第25卷 第1期
1999年1月

自 动 化 学 报
ACTA AUTOMATICA SINICA

Vol. 25, No. 1
Jan., 1999

一种新的基于顶点聚类的网格简化算法[1)]

周 昆 潘志庚 石教英

(浙江大学 CAD & CG 国家重点实验室 杭州 310027)

摘 要 在计算机图形学中,经常采用多边形网格来描述物体模型.由于绘制时间和存储量与多边形的数量成正比,过于庞大的物体网格模型通常是不实用的.模型简化在计算机动画、虚拟现实和交互式可视化等计算机图形应用领域有着广阔的应用前景.为此提出一种新的基于顶点聚类的网格简化算法.该算法利用八叉树对网格进行自适应划分,给出了一种基于点到平面距离的有效的误差控制方法,并能在用户指定的误差范围内通过使原始网格中的顶点聚类达到大量简化的目的.该算法实现简单,速度快且能很好地保持边界特征.给出的一组图例说明了该算法的有效性.

关键词 多边形网格,网格简化,顶点聚类,细节层次.

A NEW MESH SIMPLIFICATION ALGORITHM BASED ON VERTEX CLUSTERING

ZHOU Kun, PAN Zhigeng, SHI Jiaoying

(*State Key Lab. of CAD&CG, Zhejiang University, Hangzhou* 310027)

Abstract In computer graphics, models are often discribed by polygonal meshes. Because the rendering and storage cost is proportional to the number of polygons, too large models are not practical. Model simplification techniques are widely used in computer graphics fields such as computer animation, virtual reality and interactive scientific visualization. In this paper a new algorithm of mesh simplification based on vertex clustering is presented. The algorithm adopts octree structure to subdivide the mesh model adaptively. A new error control method is also presented. The implementation of the algorithm is simple and it runs very fast. Examples illustrate the efficiency of the algorithm.

Key words Polygonal mesh, mesh simplification, vertex cluster, level of detail.

1 引 言

在计算机图形学应用领域,经常采用多边形网格来描述物体模型.使用激光扫描系统可以获得各种复杂的网格,另外,由三维模型重构方法得到的多边形网格模型通常由上万

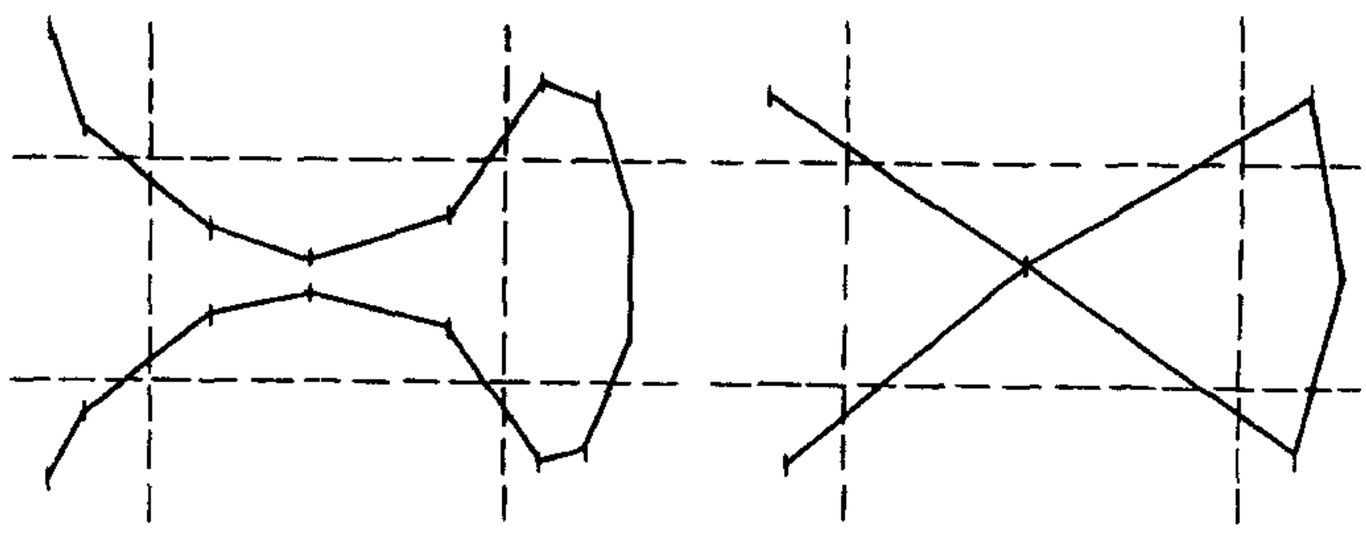
1)本课题得到国家自然科学重点基金项目"真实感图形实时生成和显示技术"和国家自然科学基金项目"虚拟现实中动态连续细节层次技术研究"共同资助.

