A direct approach for subdivision surface fitting from a dense triangle mesh

Weiyin Ma, Xiaohu Ma, Shiu-Kit Tso, Zhigeng Pan

Abstract

This article presents a new and direct approach for fitting a subdivision surface from an irregular and dense triangle mesh of arbitrary topological type. All feature edges and feature vertices of the original mesh model are first identified. A topology- and feature-preserving mesh simplification algorithm is developed to further simplify the dense triangle mesh into a coarse mesh. A subdivision surface with exactly the same topological structure and sharp features as that of the simplified mesh is finally fitted from a subset of vertices of the original dense mesh. During the fitting process, both the position masks and subdivision rules are used for setting up the fitting equation. Some examples are provided to demonstrate the proposed approach.

Keywords: Subdivision surfaces; Loop subdivision; Topological modeling; Mesh simplification; Surface fitting

1. Introduction

In literature, considerable work has been done on fitting B-spline surfaces to 3D points. Most of the approaches deal with a surface of simple topological type. In case of complex topology, it is an extremely difficult task in handling continuity conditions among neighboring surfaces. The use of subdivision surfaces provides a promising alternative approach for modeling shapes with arbitrary topology.

The basic idea of subdivision is to define a smooth shape from a polyhedron (or polygonal mesh) by repeatedly and infinitely adding new vertices and edges according to certain subdivision rules. Subdivision techniques were first proposed for surface modeling by Catmull and Clark [1] and Doo and Sabin [2]. Today, many subdivision schemes have been developed for modeling and animation [3]. One can also find several approaches for fitting subdivision surfaces to range or scanning data, an issue to be addressed in this article.

Among others, Hoppe et al. [4] presented an approach for automatically fitting subdivision surfaces from a dense triangle mesh. The fitting criterion is defined such that the initial dense mesh will be most close to a linear approximation of the final fitted subdivision surface, which provides sufficient approximation for practical applications. Both smooth and sharp models can be handled. This approach produces high quality visual models. However, it may need extensive computing time due to the large amount of data to be processed and a procedure involved for mesh optimization.

The method of Suzuki et al. [5] for subdivision surface fitting starts from an interactively defined initial control mesh. For each of the control vertices, a corresponding limit position is obtained from the initial dense mesh. The final positions of the control vertices are then inversely solved following the relationship between the control vertices and the corresponding limit positions through an iterative local approximation method. The topological structure of the control mesh is further subdivided and a refined mesh of limit positions is obtained. A new subdivision surface is then fitted from the corresponding refined limit positions. The procedure is repeated until the number of control vertices of the fitted subdivision surface exceeds a user-defined number. With this approach, the fitting criterion is defined such that the resulting subdivision surface interpolates the corresponding limit positions of the current control mesh. Since only...
With the proposed fitting algorithm, the topological fitting approach is developed for subdivision surface fitting. A network of boundary curves is first interactively defined for topological modeling. A set of base surfaces is then defined from the topological model for sample data parameterization. A set of observation equations is obtained based on ordinary B-splines for regular surface patches and an evaluation scheme of Stam [7] for extraordinary surface patches. A Catmull–Clark subdivision surface is finally obtained through linear least squares fitting. The approach makes use of all known sample points and the fitting criterion ensures best fitting condition between the input data and the final fitted subdivision surface. Apart from the parameterization procedure, the fitting process alone is pretty fast.

All of the above fitting approaches are built upon schemes for limit surface query, either at discrete positions [4,5] or at arbitrary parameters [6]. In literature, one can also find a scheme interpolating a set of surface limit positions with fairness conditions [8] and a systematic study on various interpolation constraints [9]. Theoretically, all interpolation schemes can be extended as a fitting scheme if the number of known surface conditions, such as limit positions, normal vectors and other constraints, are more than that of the unknown control vertices of the subdivision surface. One can also find an approach for adaptively fitting a Catmull–Clark subdivision surface to a given shape through a fast local adaptation procedure [10]. The fitting process starts from an initial approximate generic model defined as a subdivision surface of the same type.

This article presents a new approach for subdivision surface fitting from a known dense triangle mesh. A new feature- and topology-preserving mesh simplification algorithm is developed for topological modeling and a direct fitting approach is developed for subdivision surface fitting.

- The developed mesh simplification algorithm falls in the class of vertex decimation algorithms [11]. However, a set of new cost functions is developed for both feature-preserving vertex decimation and local re-triangulation. The cost functions are developed based on a quadric error metric (QEM) often used for edge contraction algorithms [12,13]. All internal creases, boundary edges and corners are preserved during mesh simplification and later surface fitting.

- With the proposed fitting algorithm, the topological structure of the initial control mesh of the final fitted subdivision surface are exactly the same as that of the simplified mesh and the vertices of the simplified mesh are considered as the limit positions of the initial control mesh of the fitted subdivision surface. This imposes a special requirement to the mesh simplification algorithm and vertices of the final simplified mesh should be a subset of vertices of the original mesh, which is different from most of the simplification algorithms for graphics applications [11–13 and references therein].

- Different from the approach of [4,6], the limit position evaluation masks, i.e. functions of limit positions defined by a linear combination of neighboring control vertices, are directly used for setting up the fitting equations with the proposed fitting algorithm. Compared with the approach of Ref. [5], the proposed fitting algorithm makes use of limit positions of a refined control mesh, not just those of the initial control mesh, for setting up the fitting equations. Both limit position masks and subdivision rules are used for setting up the fitting equation.

