A NEW SIMPLIFICATION ALGORITHM FOR COLORED OR TEXTURED POLYGONAL MODELS

Kun Zhou, Mingmin Zhang, Jiaoying Shi
Zhejiang University, P.R. China

Zhigeng Pan
Hong Kong Polytechnic University, Hong Kong

ABSTRACT

Many applications in computer graphics require complex and highly detailed models. However, the level of detail actually necessary may vary considerably. It is often desirable to use approximations in place of excessively detailed models to control processing time. A new polygonal mesh simplification algorithm is presented for colored or textured models based on vertex clustering, and a more accurate error-measuring method for vertex clustering is introduced. The algorithm can produce high quality approximations of polygonal models. It makes adaptive subdivision of the bounding box in the original model using octree structure and performs vertex clustering in an error range specified by users. The color or texture information defined over the mesh can be preserved during simplification by constructing a texture map for the simplified mesh. To make a continuous transition between level of detail (LoD) models possible, an efficient interpolating method is also proposed. The efficiency of the algorithm is demonstrated in the experimental results.

1. Introduction

Many computer graphics applications require complex, highly detailed models to maintain a convincing level of realism. Consequently models are often created or acquired at a very high resolution to meet the requirement of details. However, the full complexity of such models is not always required, and rendering cost of a model is directly related to its complexity. It is useful to have simpler versions of complex models [1].

Surface simplification algorithms can be divided into three classes: adaptive subdivision [2], geometry removal [3-10], and re-sampling [11]. Many algorithms fall into the second catagory, and may be broadly categorized into the following 3 sub-classes:

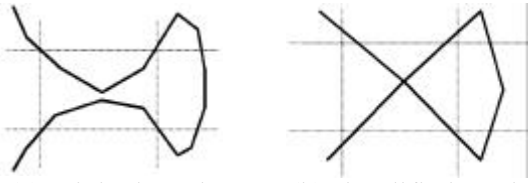
Vertex or triangle decimation [3,6,7]: Schroeder et al [3] described an algorithm that iteratively selects a vertex for removal, removes all adjacent faces, and re-triangulates the resulting hole. Ma et al. [6] presented a new algorithm which deletes unimportant triangles and re-triangulates the resulting hole.

Vertex clustering: Rossignac et al [5] presented a surface simplification algorithm based on vertex clustering. A bounding box is placed around the original model and divided into a grid of cells. Within each cell all vertices are clustered together into a single vertex and the model faces are updated accordingly. This process can be very fast and creates a significant topological alternative for the model.

Edge contraction [4,8-10]: Several algorithms simplify models by contracting edges. The essential difference between these algorithms lies in how they choose an edge to contract. Garland's algorithm [10] can rapidly produce high quality approximations to polygonal models. It uses iterative contractions of vertex pairs to simplify models and maintains surface error approximations using quadric error metrics. By contracting arbitrary vertex pairs, the algorithm is able to join the unconnected regions of models.

Among the algorithms listed above, the algorithm presented by Rossignac and Borrel is the fastest and thus most suitable for use in interactive applications [12]. In addition, it is capable of processing arbitrary polygonal input. However, this method has some obvious drawbacks. First, since the distribution of vertices is unknown when the bounding box is uniformly subdivided, it is likely that some cells will contain a lot of vertices, while others will contain few vertices. This situation may cause some part of the model to be over simplified. In addition, more space and time are required. Second, when generating a new vertex for a cell, only the weighted center is selected, and no better error controlling method is available. Third, there is possibility that the vertices falling into a cell (or box) may belong to different parts (or objects, if the scene may be described by individual objects). Thus the simplification may cause noticeable distortion (see Figure 1 as an example). Simplified mesh accuracy is improved by Low et al [13] by adopting enhanced vertex weighting and floating cell clustering. Fourth, color and other mesh attributes are not taken into account in both Rossignac's and Low's algorithms.

To address these limitations we present a new algorithm for colored or textured polygonal models based on vertex clustering. The work presented in this paper can be viewed as an extension of Rossignac's algorithm. Our algorithm employs octree structures to perform an adaptive subdivision of the bounding box of the original model. The color or texture attributes defined over the mesh can be preserved during simplification by constructing a texture map for the simplified mesh.



(a) original mesh (b) simplified mesh after vertex clustering

Figure 1 The distortion with Rossignac's algorithm

In addition, we present an efficient method for controlling the error in vertex clustering. The speed of our algorithm is comparable to that of Rossignac's, and the quality of simplified models is greatly improved. Experimental results show the efficiency of our algorithm.

Our mesh simplification algorithm for colored or textured models is composed of two separate processes: (a) the geometric simplification algorithm based on vertex clustering; and (b) the algorithm for controlling the color or texture attributes. In Section 2 we first provide a brief overview of the vertex clustering method; then discuss our enhancements toward high quality approximations. In Section 3 we discuss the computation of the error matrix used in our algorithm. In Section 4 we present a method for preserving color or texture information. In Section 5 the interpolating method for LoD models is discussed. In Section 6 experimental results are given, and finally, Section 7 gives our conclusions.

2. Geometric Simplification Algorithm

As with most other research work done in this area, we focus on the simplification of polygonal models. We assume that the model consists of triangles only. This implies no loss of generality, since every polygon in the original model can be triangulated as part of a preprocessing phase.

2.1 Overview of Vertex Clustering

The vertex clustering method can achieve a high data reduction rate with low computational cost by omitting the preservation of model topology. The clustering process determines the closeness of the vertices in the object space, and a new representative vertex is created to replace those vertices found to be close to one another,. The whole process has the following steps [5,13]: (1) grading: a weight is computed for each vertex according to its visual importance. (2) triangulation: polygons are divided into triangles. (3) clustering: vertices are grouped into clusters based on geometric proximity. (4) synthesis: a representative vertex is computed to replace the vertices in each cluster. (5) elimination: duplicated triangles, edges, and points are removed. (6) adjustment of normals.

2.2 DEFINITIONS

To facilitate the description of our algorithm, we first introduce some basic symbols and concepts.

Definition 1: Given a set of triangles, if each triangle has a common edge with every adjacent triangle, then these triangles are called a triangle mesh (call it *TM*), *TM* may be expressed by a vertex set *V* and a triangle set *T*, where $V=(V_1,V_2,\ldots\ldots V_n)$, $T=(T_1, T_2, \ldots\ldots T_m)$. In Figure 2, *V=(a,b,c,d,e,f,g), T=*(*Dafb, Dbfd, Ddec, Ddge, Ddfg, Dfag*).