收稿日期 1997-08-01 收到修改稿日期 1998-03-10

个、几十万个、甚至几百万个多边形面片组成. 由于绘制时间和存储量与多边形的数量成正比,过于庞大的物体网格模型通常是不实用的. 另一方面,在虚拟环境中,通常并不需要对物体的细节刻划得很详细的模型,我们可以为每个物体构造若干个不同细节层次的模型,根据需要选用某种模型. 为满足计算机分析、显示与存储的要求,必须对这类复杂的模型进行简化. 模型简化在计算机动画、虚拟现实和交互式可视化等计算机图形应用领域有着广阔的应用前景.

国外对模型简化的研究已有一系列成果. Schroeder[1]提出了基于顶点删除的网格删减方法;Turk[2]给出了基于重新划分的多边形网格模型简化方法;Hoppe[3]采用能量函数最优化的网格简化方法;Rossignac[4]提出了一种基于顶点聚类的网格简化方法;Eck[5]将小波技术用于模型简化;Cohen[6]通过构造包络网格来控制网格简化的全局误差;Garland[7]利用二次误差方法来控制网格简化误差. 国内在这方面也开展了一些卓有成效的研究工作[8,9,10].

在以上方法中,Rossignac[4]所提出的方法简单明了,速度最快. 其基本思想是:用一个包围盒把原网格模型包围起来,通过等分包围盒的各棱边将包围盒等分成若干个小的长方体,这样原模型的所有顶点就分别落在这些长方体内;扫描这些长方体,如果某个长方体内有顶点,则把该长方体内的所有顶点删除并生成一个新顶点,这个新顶点是被删除的原顶点的加权平均. 这种方法存在着三个局限性:首先,由于原网格模型上的点在空间的分布是未知的,这种方法对包围盒进行等分,可能导致等分后某些区域的长方体内包含很多的顶点,而某些区域的长方体内没有或只有很少的顶点,这一方面造成空间和时间的浪费,另一方面造成模型的某些部分过分简化;其次,生成某个长方体内的新顶点时,这种方法只是取简单的加权平均而并没有给出一种较好的误差控制方法;再次,当原网格模型比较特殊时,落在某些长方体内的顶点可能属于原模型的差别较大的部分,如果这些顶点聚类成一个顶点就会造成很大的变形. 图1就是一个例子,其中实线表示网格,虚线表示对网格包围盒的划分线. 图1(a)是原始网格,图1(b)是 Rossignac[4]方法得到的简化;不难看出,这种简化的变形太大.



(a)原始网格 (b)顶点聚类后生成的简化网格

图 1 Rossignac 算法对特殊网格造成的变形

本文提出了一种新的基于顶点聚类的网格简化算法,该算法利用八叉树给出了网格包围盒的自适应划分,同时给出了控制顶点聚类误差的有效方法,与 Rossignac[4]提出的算法相比,具有速度相当、效果更好且适用范围更广的优点,文中给出的一组图例说明了该算法的有效性. 由于多边形网格中最常见的是三角形网格,且任何多边形网格都可转化为三角形网格,下面所指的网格都是三角形网格.

2　算法描述

为叙述方便和清楚起见(参考文献[8]),首先引入一些基本概念.

2.1　基本概念

定义1. 空间中一组三角形,沿公共边及在顶点处相邻接,把这样的一组三角形定义为三角形网格 TM,TM 可由顶点集 $V=(v_1,v_2,\cdots,v_n)$和三角形集合 $T=(t_1,t_2,\cdots,t_m)$所组成的二元组(V,T)来表示.