- As a result of refinement, the number of fitting equations will be larger than that of the initial control vertices, i.e. the unknowns, and hence it results in a fitting solution to the given limit positions, but not interpolation as in the case of Ref. [5]. Similar to the counter part of B-spline surfaces, simple interpolation conditions often result in excessive undulation [8]. A fitting condition, if possible, is preferred as it usually produces a numerically stable fitting system and a better fitting surface for the same number of control vertices.

- To find the limit positions of the refined control mesh, the simplified mesh is refined using the same topological rules as those of the subdivision surface and the new positions after refinement are replaced by the corresponding nearest vertices found in the initial dense mesh. The refinement of the simplified mesh is thus a procedure to sample a fine mesh having the same topological structure as that of the initial control mesh from the initial dense surface mesh.

The fitting algorithm reported in this article can be applied to all stationary subdivision schemes. A modified Loop subdivision scheme [4,14] is used as an example. The remainder of this article is organized as follows. In Section 2, we first present the entire fitting procedure. A modified Loop subdivision scheme and the evaluation of limit positions [4,5] used for setting up the fitting equations are further summarized in Section 3. Details on mesh simplification and surface fitting are further developed in Sections 4 and 5, respectively. Some examples are provided in Section 6 for evaluating the proposed method using an own developed prototype fitting system followed by Section 7 with final concluding remarks.

2. Overview of the entire fitting procedure

Fig. 1 illustrates the major steps of the approach used in this article for subdivision surface fitting. As shown in Fig. 1, the input data is an existing dense triangle mesh. We denote the initial dense triangle mesh as \( M_0 \). All sharp features of the model are first identified and marked for later
processing. The marked initial dense mesh with feature and topological information is denoted as \( M_{AF}^0 \). The marked triangle mesh is further simplified to the required resolution while preserving both sharp features and topology. The mesh simplification algorithm used in our approach is a kind of mesh decimation based on vertex classification. It is a stepwise process to delete one vertex at each step. The final simplified triangle mesh is denoted as \( M_{AF}^N \). A subdivision surface is finally fitted from a subset of the vertices of the original dense mesh. The final surface is shown in Fig. 1 as output represented by its control mesh \( M_C^0 \).

With our approach, the control mesh \( M_C^0 \) of the final fitted subdivision surface has exactly the same topological structure as that of the simplified mesh \( M_{AF}^N \). To obtain the set of vertices used for surface fitting, the simplified mesh \( M_{AF}^N \) is first refined by a mid-point linear subdivision scheme. Each of the newly inserted vertices is further replaced by the nearest point found in the original dense mesh. The collection of the vertices in \( M_{AF}^N \) plus the newly inserted vertices form a subset of vertices of the original dense mesh for surface fitting. The number of subdivisions needed for sampling extra vertices is determined by the nature of the object and, as discussed at the end of Section 5.1, one refinement or two is usually sufficient for fitting purposes.

For the fitting procedure, the selected vertices are regarded as the limit positions of the subdivision surface to be fitted. Both position masks and subdivision rules are used for setting up the fitting equation. The final control mesh of the subdivision surface is solved using a fast iterative conjugate gradient method without any constraints and the computing time is pretty fast. With our fitting criteria, the limit surface \( S^w \) is most close to a subset of the initial dense mesh in the least squares sense.

3. The loop subdivision scheme

The Loop subdivision scheme is first developed in Ref. [14] for processing smooth models. In this article, we use an extended Loop subdivision scheme for handling boundary and sharp features [4]. Similar to other schemes based on triangle meshes, the Loop subdivision is based on an one-to-four splitting scheme for subdividing a triangle mesh [5]. During a subdivision procedure, each of the triangle faces is split into four smaller triangles and the number of the triangular faces after one subdivision will be four times the number before subdivision.

3.1. Edge and vertex classification

To clearly describe the subdivision rules, we use the following conventions throughout the article. Let \( M_C^0 \) be a triangle control mesh. The position of the \( i \)th vertex of \( M_C^0 \) is denoted by \( p_i \). Each of the vertices is connected to \( k \) neighboring vertices through edges. For an interior vertex, \( k \) is also the number of neighboring triangles of the \( i \)th vertex and is called the valence of the corresponding vertex. If the valence of an interior vertex is six and the surface is smooth in the neighborhood of that vertex, the vertex is called an ordinary vertex. A vertex whose valence is not six is called an extraordinary vertex. To represent models with sharp features, we use the following classifications proposed in Ref. [4]:

- **Edge classification.** The model under discussion may have one or more crease edges over the entire surface. Boundary edges of an open surface are regarded as sharp edges during the subdivision process. All other edges are called smooth edges. Sharp edges are tagged in the control mesh \( M_C^0 \).
- **Vertex classification.** Let \( s \) be the number of sharp edges meeting at a vertex. One can classify vertices into the following types according to the number of meeting crease edges \( s \) of the corresponding vertex.
  - A smooth vertex has no crease edges with \( s = 0 \).
  - A dart vertex is one where a crease edge terminates with \( s = 1 \).
  - A crease vertex is located on a sharp edge with \( s = 2 \).
  - A corner vertex has \( s \geq 3 \).

A boundary vertex with \( s = 2 \) can also be defined as a corner vertex if the boundary curve is \( C^0 \) and the surface goes through that vertex.