Definition 2: If an edge in *TM* is shared only by one triangle, then this edge is called a boundary edge, and the vertices of this edge are called boundary vertices. Moreover, the triangle containing

this edge is called boundary triangle. In Figure 2, *ge, ec, cb, ab,* and *ga* are boundary edges, and vertices *g,e,c,b,a* are called boundary vertices.

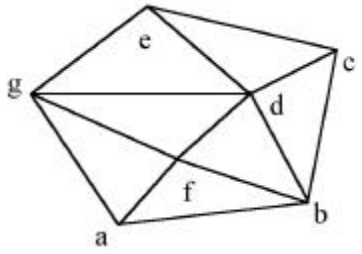


Figure 2 Boundary edge and related triangle set

Definition 3: For a vertex V_i in *TM*, the triangles which contain vertex V_i are called the related triangle set (denoted as P_i) of vertex V_i. In Figure 2 the related triangle set of vertex *f* is composed of the following triangles: *Dafb, Dbfd, Ddfg,* and *Dfag*.

Definition 4: For a vertex set $Vs = \{V_0, V_1, \ldots V_r\}$, the union of related triangle sets is called related triangle set (noted as $Ts(Vs)$) of the vertex set *Vs*, where

$$Ts(Vs) = \bigcup_{v_i \in Vs} P_i(V_i)$$

2.3 GENERATION OF NEW VERTEX AND ERROR METRIC

Let us first consider the following problem: if there are a set of vertices inside a cell, when they are deleted, how is the new vertex generated to represent the deleted vertices, and how do we compute the approximation error. If the vertex set Vs_i is clustered into a new vertex, then we allocate a 4×4 symmetric matrix Q_i for vertex set Vs_i. The matrix Q_i is called the error matrix (see Section 3 for detailed discussion on the evaluation of error matrix), and the cost of this vertex clustering operation is determined by the following equation:

$$\begin{aligned} e(Vs_i) &= v_{i0}{}^T Q_i v_{i0} \\ &= q_{i11}x_{i0}^2 + 2q_{i12}x_{i0}y_{i0} + 2q_{i13}x_{i0}z_{i0} + 2q_{i14}x_{i0} + q_{i22}y_{i0}^2 \\ &\quad + 2q_{i23}y_{i0}z_{i0} + 2q_{i24}y_{i0} + q_{i33}z_{i0}^2 + 2q_{i34}z_{i0} + q_{i44} \end{aligned} \tag{1}$$

The location of v_{i0} may has multiple possibilities. The simplest method is to select the weighted center of the vertex set Vs_i, which is not optimized. The best method is to select the vertex v_{i0} that minimizes $e(Vs_i)$. We may evaluate the partial derivatives of x_{i0}, y_{i0} and z_{i0} in equation (1), and set them to zero. Then we have:

$$\partial e(Vs_i)/\partial x_{i0} = \partial e(Vs_i)/\partial y_{i0} = \partial e(Vs_i)/\partial z_{i0} = 0$$

and we obtain the following:

$$\begin{bmatrix} q_{i11} & q_{i12} & q_{i13} & q_{i14} \\ q_{i12} & q_{i22} & q_{i23} & q_{i24} \\ q_{i13} & q_{i23} & q_{i33} & q_{i34} \\ 0 & 0 & 0 & 1 \end{bmatrix} v_{i0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2}$$

If this equation has a solution, then the location of the new vertex v_{i0} is determined by equation (3). Otherwise we may select one from all vertices or weighted center of Vs_i as the location of v_{i0} to minimize $e(Vs_i)$.

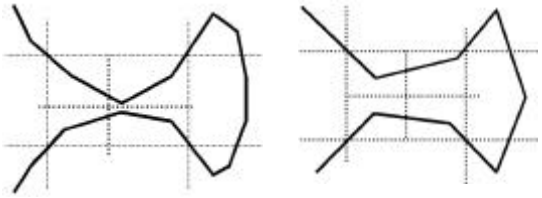
$$v_{i0} = \begin{bmatrix} q_{i11} & q_{i12} & q_{i13} & q_{i14} \\ q_{i12} & q_{i22} & q_{i23} & q_{i24} \\ q_{i13} & q_{i23} & q_{i33} & q_{i34} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3}$$

2.4 ADAPTIVE SUBDIVISION OF MESH MODEL

Our algorithm can perform adaptive subdivision of the bounding box of the original model. The basic tool we employ is the octree structure. The bounding box of the original mesh model is selected as the root node of the octree, and all vertices are inserted into the octree in order. When a vertex is inserted into one of the leaf nodes in the octree, if there is no vertex in the leaf node, the vertex is put into the node directly. Otherwise, we need to determine: (a) whether the size of the bounding box represented by this leaf node is greater than the maximum size the user specified; or (b) whether the number of vertices is greater than the maximum vertex number a box may contain.

If case (a) or (b) is true, the leaf node is subdivided, and the vertices in the node are allocated into its child nodes. Otherwise, to guarantee the condition presented in Section 3 that all vertices in each box should be related, and to avoid distortion in the clustering operation, the algorithm is needed to determine the following. If, when the vertex is put into the cell, the normals of some triangles in the related triangle set of new vertex set are opposite the original normals of the faces in the cell, then the node needs to be subdivided.

With the construction process described above, we can obtain an adaptive subdivision for the original mesh model. Figure 3 shows the result of adaptive subdivision to the mesh of Figure 1. Solid lines represents the mesh, and dotted lines represent subdivision lines. It is obvious that the simplification result in Figure 3(b) preserves the basic features of the original mesh in Figure 3(a).



(a) Original mesh *(b) Simplified mesh after vertex clustering*

Figure 3 Simplification of a typical mesh

2.5 Algorithm Outline

Our geometric mesh simplification algorithm consists of the following steps:

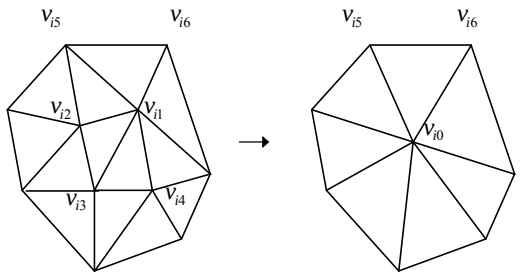
Step1: Create an initial octree subdivision using the method described above.
Step2: For every non-empty leaf node, compute the error matrices of the vertices in the box expressed by the node.
Step3: If the error is lower than the specified value (say ε), perform vertex clustering and compute the new vertex position and its corresponding approximation error. Otherwise subdivide the node and repeat the vertex clustering process until some specified error requirement is satisfied.
Step4: Finish

3. Computation of the Error Matrix

The 4×4 error matrix Q_i defined in the last section may differ according to the error measuring strategy employed by the algorithm. In our method we use the distance from a vertex to its average plane as the basic measurement of error.