定义2. 对 TM 中任一条边,如果该边只为一个三角形所享有,则称该边为边界边,该边的两个顶点被称为边界顶点,该边所在的三角形被称为边界三角形.

定义3. 对 TM 中任一顶点 v_i,所有以 v_i 为一个顶点的三角形 T_{ik}构成的集合,称为与顶点 v_i 相关的三角形集合 P_i.

定义4. 对 TM 中一个顶点集 $V_s=\{v_0,v_1,\cdots,v_k\}$,与 V_s 中的每个顶点相关的三角形集合的并集,称为与顶点集合 V_s 相关的三角形集合 $T_s(V_s)$,即

$$T_s(V_s)=\bigcup_{v_i\in V_s}P_i. \tag{1}$$

2.2　顶点聚类的新顶点的生成和误差的度量

首先考虑,如果一些顶点落在一个长方体内时,删除这些顶点后,如何生成一个新顶点以及用该新顶点代替被删除的顶点所产生的误差是多少. 这就有一个选择误差标准的问题,本文算法以点到平面的距离[7]为误差标准.

假设落在每个长方体内顶点和边所构成的图是连通的,在2.3节中我们将讨论算法是如何保证这一点的. 有了这个前提,现在来考虑一个顶点集合 $V_s=\{v_0,v_1,\cdots,v_k\}$落在一个长方体内时,如何求新顶点和简化误差. 我们可以先得到与顶点集合 V_s 相关的三角形集合 $T_s(V_s)$,由前面的假设不难发现,这个三角形集合构成了原网格模型上的一个区域. 图2就是一个简单的例子,顶点集合为$\{v_1,v_2,v_3,v_4\}$,聚类为 v_0. 如果顶点集合 V_s 聚类为顶点 $v_{i0}=[x_{i0}\quad y_{i0}\quad z_{i0}\quad 1]^{\mathrm{T}}$,定义这个聚类带来的误差 $\varepsilon(V_s)$为 v_{i0}到三角形集合 $T_s(V_s)$中每个三角形所在平面的距离平方之和 $\varepsilon_{i,1}$加上 V_s 中的顶点到每个与 v_{i0}相关的三角形所在平面的距离平方之和 $\varepsilon_{i,2}$,即

$$\varepsilon(V_s)=\varepsilon_{i,1}+\varepsilon_{i,2}. \tag{2}$$

为了使 $\varepsilon(V_s)$尽可能小,可以对式(1)中的 x_{i0},y_{i0}和 z_{i0}求偏导数,并令其为零,即

$$\partial\varepsilon(V_s)/\partial x_{i0}=\partial\varepsilon(V_s)/\partial y_{i0}=\partial\varepsilon(V_s)/\partial z_{i0}=0. \tag{3}$$

令 $p=[a\quad b\quad c\quad d]^{\mathrm{T}}$ 为每个与顶点集合 V_s 相关的三角形集合 $T_s(V_s)$中的三角形所在的平面的平面方程 $ax+by+cz+d=0$,且有 $a^2+b^2+c^2=1$,$\varepsilon_{i,1}$可表示为