Crease vertices can also be further defined as regular and non-regular depending on the arrangement of smooth edges. An interior crease vertex is regular if it has valence six with exactly two smooth edges on each side of the crease. A boundary crease vertex is regular if it has valence four. All other crease vertices, whether interior or boundary, are non-regular. Fig. 2 shows some types of the vertices.

In addition, those newly inserted vertices after subdivision are called odd vertices, while the old vertices before subdivision are called even vertices. After subdivision, all the odd vertices have a valence of six and the valence of even vertices, including extraordinary vertices, remains
unchanged. In other words, extraordinary vertices remain extraordinary after subdivision.

3.2. Loop subdivision rules

After edge and vertex classification, one can apply the Loop subdivision rules for producing a refined mesh \([4,5,14]\). For a smooth vertex \(p\), the updated vertex \(p'\) is defined by the following equation:

\[
p' = (1 - k\beta)p + \beta(p_1 + p_2 + \cdots + p_k)
\]

where \(k\) is the valance of \(p\) and \(\{p_i\}_{i=1}^k\) are the immediate neighboring vertices of \(p\) before subdivision. The term \(\beta\) is defined by the following formula

\[
\beta = \begin{cases} 
\frac{3}{16} & (k = 3) \\
\frac{1}{2} \left( \frac{5}{6} - \left( \frac{3}{2} + \frac{1}{4} \cos \frac{2\pi}{k} \right)^2 \right) & (k > 3)
\end{cases}
\]

In case of ordinary vertex with \(k = 6\), Eq. (1) resumes to

\[
p' = \frac{5}{8}p + \frac{1}{16}(p_1 + p_2 + \cdots + p_6)
\]

For an odd vertex, the newly inserted vertex is defined as follows

\[
p' = \frac{1}{8}(3p_1 + 3p_2 + p_3 + p_4)
\]

where \(p_1\) and \(p_2\) are the two end vertices of the corresponding split edge, and \(p_3\) and \(p_4\) are those of the other two vertices of the triangles sharing this edge. Fig. 3 illustrates the subdivision masks for smooth vertices defined by Eq. (1) and smooth edge vertices by Eq. (4).

For vertices on sharp features or boundary edges, similar subdivision masks can be defined as shown in Fig. 4. In this figure, a double line indicates a crease or a boundary edge. As shown in Fig. 4(a) and (b), subdivision rules for crease edges are the same as that for boundary edges. Following the subdivision rules shown in Fig. 4, a corner vertex remains the same after subdivision and the inserted vertex on a crease edge or a boundary edge is the mid-point of the corresponding edge. A crease vertex or a boundary vertex \(p'\) is defined by

\[
p' = \frac{1}{8}p_1 + \frac{3}{4}p + \frac{1}{8}p_2
\]

where \(p\) is the vertex before updating and \(p_1\) and \(p_2\) are the two neighboring crease points of that vertex on the same crease edge.

In the present implementation, we mainly use the above rules for model reconstruction. We also use the same subdivision masks for both the regular and non-regular crease vertices. In literature, one could also find other modified rules for the Loop subdivision scheme \([4,15]\) that can be incorporated in the fitting procedure. In general, the reported fitting procedure applies to all stationary subdivision schemes (see Section 5).

By infinitely subdividing a triangle mesh, it converges to a limit surface. The Loop scheme is based on the three-directional box spline, and the original mesh before subdivision is the control mesh of this box spline surface. Away from crease edges, corners and darts, the limit surface of the Loop subdivision scheme is \(C^2\)-continuous except at extraordinary vertices, where \(C^1\)-continuity is obtained. It produces creases, corners and darts in the neighborhood of the corresponding sharp features of the initial mesh. A crease
edge is a tangent line smooth curve along which the surface is \( C^0 \) but not \( C^1 \).

### 3.3 Limit position evaluation

For later surface fitting, we also need to evaluate the limit position of a corresponding control point. For a given vertex \( p_i \) of the control mesh, the position of the vertex can be repeatedly updated as the subdivision process proceeds. At the limit, the vertex finally converges to a position on the limit surface. This limit position can be obtained through the analysis of the eigenvalues and eigenvectors of the subdivision matrix. For the Loop subdivision scheme, all the limit positions of various types of vertices discussed in the previous subsections can also be graphically represented by the so-called position masks. Fig. 5 illustrates the position masks used for the evaluation of the limit positions in our research. It should also be noted that, with our implementation for testing, we use the same position masks for both the regular and non-regular crease vertices.

Following the position mask of Fig. 5(a), one obtains the limit position of a smooth or a dart vertex by the following equation

\[
p_i^\infty = (1 - k\alpha)p_i + \alpha \sum_{j=1}^{k} p_j
\]

where \( \alpha \) is defined by

\[
\alpha = (\frac{3}{8\beta} + k)^{-1}
\]

Similarly, the limit position of a crease vertex is shown in Fig. 5(b) and is defined by

\[
p_i^\infty = \frac{1}{6}p_1 + \frac{2}{5}p_i + \frac{1}{6}p_2
\]

Following Fig. 5(c), the limit position of a corner vertex remains the same as that in the control vertex.

### 4. Mesh simplification

This section describes the method used for simplifying a dense triangle mesh into a coarse one while preserving both the sharp features and topology. The topological structure of the simplified mesh will be exactly used as that of the control mesh for later surface fitting. The simplified mesh will also be used for sampling the initial dense mesh for subdivision surface fitting. For clarity, we first introduce some definitions. The mesh simplification or decimation algorithm is then further discussed.