We first assume that the vertices in a cell are related to each other, that is to say, every vertex has at least one edge connecting to other vertex in the vertex set $Vs_i = \{v_0, v_1, \cdots, v_k\}$ inside the cell. Subsection 2.4 shows how our algorithm guarantees this assumption.

Based on this assumption, we can first obtain the relative triangle set $Ts(Vs_i)$ of Vs_i. It is obvious that the vertex set constitutes a specific area in the original model. Figure 4 illustrates a very simple example. The vertex set $\{v_{i1}, v_{i2}, v_{i3}, v_{i4}\}$ is clustered to a new vertex v_{i0}.



(a) original mesh *(b) simplified mesh after vertex clustering*

Figure 4 The illustration of vertex clustering operation

According to the approximate error-evaluating formula for the simplified model to the original model [10], we define the error of the vertex set $e(Vs_i)$ as the sum of $e_{i,1}$ and $e_{i,2}$, where $e_{i,1}$ is the sum of squared distances of $v_{i0} = [x_{i0} \quad y_{i0} \quad z_{i0} \quad 1]^T$ to all triangles in the triangle set, and $e_{i,2}$ is the sum of squared distances of vertices in Vs_i to triangles related to v_{i0}. We set

$$e(Vs_i) = e_{i,1} + e_{i,2} \tag{4}$$

This error value should be minimized to obtain a high quality approximation. It is obvious that our error measuring method for vertex clustering is more accurate than that of Garland's method of vertex contraction [10] in which the error value is defined as the first term $e_{i,1}$.

(1) Evaluation of $e_{i,1}$

The evaluation of $e_{i,1}$ is similar to Garland's method. Let $p=[a \ b \ c \ d \ ]^T$ represent the plane defined by the equation $ax+by+cz+d=0$ where $a^2+b^2+c^2=1$, $e_{i,1}$ is expressed by

$$S_{i,1} = \sum_{p \in Ts(Vs_i)} (p^T v_{i0})^2 \tag{5}$$

Translating Equation (5) into the form of equation (1) gives

$$\begin{aligned} e_{i,1} &= \sum_{p \in Ts(Vs_i)} \left((v_{i0}^T p)(p^T v_{i0}) \right) \\ &= \sum_{p \in Ts(Vs_i)} \left(v_{i0}^T (pp^T) v_{i0} \right) = \sum_{p \in Ts(Vs_i)} \left(v_{i0}^T M_p v_{i0} \right) \\ &= v_{i0}^T \Big(\sum_{p \in Ts(Vs_i)} M_p \Big) v_{i0} = v_{i0}^T Q_{i,1} v_{i0} \end{aligned} \tag{6}$$

where M_p is a 4×4 symmetric matrix, and

$$M_p = pp^T = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} \tag{7}$$

If vertex set Vs_i contains any boundary vertex, we generate a perpendicular plane running

through the boundary edge to preserve the boundary feature of the original model. The planes are taken into account when the error matrix is computed. This method works quite well for preserving the boundary discontinuity.

(2) Evaluation of $e_{i,2}$

Since the location of v_{i0} is unknown, the equation of the plane where the triangle related to v_{i0} resides is also unknown. The computation of $e_{i,2}$ is quite complex. In Figure 4 we first evaluate the squared distance (d) of v_{i1} to the triangle $\{v_{i0}, v_{i5}, v_{i6}\}$ and set

$$v_{i1} = \begin{bmatrix} x_{i1} & y_{i1} & z_{i1} & 1 \end{bmatrix}^T$$

$$v_{i5} = \begin{bmatrix} x_{i5} & y_{i5} & z_{i5} & 1 \end{bmatrix}^T$$

$$v_{i6} = \begin{bmatrix} x_{i6} & y_{i6} & z_{i6} & 1 \end{bmatrix}^T$$

Then the equation of the plane in which triangle $\{v_{i0}, v_{i5}, v_{i6}\}$ resides is

$$\begin{vmatrix} x & y & z & 1 \\ x_{i0} & y_{i0} & z_{i0} & 1 \\ x_{i5} & y_{i5} & z_{i5} & 1 \\ x_{i6} & y_{i6} & z_{i6} & 1 \end{vmatrix} = 0 \tag{8}$$

If we then let

$$a_{i0} = \begin{vmatrix} y_{i0} & z_{i0} & 1 \\ y_{i5} & z_{i5} & 1 \\ y_{i6} & z_{i6} & 1 \end{vmatrix} \quad b_{i0} = -\begin{vmatrix} x_{i0} & z_{i0} & 1 \\ x_{i5} & z_{i5} & 1 \\ x_{i6} & z_{i6} & 1 \end{vmatrix} \quad c_{i0} = \begin{vmatrix} x_{i0} & y_{i0} & 1 \\ x_{i5} & y_{i5} & 1 \\ x_{i6} & y_{i6} & 1 \end{vmatrix}$$

where v_{i1}, v_{i5} and v_{i6} are known, $p=[a\ b\ c\ d\ ]^{\mathrm{T}}$ represents the triangle plane defined by the equation $ax+by+cz+d=0$ where $a^2+b^2+c^2=1$, and $M_p = pp^T$, we can define

$$a_{i1} = \begin{vmatrix} y_{i1} & z_{i1} & 1 \\ y_{i5} & z_{i5} & 1 \\ y_{i6} & z_{i6} & 1 \end{vmatrix} \quad b_{i1} = -\begin{vmatrix} x_{i1} & z_{i1} & 1 \\ x_{i5} & z_{i5} & 1 \\ x_{i6} & z_{i6} & 1 \end{vmatrix} \quad c_{i1} = \begin{vmatrix} x_{i1} & y_{i1} & 1 \\ x_{i5} & y_{i5} & 1 \\ x_{i6} & y_{i6} & 1 \end{vmatrix}$$

The squared distance of v_{i0} to the plane where the triangle $\{v_{i1}, v_{i5}, v_{i6}\}$ resides is then

$$v_{i0}{}^T M_p v_{i0} = \frac{\begin{vmatrix} x_{i0} & y_{i0} & z_{i0} & 1 \\ x_{i1} & y_{i1} & z_{i1} & 1 \\ x_{i5} & y_{i5} & z_{i5} & 1 \\ x_{i6} & y_{i6} & z_{i6} & 1 \end{vmatrix}^2}{a_{i1}^2 + b_{i1}^2 + c_{i1}^2} \tag{10}$$