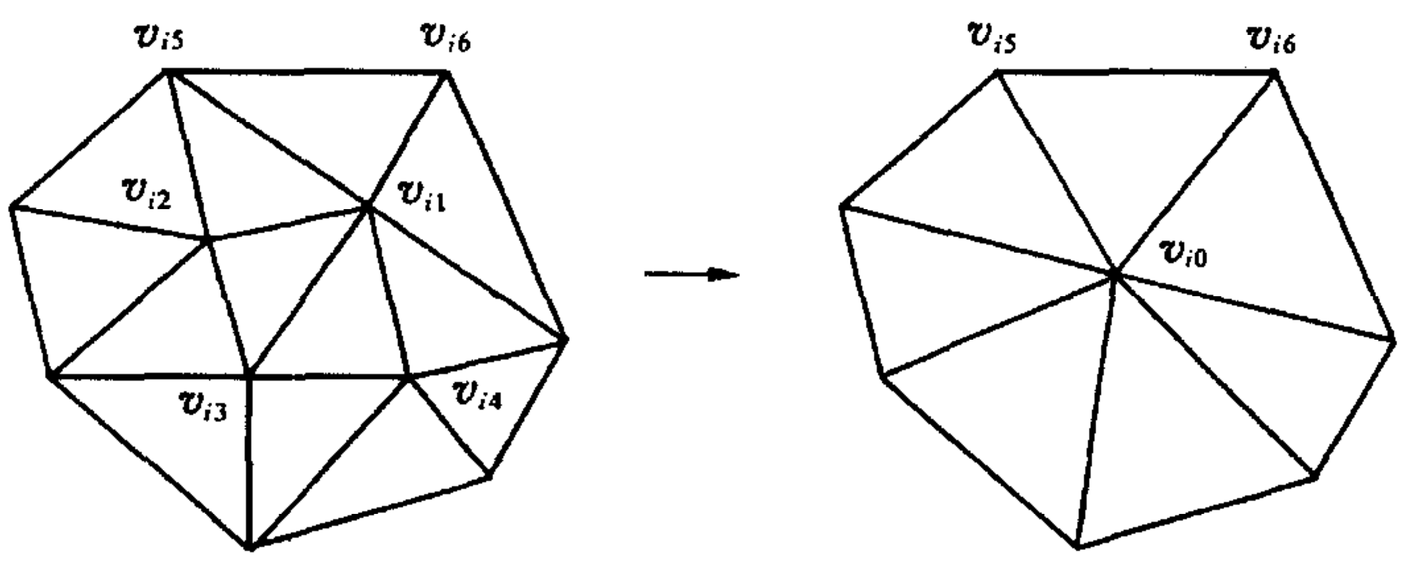
$$\begin{aligned}\varepsilon_{i,1}&=\sum_{p\in T_s(V_s)}((v_{i0}^{\mathrm{T}}p)(p^{\mathrm{T}}v_{i0}))=\sum_{p\in T_s(V_s)}({v^T}_{i0}(pp^{\mathrm{T}})v_{i0})\\&=\sum_{p\in T_s(V_s)}(v_{i0}^{\mathrm{T}}M_pv_{i0})=v_{i0}^{\mathrm{T}}(\sum_{p\in T_s(V_s)}M_p)v_{i0}=v_{i0}^{\mathrm{T}}Q_{i,1}v_{i0},\end{aligned} \tag{4}$$

其中 M_p 是4×4的对称矩阵 $T_s(V_s)$

$$M_p = pp^{\mathrm{T}} = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}. \tag{5}$$

由于 v_{i0}的位置和与 v_{i0}相关的三角形所在平面的方程未知，所以 $\varepsilon_{i,2}$的计算比较复杂. 如图2所示，我们先计算点 v_{i1}到三角形$\{v_{i0}, v_{i5}, v_{i6}\}$所在平面的距离的平方 d. 令 $v_{i1} = [x_{i1} \quad y_{i1} \quad z_{i1} \quad 1]^{\mathrm{T}}$, $v_{i5} = [x_{i5} \quad y_{i5} \quad z_{i5} \quad 1]^{\mathrm{T}}$, $v_{i6} = [x_{i6} \quad y_{i6} \quad z_{i6} \quad 1]^{\mathrm{T}}$. 则三角形$\{v_{i0}, v_{i5}, v_{i6}\}$所在平面的方程为

$$\begin{vmatrix} x & y & z & 1 \\ x_{i0} & y_{i0} & z_{i0} & 1 \\ x_{i5} & y_{i5} & z_{i5} & 1 \\ x_{i6} & y_{i6} & z_{i6} & 1 \end{vmatrix} = 0. \tag{6}$$



(a)原始网格 (b)顶点聚类后的网格

图 2 顶点聚类示意图

设

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

则有

$$d = \frac{\begin{vmatrix} x_{i1} & y_{i1} & z_{i1} & 1 \\ x_{i0} & y_{i0} & z_{i0} & 1 \\ x_{i5} & y_{i5} & z_{i5} & 1 \\ x_{i6} & y_{i6} & z_{i6} & 1 \end{vmatrix}^2}{a_{i0}^2 + b_{i0}^2 + c_{i0}^2} = \frac{\begin{vmatrix} x_{i0} & y_{i0} & z_{i0} & 1 \\ x_{i1} & y_{i1} & z_{i1} & 1 \\ x_{i5} & y_{i5} & z_{i5} & 1 \\ x_{i6} & y_{i6} & z_{i6} & 1 \end{vmatrix}^2}{a_{i0}^2 + b_{i0}^2 + c_{i0}^2}. \tag{7}$$