In general, the main objective of a decimation algorithm is to reduce the total number of triangles in a triangle mesh, while preserving as accurately as possible the mesh geometry and its important features \[11\]. In other words, a reduced mesh must meet two requirements. First, the decimated mesh must form a good geometric approximation to the original mesh. Second, the reduced mesh must preserve the original topology and local features of the mesh. In addition, the vertices of the decimated mesh should also be a subset of the original vertices for later surface reconstruction. No new vertices are created, but relatively unimportant vertices and the associated triangles are removed from the mesh.

The mesh simplification algorithm developed in this article works in the following way. Multiple passes are applied to all vertices in the mesh. During a single pass, each vertex is a candidate for possible removal. If the vertex meets the specified decimation criteria, the vertex is deleted and the local region incident to the vertex is re-triangulated. Before performing mesh simplification, we first evaluate the cost parameter for each of the vertices of the mesh. The vertices of the mesh are sorted and the mesh model is simplified by deleting the least important vertices one at a time until the termination criteria are satisfied. After deleting a vertex, the cost parameters of the affected vertices are updated and the model vertices are resorted. The termination criterion can be a percentage of the simplified mesh versus the original dense mesh. It can also be a maximum decimation value measured against the cost parameter controlling the allowable geometric error if a vertex is removed. The basic algorithm involves three key steps, i.e. feature detection and vertex classification, evaluation of the cost of a vertex if it is removed, and simplification through vertex removal and local re-triangulation. The mesh simplification algorithm preserves both sharp and local features and the topology of the mesh model.
It falls in the class of vertex decimation algorithms [11]. However, the cost function is based on a QEMs often used for edge contraction algorithms [4,12,13]. Our new local re-triangulation algorithms are also based on the QEMs [12,13].

4.1. Feature detection and vertex classification

The first step of the mesh simplification algorithm is to characterize the local geometry and topological structure of a given vertex. The classification closely follows the discussions of the previous section for mesh subdivision. This process determines whether a vertex is a potential candidate for removal, and if it is a candidate vertex, what criteria will be used.

If the angle between the face normal vectors of two adjacent triangles sharing the same edge is greater than a specified feature angle, the corresponding edge is then classified as a feature edge. A feature edge in the context of mesh simplification is the same as a crease edge or sharp edge used for model subdivision. A vertex of a mesh can be either a simple vertex or a boundary vertex.

- A simple vertex or interior vertex is surrounded by a closed cycle of triangles, and each edge that connects the vertex is shared by exactly two triangles.
- A vertex located on the boundary of a mesh, i.e. connecting a semi-cycle of triangles, is a boundary vertex.

A simple vertex can be further classified as a smooth, a dart, a crease or a corner vertex. These classifications are based on the local mesh geometry discussed in the previous section. The number of meeting feature edges at a smooth, a dart and a crease vertex is 0, 1 and 2, respectively. If three or more feature edges meet at a vertex, the corresponding vertex is classified as a corner vertex. For mesh simplification, we also use the term feature vertex. A feature vertex can be any of the following types of vertices, i.e. a dart, a crease, a corner, or a boundary vertex.

During the process of mesh simplification, dart and corner vertices are not removed from the mesh. All other vertices are candidates for possible removal.

4.2. Cost evaluation for vertex decimation

To evaluate whether a vertex deserves removal and to formulate a new triangulation if a vertex is removed, we develop a cost function based on the QEM [12,13]. For each vertex, we formulate an ordered loop of immediate neighboring vertices and triangles. The corresponding vertex and triangles forming the loop are further evaluated. The evaluation determines whether the vertex can be removed and how a new local triangulation can be formulated after vertex removal. In the following paragraphs, we first summarize the QEM followed by the definition of the cost function of a vertex for possible removal.

Let \( p \) be any point on the plane containing face \( f \), \( n \) be the face normal and \( d \) be a scalar representing the distance from the origin of the coordinate system to the plane. As shown in Fig. 6, the quadric measure \( Q^f(v) \) of a vertex \( v \) to a face \( f \) of the mesh is defined as the squared distance from vertex \( v \) perpendicularly to the plane containing face \( f \),

\[
Q^f(v) = D^2 = (n^T(v - p))^2 = (n^Tv - n^Tp)^2
\]

In Fig. 6, the equation \( n^Tv + d = 0 \) provides the definition of the plane containing face \( f \). Noting the fact that \( d = -n^Tp \), we thus have

\[
Q^f(v) = (n^Tv + d)^2 = v^T(m^Tv + 2dn^Tv + d^2).
\]

To set up the cost function, let adjacent_face(\( v \)) be a face set containing all triangles incident to vertex \( v \) and let adjacent_vertex(\( v \)) be a vertex set containing all vertices adjacent to vertex \( v \). We first define

\[
Q(v) = \sum_{f \in \text{adjacent}\_\text{face}(v)} (Q^f(v))
\]

where \( v_i \) are vertices adjacent to the vertex \( v \). The quantity \( Q(v) \) represents the sum of squared distances from vertex \( v_i \) to each of the faces in adjacent_face(\( v \)). We then define the cost function in the following cases.

- **Smooth vertices.** For a smooth vertex \( v \), the cost function \( Q(v) \) is defined by the following equation

\[
Q(v) = \min_{v_i \in \text{adjacent}\_\text{vertex}(v)} (Q(v_i))
\]

In other words, the cost function is defined as the achievable minimum decimation error if that vertex is removed.