Using Equation (10) and Equation (9), we then obtain

$$d = v_{i0}{}^{T}\left(\frac{a_{i1}^2 + b_{i1}^2 + c_{i1}^2}{a_{i0}^2 + b_{i0}^2 + c_{i0}^2} M_p\right) v_{i0} = v_{i0}{}^{T} M_p^* v_{i0}{}^{T} \tag{11}$$

It is obvious that $Q_{i,2}$ is only related to the location of v_{i0}, and the value of v_{i0} is unknown, so $Q_{i,2}$ is also unknown. The evaluation process of v_{i0} is equivalent to that of $Q_{i,2}$. This problem can be resolved by an iteration method. First, $Q_{i,2}$ is assigned an initial value (zero matrix) in the iteration process, and we get the initial value of the error matrix Q_i, called $Q_{i,1}$. The approximation value of v_{i0}, called v_{i0}^*, is then obtained using equation (3). Second, we evaluate the approximation value of $Q_{i,2}$, called $Q_{i,2}^*$, using the approximation value of v_{i0}, and then evaluate the next approximation value of v_{i0}, which we call v_{i0}^*. This iteration process proceeds until the difference of the two successive approximation values is smaller than a specified value. Since this iteration process is convergent and the initial value is well defined, the convergence speed is very fast. Experimental results show that the approximation result is very good after the process is iterated 3 times. With such a method we can get good accuracy in the vertex clustering operation.

4. Controlling Color and Texture Attributes

4.1 RELATED WORK

The preservation of the color or texture attribute discontinuity during the mesh simplification is an important feature. The control of the purely geometric approximation is not sufficient in many applications to assure the required accuracy.

Hoppe [8] presented a method for processing scalar attributes, such as color or normal. A separate energy term E_{scalar} is introduced to measure attribute deviation. The method is simple, but it cannot process a complex attribute such as texture. In Certain's method [14] a geometric attribute and color information are represented using a multi-resolution form. Since parameterization of mesh and the wavelet transform are required, its implementation is very complex, and the method for processing the texture attribute is not addressed.

Soucy et al [15] presented a texture-mapping approach for the compression of colored 3D triangulation. In his method, once the mesh is simplified, a texture is built for each face using colors of the associated removed vertices. The texture space is tessellated to accommodate the triangles of the simplified model. Each triangle has its own texture space. Each high-resolution vertex is projected in the texture space, and its RGB colors are associated with the nearest texel center. Finally, an interpolation algorithm assigns each of the remaining empty texels the RGB values of the nearest texel center on the 3D surface. Constraints are imposed to ensure that the texture space is continuous. This method can maintain good visual similarity between the original and the simplified models. However, it has the following limitations:

(a) It requires that the vertices in the simplified mesh must be a subset of the vertices in the original mesh. Thus it is not suitable for other simplification methods such as vertex clustering or edge collapsing.
(b) The algorithm cannot process the texture attribute.
(c) The 3D nearest-neighbor interpolation method is employed for empty texels that is quite complex.
(d) The corresponding texture map needs to be computed for each simplified model.

4.2 OUR METHOD

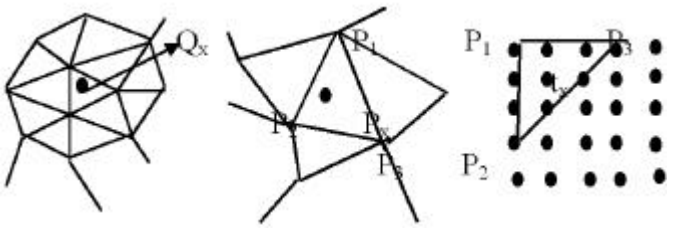
We present a unified method for processing color and texture information to address color and texture problems. The method can be considered as a modified version of Soucy's algorithm. We also construct a texture map for simplified model. However, when evaluating RGB values for each texel in the texture space, the implementation of our method is more straightforward and efficient. In Soucy's method the vertices in the high-resolution model are mapped to texture triangles corresponding to triangles in the simplified model, and that results in two limitations: (a) multiple vertices are mapped to a single texel; and (b) there may be unassigned texels (or empty texels). Our method evaluates the corresponding point and its RGB value in the high-resolution model for each texel in the texture triangle to avoid the above two limitations.

In addition, our algorithm is independent of the mesh simplification process of the geometric attribute. It can be integrated into any mesh simplification algorithm. Furthermore, it can also process the texture attribute defined in the original mesh. Our method is composed of two stages:

(a) mapping each triangle in the simplified mesh to a texture triangle in texture space;
(b) evaluating RGB values of each texel in texture triangle.

It is assumed that an initial high-resolution triangle mesh is available, and the RGB color components or texture coordinates are attached to each triangulation vertex.

(1) *Mapping each triangle in simplified mesh to a texture triangle in texture space:* This stage is similar to that of Soucy's algorithm. The texture coordinates are recorded for each triangle in the simplified mesh, determining the corresponding triangle in the specified area of texture space. It is noted that each texture triangle should not overlap other texture triangles. For convenience of operation and maintaining the continuity of texture space, a rectangular texture map is tessellated into a set of half-square texture triangles with dimensions that are powers of 2. Thus the vertices of texture triangles are only located at the position of texels. The areas of the triangles in the simplified mesh determine the area of the texture triangle. In Figure 5, $\Delta P_1P_2P_3$ in the simplified mesh corresponds to the texture triangle $\Delta p_1p_2p_3$ in texture space.



(a) Original mesh (b) simplified mesh (c) texture space

Figure 5 Computation of RGB value of texel t_x

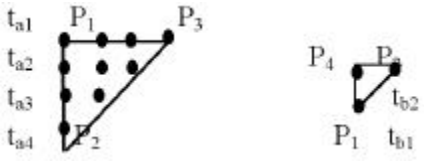


Figure 6 Two texture triangles with common edge P_1P_2

(2) *Evaluating RGB values for each texel in the texture triangle.* For the texture triangle $\Delta p_1p_2p_3$ we may assume that the corresponding triangle in the simplified mesh is $\Delta P_1P_2P_3$. For each texel t_x contained in $\Delta p_1p_2p_3$ the corresponding point P_x in $\Delta P_1P_2P_3$ is computed to determine point Q_x on the original mesh that is the nearest to P_x. Then we can assign the RGB value of Q_x to t_x. (The RGB value of Q_x may be evaluated according to the color attribute or texture attribute attached to the original mesh). With this method we can compute the RGB values for each texel in the texture triangle.