v_{i1}, v_{i5}和 v_{i6}已知，令 $p = [a \quad b \quad c \quad d]^{\mathrm{T}}$ 表示三角形$\{v_{i1}, v_{i5}, v_{i6}\}$所在平面方程 $ax + by + cz + d = 0$，且有 $a^2 + b^2 + c^2 = 1$. 令 $M_p = pp^{\mathrm{T}}$，同时令

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

就不难得到 v_{i0}到三角形$\{v_{i1}, v_{i5}, v_{i6}\}$所在平面的距离的平方为

$$v_{i0}^{\mathrm{T}}M_p v_{i0} = \frac{\begin{vmatrix} x_{i0} & y_{i0} & z_{i0} & 1 \\ x_{i1} & y_{i1} & z_{i1} & 1 \\ x_{i5} & y_{i5} & z_{i5} & 1 \\ x_{i6} & y_{i6} & z_{i6} & 1 \end{vmatrix}^2}{a_{i1}^2 + b_{i1}^2 + c_{i1}^2}. \tag{8}$$

由式(7),式(8)可以推导出

$$d = v_{i0}^{\mathrm{T}}\left(\frac{a_{i1}^2 + b_{i1}^2 + c_{i1}^2}{a_{i0}^2 + b_{i0}^2 + c_{i0}^2}M_p\right)v_{i0} = v_{i0}^{\mathrm{T}}M_p^* v_{i0}^{\mathrm{T}}. \tag{9}$$

由于 v_{i1},v_{i5}和 v_{i6}已知,M_p^* 只与 v_{i0}的位置有关.我们再考虑 v_{i1}到其它与 v_{i0}相关的三角形所在平面的距离平方以及 V_s 中的其余顶点到与 v_{i0}相关的三角形所在平面的距离平方,并对这些距离平方求和就可以得到

$$\varepsilon_{i,2} = v_{i0}^{\mathrm{T}}(\sum M_p^*)v_{i0} = v_{i0}^{\mathrm{T}}Q_{i,2}v_{i0}. \tag{10}$$

$Q_{i,2}$只与 v_{i0}的位置有关,v_{i0}未知,所以 $Q_{i,2}$也未知;求 v_{i0}的过程实际上也就是求 $Q_{i,2}$的过程.对此我们采用迭代法来求解:首先赋给 $Q_{i,2}$以迭代初值(零矩阵)得到 Q_i 的初值 $Q_{i,1}$,由式(3)求得 v_{i0}的近似值 v_{i0}^*;再由 v_{i0}的近似值 v_{i0}^* 求得 $Q_{i,2}$的近似值 $Q_{i,2}^*$,然后求得 v_{i0}的下一个近似值 v_{i0}^*.这个迭代过程一直继续到 v_{i0}的前后两个近似值之间的差别小于某个控制精度.至于这种迭代法的收敛性,由于初值选择得较好,收敛得相当快,实验证明迭代次数到三次时,结果已经很好了.这样我们就可以求得每次边折叠操作新生成的点的位置和该操作带来的误差.