- **Crease vertices.** For a crease vertex \( v \), the cost function \( Q(v) \) is defined by the following equation

\[
Q(v) = \min_{i \in \{a, b\}} (Q(v_i)),
\]

where \( a \) and \( b \) are neighboring vertices of \( v \) on the same crease edge.

- **Boundary vertices.** For boundary vertices of an open surface, the cost function is defined as the distance from...
the vertex to the line segment defined by its two neighboring vertices \( a \) and \( b \).

- **Darts and corners.** Darts and corners are never decimated and hence, no cost values are associated to these vertices.

In case of a boundary vertex for an open surface, it should be noted that the adjacent vertices and adjacent faces of \( v \) may not form a closed loop. Eqs. (11)–(13) are also used in collaboration with the simplification and local re-triangulation strategies to be discussed in Section 4.3.

### 4.3. Mesh decimation and local re-triangulation

When deleting a vertex in case of vertex decimation, the neighboring triangles are also deleted and the hole must be filled with a new local triangulation. Different from other approaches found in literature, we make use of the QEM for local re-triangulation after vertex decimation. As shown in Fig. 7, the following strategies are used for local re-triangulation:

- **Smooth vertices.** For a smooth vertex \( v \), let \( v_i \), for all \( i \), and \( a \in \{ v_i \}, \text{ for all } i \) be vertices adjacent to \( v \) and \( a \) be selected such that

\[
\sum_{f \in \text{adjacent face}(v)} (Q'(a)) = \text{Min}_{v_j \in \text{adjacent vertex}(v)} \left( \sum_{f \in \text{adjacent face}(v_j)} (Q'(v_j)) \right).
\]

- **Crease vertices.** For a crease vertex \( v \), we use a similar strategy as that of a smooth vertex for re-triangulation. Let \( a \) and \( b \) be two neighboring vertices on the same crease edge as that of vertex \( v \). If \( Q(a) \) is smaller than \( Q(b) \), vertex \( v \) is moved to vertex \( a \), i.e. the variation of the model should be as small as possible after vertex removal. Otherwise, vertex \( v \) is moved to vertex \( b \). Fig. 7(b) illustrates the local triangulation before and after a crease vertex decimation in case that \( v \) is moved to \( a \).

- **Boundary vertices.** As shown in Fig. 7(c), if the distance \( d \) is less than a user-specified threshold, the boundary vertex \( v \) can also be removed. Similar to a crease vertex, if \( Q(a) \) is smaller than \( Q(b) \), vertex \( v \) is moved to vertex \( a \). Otherwise, vertex \( v \) is moved to vertex \( b \). Fig. 7(c) illustrates the new triangulation after moving the vertex to \( a \) in the event that \( Q(a) \) is smaller than \( Q(b) \).

In case of equal \( Q(v_i) \) for all \( i \), some geometric rules are applied to ensure the quality of the local new triangulation. In addition, some feature-preserving mesh optimization criteria for improving the mesh quality are also applied to the simplified mesh before fitting a subdivision surface from the sample data. Some optimization criteria considered at the moment include edge swapping and a few other rules for simple vertex removal and vertex relocation. Some issues on mesh optimization can be found in Ref. [4]. Similar to the simplification algorithm, there is also a special requirement for our implementation that vertices of the optimized mesh should also be a subset of the original vertices from the dense mesh for later subdivision surface fitting.

### 5. Least square fitting

#### 5.1. Setup of fitting equations

In this section, we describe a method to fit a subdivision surface to the dense triangle mesh \( M_A^3 \). Let \( Q = \{ q_i \in \mathbb{R}^3, \text{ for } i = 1, 2, \ldots, n \} \), be the set of \( n \) vertices of the simplified triangle mesh \( M_{A^r}^3 \), or \( M_A^N \) for simplicity, of the original dense mesh \( M_A^3 \). We can produce a fine triangle mesh \( M_A^N \) by first subdividing the simplified mesh \( M_A^N \) using a mid-point interpolatory scheme, and further replacing each of the newly inserted vertices by the nearest vertex found in the original dense mesh \( M_A^3 \). This leads to \( M_A^N \subset M_A^3 \subset M_A^3 \) with \( M_A^N \) and \( M_A^3 \) having a similar topological structure. Let \( \hat{Q} = \{ \hat{q}_j \in \mathbb{R}^3, \text{ for } j = 1, 2, \ldots, m \} \), be the set of \( m \) vertices of the triangle mesh \( M_A^N \) after refinement. For simplicity, we also use \( \hat{Q} \) to denote the \( m \) vertices in matrix form \( \hat{Q}_{m \times 3} = [\hat{q}_1, \hat{q}_2, \ldots, \hat{q}_m]^T \).

Now let \( X = \{ x_i \in \mathbb{R}^3, \text{ for } i = 1, 2, \ldots, n \} \), be a set of \( n \) vertices of the initial control mesh of a Loop subdivision surface with exactly the same topological structure as that of
the simplified mesh $M_X$ after optimization, which is still noted as $M_X$ for simplicity. Let $X = \{x_j \in \mathbb{R}^3, \text{ for } j = 1, 2, ..., m\}$ be the set of $m$ vertices of the control mesh after one further refinement of $X$ using the Loop subdivision scheme. It should be noticed that the refined control mesh $X$ will have exactly the same topological structure as that of $\tilde{Q}$, the refined triangle mesh $M_X$. For simplicity, let $X = X_{Q^3} = [x_1, x_2, ..., x_m]^T$ and $\tilde{X} = \tilde{X}_{Q^3} = [\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_m]^T$ denote the control vertices of the initial control mesh and the refined control mesh, respectively, in matrix form.