Some constraints are imposed to this stage to retain continuity of the texture space. For example, in Figure 6 the corresponding triangle in the simplified mesh of the texture triangle $\Delta p_1p_2p_3$ has a common edge (P_1P_2) with the corresponding triangle in the simplified mesh of the texture triangle $\Delta p_1p_2p_4$. The following conditions should be satisfied to keep the continuity of texture space: (a) $t_{a1} = t_{a2} = t_{b1}$; (b) $t_{a3} = t_{a4} = t_{b2}$.

With the two stages mentioned above we can compute the texture coordinates in texture space for vertices in the simplified mesh, thereby obtaining a texture map for the simplified mesh. Thus the color or texture attributes of the original mesh are preserved in the simplified mesh.

5. Interpolation Between Different LoD Models

Turk [11] presented an interpolation algorithm between two different LoD models. In his method vertices of the coarse model must be a subset of the vertices of the detailed model, so it is not suitable for LoD models generated by a simplification algorithm based on vertex clustering, edge collapses, or triangle collapses. In addition, the implementation of the algorithm is very complex.

In this section we present a very simple and efficient method for model interpolation. It works well for any kind of LoD model. The basic idea of our algorithm is to find the corresponding points in both models and perform linear interpolation directly. The algorithm is described as follows:

Step 1: For every vertex in the coarse model, find the nearest vertex in the detailed model. This builds a relationship between the two vertices.

Step 2: For those vertices in the detailed model which do not have corresponding vertices, try to find the nearest vertex in the coarse model. This also builds a relationship between the two vertices.

Step 3: Given an interpolation parameter t ($0 \pounds t \pounds 1$), for each vertex P_h in the detailed model we assume that its corresponding vertex in coarse model is P_l. Then the interpolated point is $P=tP_l+(1-t)P_h$ in the intermediate model.

Step 4: For a given error measure ε, if the distance between two adjacent vertices is lower than ε, then the two vertices are collapsed into one, and the degenerated triangles are removed.

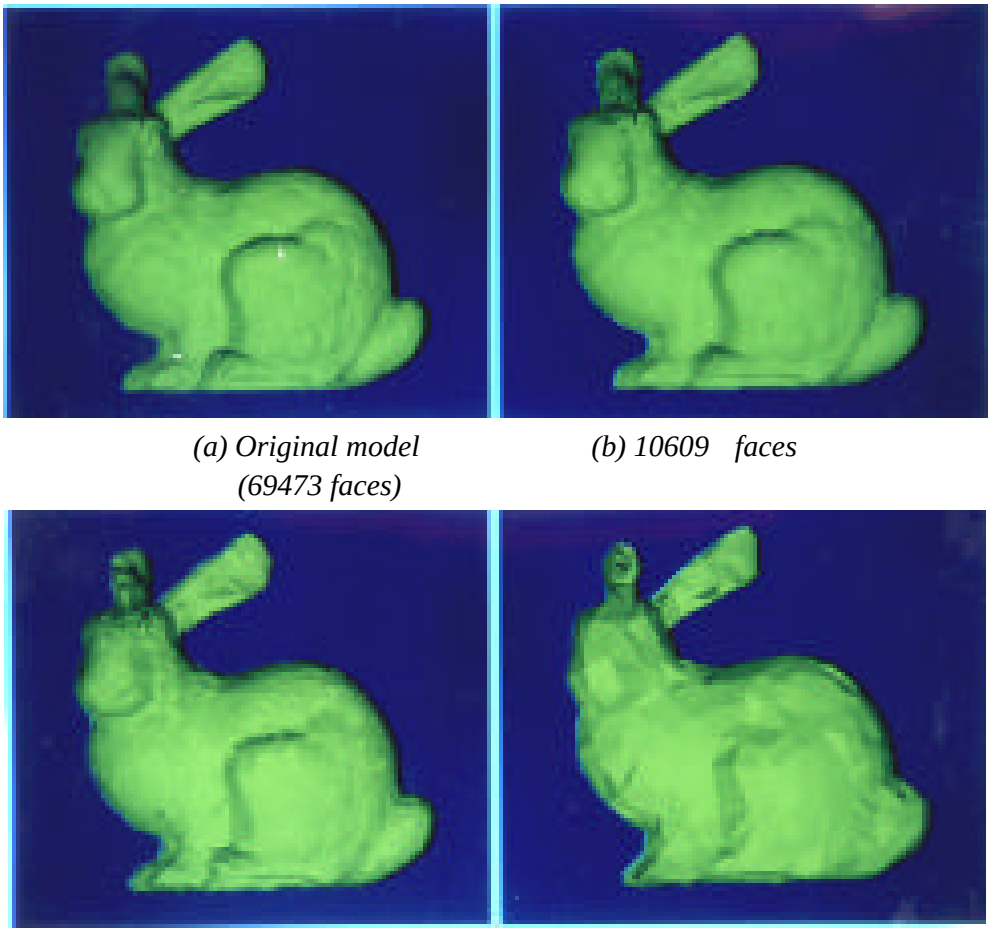
Step 5: Finish.

It is obvious that our method is very straight forward. Moreover, we construct an octree structure for the coarse model and a detailed model to accelerate the executing speed in step 1 and step 2. Since the topology of the intermediate models is the same as that of the detailed model for colored or textured models, computation of the texture map for the intermediate model generated by interpolation is not required. It can be rendered with the texture map of the detailed model.

6. Experimental Results

Since the algorithm may be used to simplify meshes with a large number of vertices, a well-designed data structure is required. In our algorithm we use the following data structures: (a) a vertex table; (b) a triangle table; and (c) an octree for subdividing the mesh model. The new vertex coordinate is put into the original vertex table directly to save space.

This algorithm is implemented with the C programming language. Some experimental results are shown in the color photos of Figure 7 to Figure 13. From these figures we can see that our algorithm is efficient for both smooth and non-smooth models. The simplified model shown in Figure 7(b) is very similar to the original model, and the result is also acceptable when 98.6% of the triangles are deleted (Figure 7(d)).

