 shows three frames from subdivision using a pen that allows users to interactively refine the model by drawing on it. As the pen moves over the mesh, the faces are incrementally subdivided. The slower the pen moves, the more the area underneath it is refined to reflect more details on the surface.

VI. CONCLUSION

Adaptive subdivision allows us to create surfaces with different subdivision depths by subdividing selected areas of the input mesh. The simple triangulation method and red-green triangulation produce surfaces with undesirable properties. Using restricted meshes, it is possible to adjust the red-green triangulation method to obtain better behaved adaptive subdivision surfaces. However, this algorithm is not efficient and it is complicated for implementation. We introduced incremental adaptive subdivision for triangle meshes. It produces surfaces that have proper connectivity and geometry with a gradual change in subdivision depth between coarse and fine areas. Based on our comparison, incremental adaptive subdivision is more efficient than other methods while it is still simple to implement and can be effectively used in both modeling and rendering.

ACKNOWLEDGEMENTS

We are grateful to Peter MacMurchy and Colin Smith for their help and discussion on adaptive subdivision. In addition, we would like to thank John Brozs and Maria Perdomo. This research is partially funded by Natural Sciences and Engineering Research Council of Canada (NSERC).



(a) Frame 17 (b) Frame 35 (c) Frame 85

Fig. 29. Three frames from pen based real-time incremental subdivision. As the pen moves over the ears and eyes of the figure head, the faces are incrementally subdivided.

REFERENCES

[1] T. DeRose, M. Kass, and T. Truong, "Subdivision surfaces in character animation," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, vol. 32. ACM Press, 1998, pp. 85–94.

[2] D. Doo and M. Sabin, "Behavior of recursive subdivision surfaces near extraordinary points," *Computer-Aided Design*, vol. 10, no. 6, pp. 356–360, Sept. 1978.

[3] E. Catmull and J. Clark, "Recursively generated b-spline surfaces on arbitrary topological meshes," *Computer-Aided Design*, vol. 10, no. 6, pp. 350–355, Sept. 1978.

[4] C. Loop, "Smooth subdivision surfaces based on triangles," Master's thesis, University of Utah, Aug. 1987.

[5] N. Dyn, D. Levine, and J. A. Gregory, "A butterfly subdivision scheme for surface interpolation with tension control," *ACM Transactions on Graphics*, vol. 9, no. 2, pp. 160–169, 1990.

[6] M. Jones and P. Plassmann, "Parallel algorithms for adaptive mesh refinement," *SIAM Journal on Scientific Computing*, vol. 18, no. 3, pp. 686–708, 1997.

[7] L. Kobbelt, "$\sqrt{3}$-subdivision," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 103–112.

[8] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle, "Piecewise smooth surface reconstruction," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. ACM Press, 1994, pp. 295–302.

[9] H. Biermann, A. Levin, and D. Zorin, "Piecewise smooth subdivision surfaces with normal control," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 113–120.

[10] D. Zorin, P. Schröder, and W. Sweldens, "Interpolating subdivision for meshes with arbitrary topology," in *Computer Graphics Proceedings (SIGGRAPH96)*, ser. annual conference. ACM Press, 1996, pp. 189–192.

[11] L. Velho and D. Zorin, "4-8 subdivision," *Computer Aided Geometric Design*, vol. 18, no. 5, pp. 397–427, June 2001.

[12] M. Meyer, M. Desbrun, P. Schröder, and A. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," in *Visualization and Mathematics III*. Heidelberg: Springer-Verlag, 2003, pp. 35–57.

[13] T. Isenberg, K. Hartmann, and H. König, "Interest value driven adaptive subdivision," in *Simulation und Visualisierung*, T. Schulze, S. Schlechtweg, and V. Hinz, Eds. SCS European Publishing House, Mar. 2003.

[14] M. C. Sousa, K. Foster, B. Wyvill, and F. Samavati, "Precise ink drawing of 3d models," *Special issue of Computer Graphic Forum*, vol. 22, no. 3, pp. 369–379, 2003.

[15] K. Müller and S. Havemann, "Subdivision surface tesselation on the fly using a versatile mesh data structure," *Computer Graphics Forum*, vol. 19, no. 3, pp. 151–159, Sept. 2000.

[16] A. Amresh, G. Farin, and A. Razdan, "Adaptive subdivision schemes for triangular meshes," in *Hierarchical and Geometric Methods in Scientific Visualization*, G. Farin, H. Hagen, and B. Hamann, Eds., 2003, pp. 319–327.

[17] R. Bank, A. Sherman, and A. Weiser, "Refinement algorithms and data structures for regular local mesh refinement," in *Scientific Computing*, R. Stepleman, M. Carver, R. Peskin, W. F. Ames, and R. Vichnevetsky, Eds. IMACS/North-Holland, 1983, vol. 1, pp. 3–17.

[18] D. Zorin, P. Schröder, and W. Sweldens, "Interactive multiresolution mesh editing," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, vol. 31. ACM Press/Addison-Wesley Publishing Co., Aug. 1997, pp. 259–268.

[19] L. Kettner, "Designing a data structure for polyhedral surfaces," in *SCG '98: Proceedings of the 14th annual symposium on Computational geometry*. ACM Press, 1998, pp. 146–154.

[20] C. Smith, P. Prusinkiewicz, and F. Samavati, "Local specification of surface subdivision algorithms," in *Applications of Graph Transformations with Industrial Relevance*, ser. LNCS, J. Pfaltz, M. Nagl, and B. Bohlen, Eds., vol. 3062. Springer-Verlag, 2004, pp. 313–327.

Hamid-Reza Pakdel received the Honours Bachelor of Science in computer science from the University of British Columbia in April 2002. He is currently completing the Master of Science degree from the University of Calgary under the supervision of Dr. Faramarz F. Samavati. His thesis research is adaptive subdivision algorithms. His research interests include multiresolution modeling, sketch based modeling and image compression.

Faramarz F. Samavati is an Assistant Professor in the Department of Computer Science at the University of Calgary. He received his Ph.D. from Sharif University of Technology (Tehran, Iran) in 1999. His research interests are computer graphics, geometric modeling, scientific visualization and computer vision. His current research is focused on multiresolution and subdivision methods, sketch based modeling and non-photorealistic rendering.