For clarity, let $\tilde{X}^\infty = \tilde{X}_{Q^3}^\infty = \{\tilde{x}_j^m \in \mathbb{R}^3, \text{ for } j = 1, 2, ..., m\}$ be the corresponding limit positions of $\tilde{X}$. Both $\tilde{X}^\infty$ and $\tilde{X}$ also share exactly the same topological structure. Following the position masks of Fig. 5 discussed in Section 3, each limit position $\tilde{x}_j^m$ of $\tilde{X}^\infty$ can be written as a linear combination of the refined control vertices $\tilde{X}$, i.e.

$$\tilde{x}_j^m = \sum_{j=1}^{m} \alpha_{ij}\tilde{x}_i,$$

for $j = 1, 2, ..., m$. (15)

Eq. (15) can be further written in matrix form as

$$\tilde{X}^\infty = A_{\text{refined}}\tilde{X},$$

where $A_{\text{refined}}$ is a square matrix and, similarly, $\tilde{X}^\infty = \tilde{X}_{Q^3}^\infty = [\tilde{x}_1^m, \tilde{x}_2^m, ..., \tilde{x}_m^m]^T$, denotes the limit positions of the control vertices of $X$ in matrix form. On the other hand, we can also obtain the following equation following the Loop subdivision scheme

$$\tilde{X} = B_{\text{refined}}X,$$

where $B_{\text{refined}}$ is the Loop subdivision matrix with $m$ rows and $n$ columns that can be defined using the subdivision masks discussed in Section 3. By introducing Eq. (17) into Eq. (16), we obtain:

$$(A_{\text{refined}}B_{\text{refined}})X = \tilde{X}^\infty,$$

or

$$C_{\text{refined}}X = \tilde{X}^\infty,$$

where $C_{\text{refined}} = A_{\text{refined}}B_{\text{refined}}$ is the coefficient matrix with $m$ rows and $n$ columns. To find the best fitting subdivision surface, we try to match the corresponding vertices of $\tilde{X}^\infty$ and $\tilde{Q}$ by minimizing the following function

$$\min_{\tilde{X}} f(\tilde{X}) = \sum_{j=1}^{m} (\tilde{x}_j^\infty - \tilde{q}_j)^2 = (\tilde{X}^\infty - \tilde{Q})^T(\tilde{X}^\infty - \tilde{Q}),$$

or in matrix form after introducing Eq. (19) into Eq. (20) as

$$\min_{\tilde{X}} f(\tilde{X}) = (C_{\text{refined}}X - \tilde{Q})^T(C_{\text{refined}}X - \tilde{Q}).$$

This leads to the following linear least squares solution for the vertices of the initial control mesh

$$C_{\text{refined}}X = \tilde{Q},$$

(22)

The above linear system has $m$ equations/observations and $n$ unknown vertices with $m > n$. Theoretically, one could apply as many number of refinements as necessary in order to obtain enough observation equations for a stable solution. For practical applications, one refinement or two is usually sufficient for fitting purposes. As for each refinement, the number of vertices will increase by a factor of four for the Loop subdivision scheme, one or two refinements guarantees that the number of equations, called the degrees of freedom in Ref. [8], will be about four or sixteen times the number of unknowns, which ensures a stable solution when solving Eq. (22) in the least squares sense. For all the examples in this article, one single refinement is used.

5.2. Considerations for solving the linear system

For a piecewise smooth subdivision surface, the vertices of the initial control mesh $X$ can be classified into three types when solving Eq. (22), i.e. (a) corner vertices $X_c$, (b) boundary and crease vertices $X_b$, and (c) darts and smooth vertices $X_s$ with $X = X_c \cup X_b \cup X_s$. A boundary vertex can also be classified as a corner vertex in case of need. Following such classification, we can partition the entire system of Eq. (22) with unknowns $X$ into three subsystems with corner vertices $X_c$, boundary and crease vertices $X_b$, and darts and smooth vertices $X_s$, as unknowns, respectively.

For practical implementation, the partition is done as follows. Since corner positions are fixed during the subdivision process, all corner vertices $X_c$ can first be extracted from Eq. (22) as known parameters. Furthermore, the limit position of a crease vertex is only a combination of the corresponding crease vertex and its two adjacent vertices, which can be dart, crease or corner vertices. Apart from the darts, these control vertices are only affected by the limit positions of the corresponding crease edge and can also be solved independently from other control points following Eq. (8). During this stage, darts are treated as known parameters and are pre-defined as estimated vertices on the simplified mesh. They are updated in the last stage together with smooth vertices. The coefficient matrix for solving creases $X_c$ has the following feature. Each of the primary diagonal elements equals to $4/6$ and each of the secondary upper and lower diagonal elements equals to $1/6$. All remaining elements are zero.