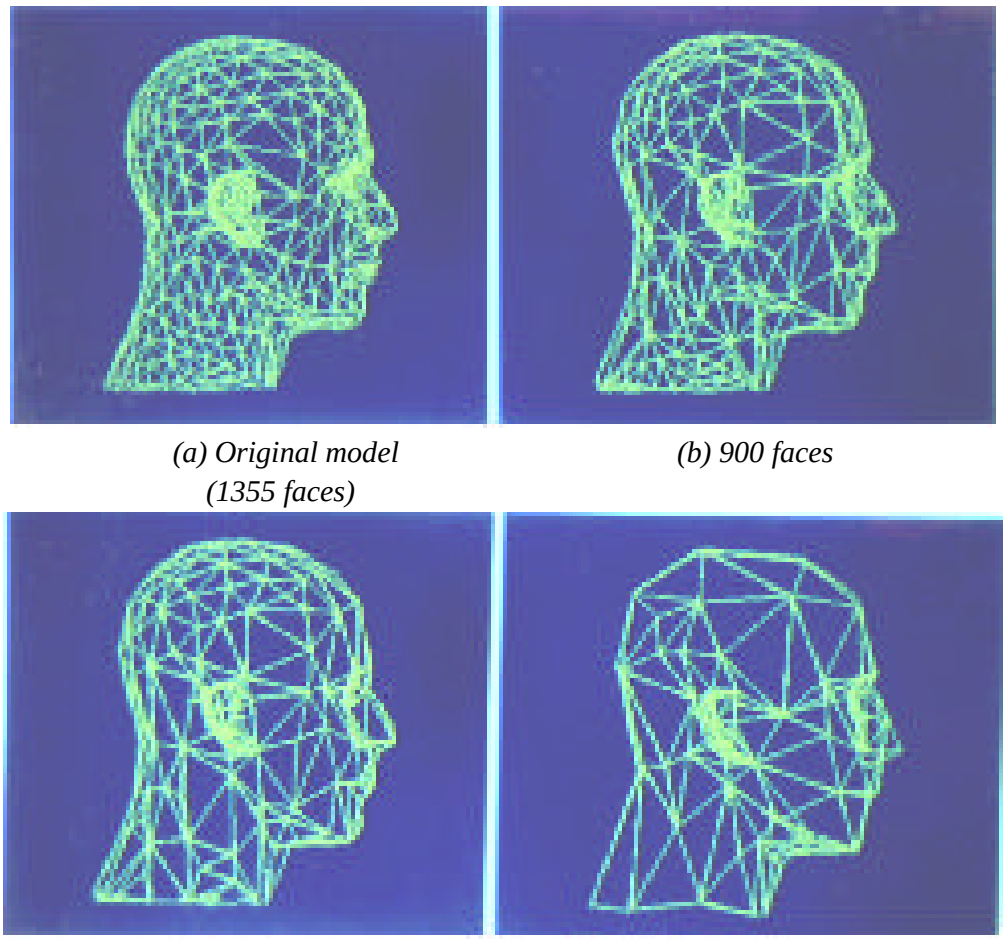


(a) Original model (69473 faces) *(b) 10609 faces*

(c) 2682 faces *(d) 1000 faces*

Figure 7 Simplification of bunny model

In Figure 8 we show the wire frame of a model of a human head. Since the number of triangles in the original model is not very large (1355), the model can only be simplified to a certain extent (the maximum being 76.6% triangles deleted).

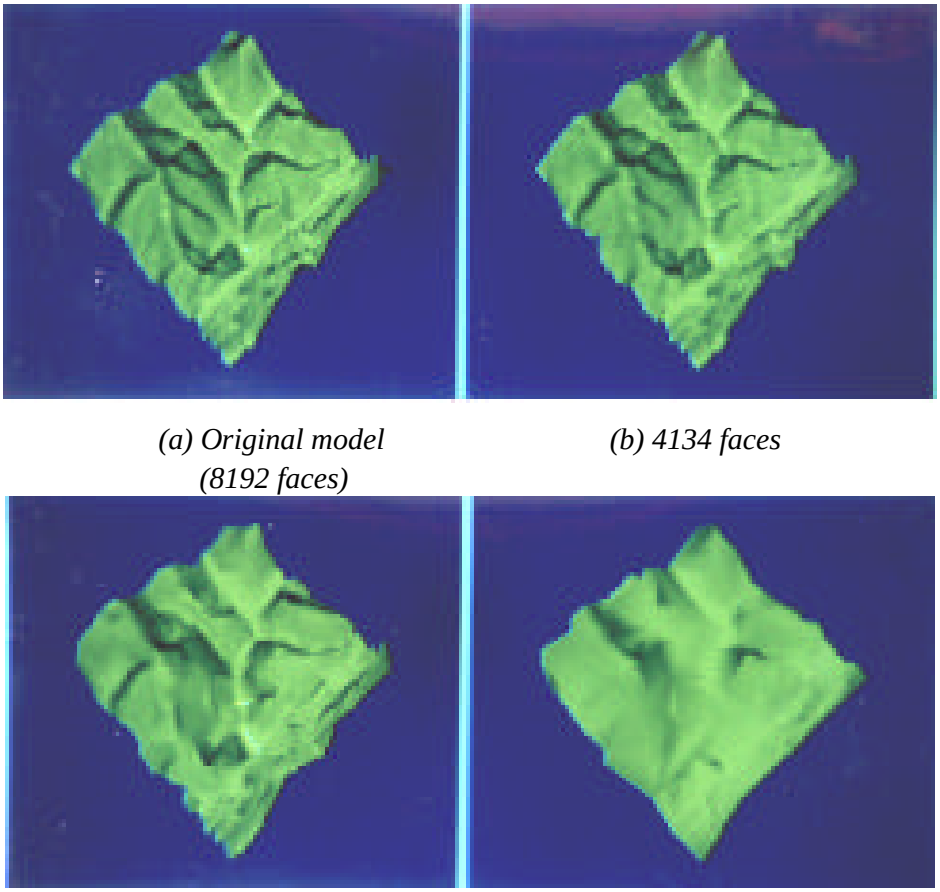


(a) Original model (1355 faces) *(b) 900 faces*

(c) 585 faces *(d) 317 faces*

Figure 8 Simplification of head model

It is noteworthy that our algorithm is especially suitable for terrain data. Figure 9(a) is the original model consisting of 8192 triangles. The simplified models at three levels of detail (49.5%, 74.8%, and 93.9% triangles deleted) are shown in Figure 9(b), 9(c), and 9(d), receptively. The boundary features are well preserved.

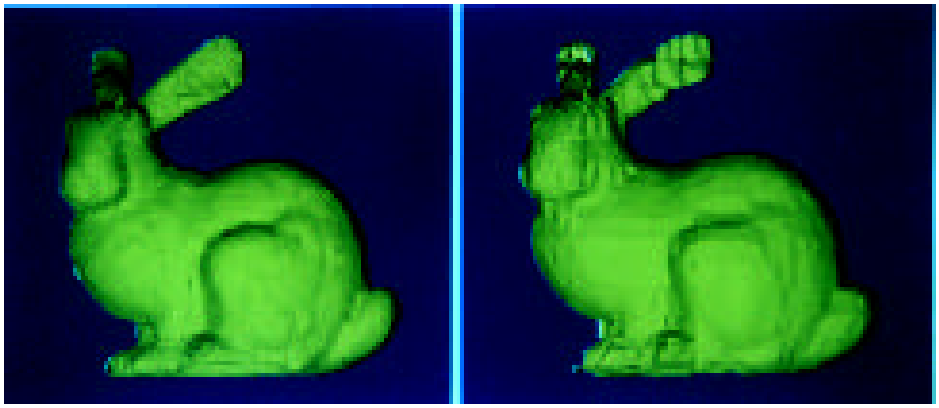


(a) Original model (8192 faces) *(b) 4134 faces*

(c) 2057 faces *(d) 503 faces*

Figure 9 Simplification of terrain model

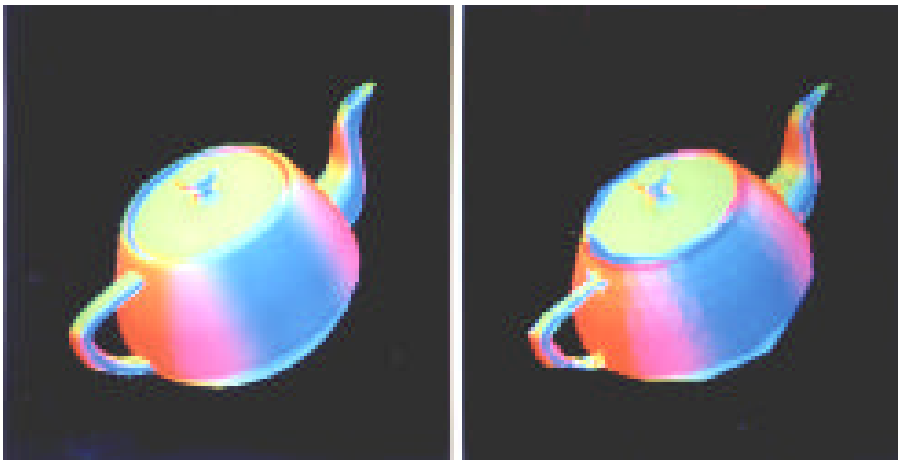
We also implemented Rossignac's algorithm [5] for comparison. Block distortion appears in Figure 10(b) when 95.8% of the triangles are deleted. Our method can avoid such distortion (see Figure 7).



(a)12505 faces *(b) 2898 faces*

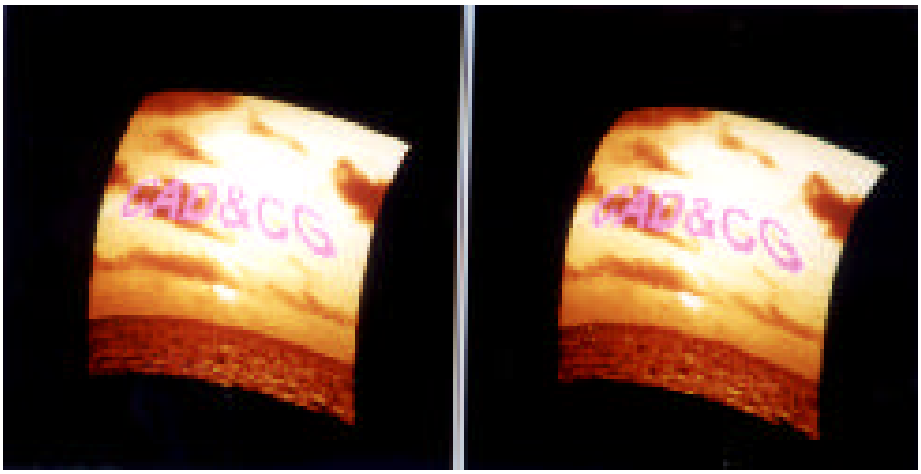
Figure 10 Simplification of bunny model with Rossignac's algorithm

In Figures 11 and 12 we show the processing result of the colored and textured models respectively.



(a)Original model with color attribute (3751 faces) *(b) Simplified model (200 faces)*

Figure 11 Simplification of colored teapot model



(a) Original model with texture (5000 faces) *(b) Simplified model of (a) (500 faces)*

Figure 12 Simplification of textured surface model

In Figure 13 the geometric interpolation results of the models are shown.

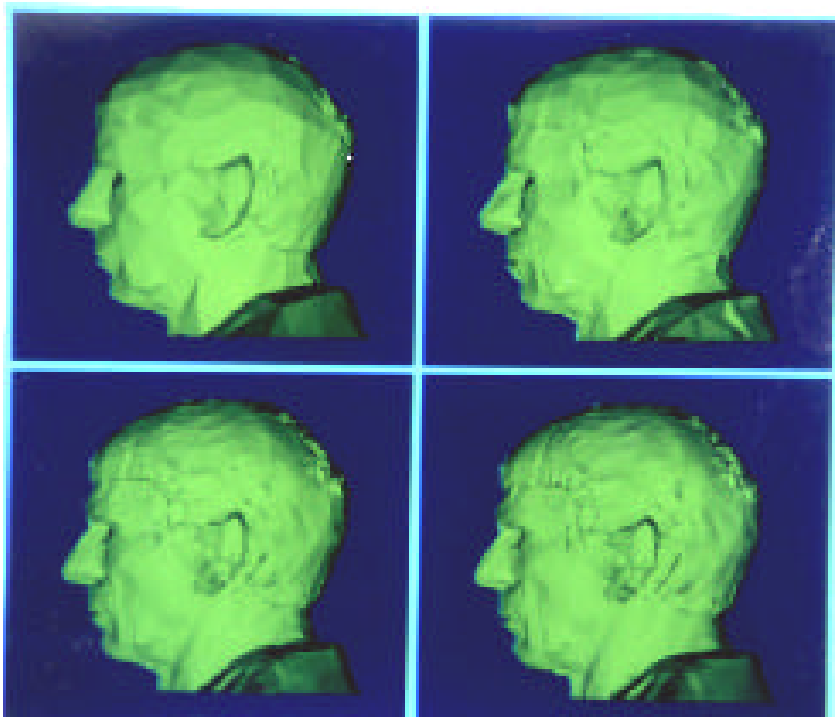


Figure 13 LoD models interpolation (without color)
top-left: low-resolution model(867 triangles)
top-right: interpolated model (t=0.4)
bottom-left: interpolated model (t=0.8)
bottom-right: high-resolution model (3525 triangles)

Table 1 gives the time comparison between our algorithm and Rossignac's. The data was tested on an IBM RS6000 Workstation. The speed of Rossiginac algorithm is about twice that of our algorithm. However, the quality of simplified meshes in our algorithm is much better.