When $X_c$ and $X_b$ are obtained, we can introduce these control vertices into Eq. (22). After simplification, we obtain the following equation for solving the remaining darts and smooth vertices

$$\tilde{C}_{\text{refined}}X_s = \tilde{Q},$$

(23)

where $r$ is the number of vertices of $\tilde{Q}$, a sub set of $\tilde{Q}$ excluding corner, boundary and crease vertices, and $s$ is the total number of smooth vertices and darts. Eq. (23) can be solved using a least squares fitting method. With the implementation discussed in the following section, Eq. (23) is solved using a fast iterative conjugate gradient method [16].
The primary reason for decomposing Eq. (22) is for improving the quality of the crease and boundary edges. If the crease and boundary vertices are mixed with other control points, there might be deviations between the final fitted crease and boundary edges and the corresponding edges in the dense control mesh. In addition, the computation time can also be reduced by partitioning the whole system into three smaller ones in case of large and complex models.

5.3. Other observations

As indicated in Ref. [8], simple interpolation using limit position evaluation masks may result in a singular system. With the proposed fitting method, however, both the linear system (Eq. (22)) and its three subsystems for solving corner vertices \( X_c \), boundary and crease vertices \( X_b \), and smooth vertices and darts \( X_s \) are non-singular due to the extra degrees of freedom provided. While the fairness of the fitted subdivision surface of Ref. [8] is obtained by explicitly minimizing membrane and thin-plate energies, the fairness of the fitted subdivision surface of the proposed fitting method is realized as a result of the extra regular limit positions used for surface fitting.

Since all stationary subdivision schemes have fixed subdivision rules, it is always possible to obtain limit position evaluation masks through eigenvalue decomposition of the subdivision matrix and convergence analysis similar to the discussions of Refs. [3,8]. As a result, although only the Loop subdivision scheme is discussed in this section, the fitting approach discussed above applies to all stationary subdivision schemes.

6. Results and discussions

This section provides some examples to demonstrate the proposed methodology. A prototype system has been developed for evaluating and testing the approach described in this article. The system is implemented on a Pentium III computer with 800 MHz CPU and 256 MB RAM under a Windows NT operating system. Figs. 8 and 9 illustrate some examples produced for fitting a subdivision surface from a dense triangle mesh. For all these examples, the models have been translated and uniformly scaled such that all coordinates of the mesh vertices fall in the interval \([-1.0,1.0]\) or, in other words, the maximum dimension of these models is two units.

Fig. 8 shows a fitting example of a typical mechanical part with sharp features, such as sharp edges, corners and darts. The entire surface is fitted as a single piecewise smooth subdivision surface. Fig. 8(a) and (e) show the original triangle mesh with 1698 vertices and 3392 triangles. Fig. 8(b) and (f) show the simplified triangle mesh after mesh optimization. Fig. 8(c) and (g) illustrate the control mesh of a fitted subdivision surface. Fig. 8(d) and (h) show the limit surface of the final fitted subdivision surface of the dense triangle mesh shown in Fig. 8(a) and (e). As shown in Fig. 8(a)–(d), all sharp features are fully preserved during the entire fitting procedure of mesh simplification, mesh optimization and subdivision surface fitting. In

![Fig. 8. Fitting a subdivision surface for a mechanical part with sharp features: (a) and (e) the initial dense mesh; (b) and (f) the simplified surface mesh; (c) and (g) control mesh of the fitted subdivision surface; (d) and (h) the fitted subdivision surface. In illustrations (a)–(d), shape edges are shown with thick lines, and corners and darts are represented by the center positions of the small spheres.](image)
Fig. 8(a)–(d), sharp edges are illustrated as thick black lines and feature vertices, such as corners and darts, are indicated by the center positions of the small spheres shown in these figures. The fitting conditions, such as the initial number of vertices, the number of vertices of the simplified mesh and the number of control points of the final fitted subdivision surface can be found in Table 1. The average, standard deviation, the maximum and the minimum fitting errors are also summarized in this table.

Fig. 9 also shows two other examples for fitting smooth subdivision surface models. Fig. 9(a)–(d) illustrate the fitting result of a smooth surface sheet and Fig. 9(e)–(h) show the fitting result of a human head model. Some fitting parameters and the final fitting errors are also summarized in Table 1.

For all the above examples, the total computation time for each model, including feature identification, mesh simplification and subdivision surface fitting, is within a few seconds. The computing speed for the entire fitting procedure is much faster than the method of Ref. [4] and should also be comparable to or faster than that of Ref. [5]. With the proposed method, the entire topological structure of the control mesh and all required limit positions are immediately available after mesh simplification and one further refinement for data sampling. Vertices of the control mesh are then solved from a linear system in the least

![Fig. 8](image)
![Fig. 9](image)

Table 1  
Technical details of the examples for subdivision surface fitting

<table>
<thead>
<tr>
<th>Model</th>
<th>No. of vertices/no. of triangles of the initial dense model</th>
<th>No. of vertices/no. of triangles of the simplified model and the control mesh</th>
<th>Fitting errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1: a mechanical part (Fig. 8)</td>
<td>1698/3392</td>
<td>173/342</td>
<td>Average error: 0.010113</td>
</tr>
<tr>
<td>Model 2: a surface sheet (Fig. 9(a)–(d))</td>
<td>3571/6928</td>
<td>282/496</td>
<td>Standard deviation: 0.007530</td>
</tr>
<tr>
<td>Model 3: a human head (Fig. 9(e)–(h))</td>
<td>6769/13472</td>
<td>369/719</td>
<td>Maximum error: 0.054177</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Minimum error: 0.000000</td>
</tr>
</tbody>
</table>
squares sense. In the fitting system, the number of sample points (limit positions) involved is just enough for the fitting solution and not all the vertices of the initial dense mesh are used in the fitting procedure. The long computing time of Ref. [4] is firstly due to an expensive procedure involved for mesh optimization and secondly due to the fact that the fitting procedure uses all vertices of the entire dense model. With the method of Ref. [5], on the other hand, both the topological structure and vertices of the control mesh are obtained adaptively through a number of iterations. Each of the iterations, however, involves a mesh refinement procedure and the solution of a linear system.