Original model (tri)	Simplified model	Rossignac algorithm	Our algorithm(s)
Head(1355)	597 tri	0.31	0.52
Terrain(8192)	2057 tri	2.33	4.56
Bunny(69374	2772 tri	20.60	44.02

Table 1 Comparison of execution time of two algorithms

7. Conclusion and Future Work

The polygonal simplification algorithm presented in this paper is very fast and provides better approximation quality. It can also process models with color or texture information. The algorithm can be used to construct multiple LoD models that are widely used in interactive graphics applications such as those found in areas of virtual reality and interactive visualization systems.

Anticipated future work includes (a) study of the inverse operation of vertex clustering for constructing progressive mesh [8]. (b) experimenting with the error control factor $e_{i,2}$, and comparing the result with the result obtained with the error control factor $(e_{i,1}+e_{i,2})$, and (c) extending our algorithm to a view-dependent realtime algorithm. Since our algorithm employs octree subdivision, it is easy for us to use different error control values for different parts of the model. Therefore an extended simplifying algorithm might run faster than the current version of our algorithm.

8. Acknowledgement

This project is funded by the National Natural Science Foundation of China.

REFERENCES

[1] Pan, Z.G, Ma, X.H., Shi, J.Y., Overview of the automatic generation of LOD models, Journal of Image and Graphics, 3(9):754-759, 1998.

[2] De Haemer, Jr., et al.. Simplification of Objects Rendered by Polygonal Approximations. Computers&Graphics, 15(2): 175-184, 1991.

[3] Schroeder, William J., Jonathan A. Zarge, and William E. Lorensen. Decimation of Triangle Meshes. Computer Graphics , 26(2):65- 70, July 1992.

[4] Hoppe, Hugues, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh Optimization. Computer Graphics (SIGGRAPH Proceedings), 27:19-26, August 1993.

[5] Rossignac, Jarek R. and Paul Borrel. Multi-resolution 3D Approximations for Rendering Complex Scenes. Technical Report RC 17697,IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10958, 1993.

[6] Ma Xiaohu, Zhigeng Pan, Jiaoying Shi. Polygonal simplification based on triangle deletion, Journal of Computer , 21(6), 1998(in Chinese).

[7] Cohen, J., et al. Simplification envelopes, Computer Graphics 30(119-128), 1996.

[8] Hoppe, H. Progressive meshes, Computer Graphics, 30(99-108),1996.

[9] Ronfard, R., J.R.Rossignac. Full-range approximation of triangulated polyhedral, Computer Graphics Forum, 15(3):67-76,1996.

[10] Garland, M., P. Heckbert. Surface simplification using quadric error metrics, Computer Graphics (SIGGRAPH Proceedings), 31, 1997.

[11] Turk, Greg. Re-Tiling Polygonal Surfaces, Computer Graphics (SIGGRAPH

Proceedings), 26(2):55-64, July 1992.

[12] Luebke, K. and C. Erikison. View-dependent simplification of arbitrary polygonal environments. Computer Graphics (SIGGRAPH Proceedings), 31, 1997.

[13] Low, Kok-Lim, Tiow-Seng Tan, Model Simplification Using Vertex-Clustering, Proceedings of Symposium on Interactive 3D Graphics, Providence, RI, USA,April27-30, pp75-77,1997.

[14] Certain, A., J.Popovic et al. Interactive multiresolution surface viewing. Computer Graphics (SIGGRAPH Proceedings) , 30(91-98),1996.

[15] Soucy, Marc, Guy Godin, Marc Rioux . A texture-mapping approach for the compression of colored 3D triangulations, The Visual Computer,12(503-514),1996.

BIOGRAPHIES

Kun Zhou is a graduate student in the Department of Computer Science and Engineering at Zhejiang University. His research interests include time-critical rendering, multi-resolution modeling, and image-based rendering/modeling. He has published around 10 papers in various journals and proceedings related to VR. In 1999 he was awarded a Microsoft fellowship.

Contact information

Kun Zhou
State Key Lab of CAD&CG,
Zhejiang University, Hangzhou, 310027, China
Phone: +86-571-7951045
Fax: +86-571-7951780
Email: mailto:kzhou@cad.zju.edu.cn

Mingming Hang received an M.S. degree from Zhejiang University in 1995. Her research interests include multimedia, VR, and Chinese Computing.

Contact information:

Mingmin Zhang
State Key Lab of CAD&CG
Zhejiang University, P.R.China
Email: mailto:pzg@cad.zju.edu.cn

Zhigeng Pan is a Professor at the State Key Lab of CAD&CG, Zhejiang University, a member of the IS&T, and a senior member of the China Image and Graphics Association. He is a content editor for *The International Journal of Virtual Reality*, and is on the editorial board of Journal of Image and Graphics. He has published more than 60 papers and is author or co-author of three books. He acted as one of the guest editors for a special issue of *Computers & Graphics*. His current research interests include VR, distributed graphics, and multimedia. Currently he is visiting the Department of Computing of the Hong Kong Polytechnic University as a research fellow.

Contact information:

Dr. Zhigeng Pan
VR Lab, Department of Computing,
The Hong Kong Polytechnic University, HK, China
Phone: +852-27667281
Fax: +852-27740842
Email: mailto:cszgpan@comp.polyu.edu.hk
or mailto:pzg@cad.zju.edu.cn

Jiaoying Shi is a Professor of the Dept. of Computer Sci. & Eng. at Zhejiang University, China. He is the Director of Academic Committee of State Key Lab of CAD&CG. He is a member of SIGGRAPH Education Committee, and on the editorial board of the *International Journal of Computers & Graphics*. Since 1990 his work has concentrated on ViSC and VR. He has published more than 100 papers and three books.

Contact information:

Prof. Jiaoying Shi,
State Key Lab of CAD&CG,
Zhejiang University, Hangzhou, 310027, China
Phone: +86-571-7951045
Fax: +86-571-7951780
Email: mailto:jyshi@cad.zju.edu.cn