Similar to B-spline surface fitting, the accuracy of the final fitted subdivision surface depends on the number of vertices of the control mesh and its parameterization, i.e. where to place the control vertices. For the proposed method of this article, model accuracy can be indirectly controlled by controlling the error for mesh decimation. A similar approach can also be applied to the method of Ref. [4]. After mesh simplification, both the number of control vertices and their distribution are determined. For the method of Ref. [5], model accuracy may be controlled by checking the error from the dense mesh and the fitted subdivision surface, but this could be an expensive procedure. For model quality, both the proposed method and the method of Ref. [4] observe sharp features and topology of the mesh model. However, only smooth models can be processed with the reported implementation of Ref. [5]. For the reported implementation of Ref. [5], since the entire control mesh will be refined and the number of control vertices will increase exponentially for each refinement, the method may easily produce a model with excessive number of control vertices, or it cannot satisfy the accuracy requirement. This problem could however, be solved by using a local adaptive fitting procedure.

7. Conclusions

This article presents an approach for reconstructing a smooth or a piecewise smooth subdivision surface model from a dense triangle mesh. All sharp features are first identified from the initial dense mesh. A feature- and topology-preserving mesh simplification algorithm is developed to simplify the dense mesh into a coarse one. A subdivision surface is finally fitted from a subset of the initial vertices of the dense triangle mesh through least squares fitting. During the fitting procedure, both subdivision rules and position masks are used for setting up the fitting equation. The fitting problem is solved using a fast iterative conjugate gradient method [16] without any constraints. The topological structure of the control mesh of the final fitted subdivision surface is exactly the same as that of the simplified surface mesh. Although only the Loop subdivision scheme is discussed in this article, the fitting method applies to all stationary subdivision schemes.

On the other hand, the quality of the fitted subdivision surface is sensitive to the quality of the simplified mesh. At the moment, only edge swapping and a few other rules for simple vertex removal and vertex relocation are implemented for mesh optimization. Some further work needs to be conducted to optimize the simplified surface mesh. An adaptive approach may also be applied for handling local detailed surface features, such as the local features of the head model shown in Fig. 9(e)–(h). With the current approach, the main objective of mesh simplification is for topological modeling. Other approaches, such as automatic model partition, can also be used for topological modeling.

Acknowledgements

The authors wish to acknowledge the support from City University of Hong Kong through Strategic Research Grants #7001074 and #7001296. The fourth author also wishes to acknowledge the support from the Excellent Youth Foundation Grants of MOE and Zhejiang Province (RC40008) of China. Thanks are also extended to the owners of the original datasets of the three models shown in Figs. 8 and 9 for making the datasets available to the public through Internet.

References


Weiyin Ma is an Associate Professor in manufacturing engineering at the City University of Hong Kong (CityU). He lectures in the areas of geometric modeling, CAD/CAM and rapid prototyping. His present research interests include computer aided geometric design, CAD/CAM, virtual design and virtual manufacturing, rapid prototyping and reverse engineering. Prior to joining CityU, he worked as a research fellow at Materialise N.V., a rapid prototyping firm in Belgium. He also worked in the Department of Mechanical Engineering of Katholieke Universiteit Leuven (K.U. Leuven), Belgium, and the Department of Space Vehicle Engineering of East China Institute of Technology (ECIT), China. He obtained a BSc in 1982 and an MSc in 1985 from ECIT, an MEngng in 1989 and a PhD in 1994 from K.U. Leuven.

Xiaohu Ma is an Associate Professor in the Department of Computer Science at Xuzhou Normal University, China. He worked as a research staff at City University of Hong Kong and Hong Kong Polytechnic University during the past two academic years. His major research interests include computer graphics, CAD and geometric modeling. He received his MSc and PhD degrees in Computer Science from Nanjing University in 1991 and Zhejiang University in 1997, respectively.

Shiu-Kit Tso is Professor (Chair) of Mechatronics and Automation in the City University of Hong Kong. He obtained his BSc (Engng) degree from the University of Hong Kong and his MSc and PhD degrees from the University of Birmingham, UK. Prior to joining the City University of Hong Kong, he had worked in the industrial electronics and control industry, and in the University of Hong Kong. He has held visiting professorships in many universities including the University of Toronto, University of Waterloo and National University of Singapore. He has served as the Chairman of the Control, Automation and Instrumentation Division, HKIE, Chairman of the IEEE Hong Kong Section and Chairman of the IEE Hong Kong Center.

Zhigeng Pan has been affiliated with the State Key Lab of CAD and CG, Zhejiang University, China, since 1993. He is at present the Director of the VR and Multimedia Division of the State Key Lab of CAD and CG. His current research interests include various issues of distributed graphics, virtual reality and multimedia. Some key issues addressed include time-dependent computer graphics, multi-resolution modeling and water making. He received his MSc and PhD degrees in Computer Science from Nanjing University in 1990 and Zhejiang University in 1993, respectively.