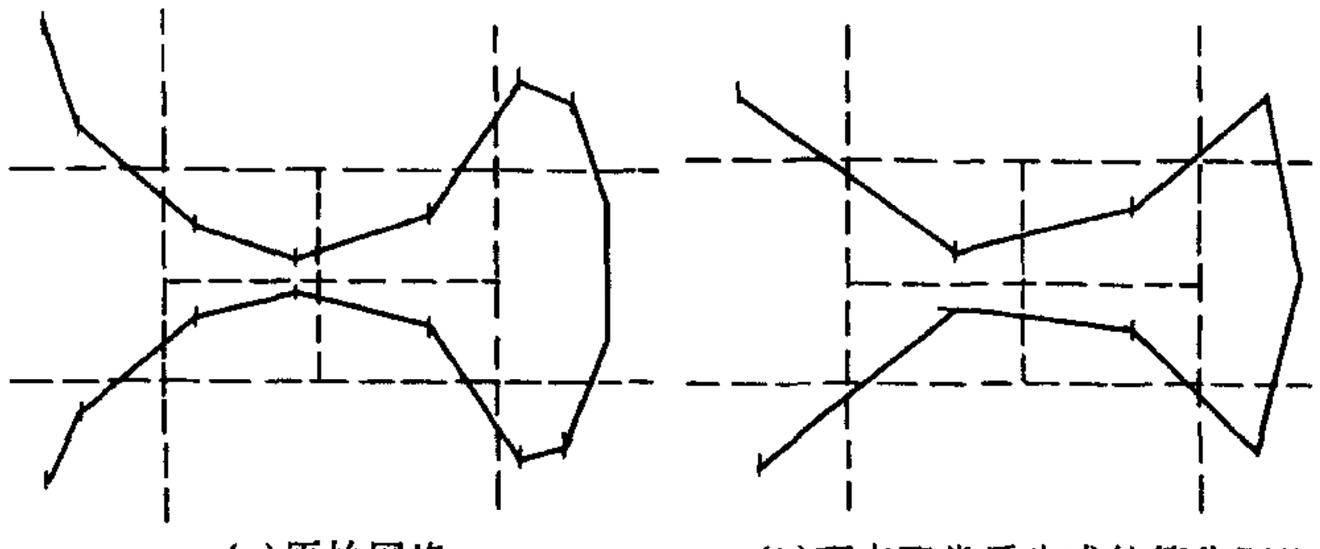
如果顶点集合 V_s 中含有边界顶点,为了保持原网格模型的边界特征,我们在三角形集合 $T_s(V_s)$的每条边界边上作一个通过该边界边并与该边界边所在的边界三角形垂直的平面.在计算误差矩阵时把这些平面也考虑在内.实践证明这种方法对保持原始网格的边界特征是非常有效的.

2.3　网格模型的自适应性划分

现在来考虑如何对原网格模型的包围盒进行划分.我们的基本工具是八叉树.原网格模型的包围盒作为八叉树的根节点,然后按顺序把原网格的所有顶点插入八叉树.本文算法是通过控制八叉树每个叶子节点的性质来实现网格模型的自适应性划分.

当一个顶点插入八叉树的某一叶子节点时,如果该叶子节点内还不包含顶点,则把该顶点直接放入该节点,不用做其它操作;如果该叶子节点内已经含有顶点,那就要判断该叶子节点所表示的长方体的尺寸是否大于用户指定的每个长方体的最大尺寸以及该节点所包含的顶点数目是否大于用户指定的每个长方体最多能包含的顶点数目,如果是,则需要分裂该节点并把该节点包含的顶点分配到其子节点中去;否则,为了保证2.2节中提到的每个长方体内的顶点和边所构成的图是连通的条件和避免图1所示的变形过大的聚类,算法还要判断在把该顶点放入该长方体后,会不会导致与新顶点集合相关的三角形集合中某些三角形的朝向(法向量)相反,如果会,该节点也要被分裂.

正是通过上面构造八叉树的过程,算法实现了网格模型的自适应性划分,也使得2.2节中描述的误差控制方法更为有效.图3是我们的自适应划分对图1所示的特殊网格划分聚类后的效果,实线表示网格,虚线表示划分线.注意到图3(b)所示的简化基本保持了图3(a)所示的原网格的形状.



(a)原始网格 (b)顶点聚类后生成的简化网格

图 3 本文算法对特殊网格的简化

2.4 算法总结

本文提出的基于顶点聚类的网格简化算法可以用下面的算法来描述:

算法1. 基于顶点聚类的网格简化算法

步骤1. 对原始网格按2.2节描述的方法建立初始八叉树划分.

步骤2. 对上一步建立的八叉树的每个非空叶子节点,计算该节点表示的长方体中包含的顶点集合的误差矩阵和聚类生成的新点的坐标以及带来得误差. 如果该误差在用户指定的范围内,则进行聚类;否则,对该节点再进行分裂,直到满足误差要求再执行聚类.

步骤3. 结束.

3 数据结构与实验结果

3.1 数据结构

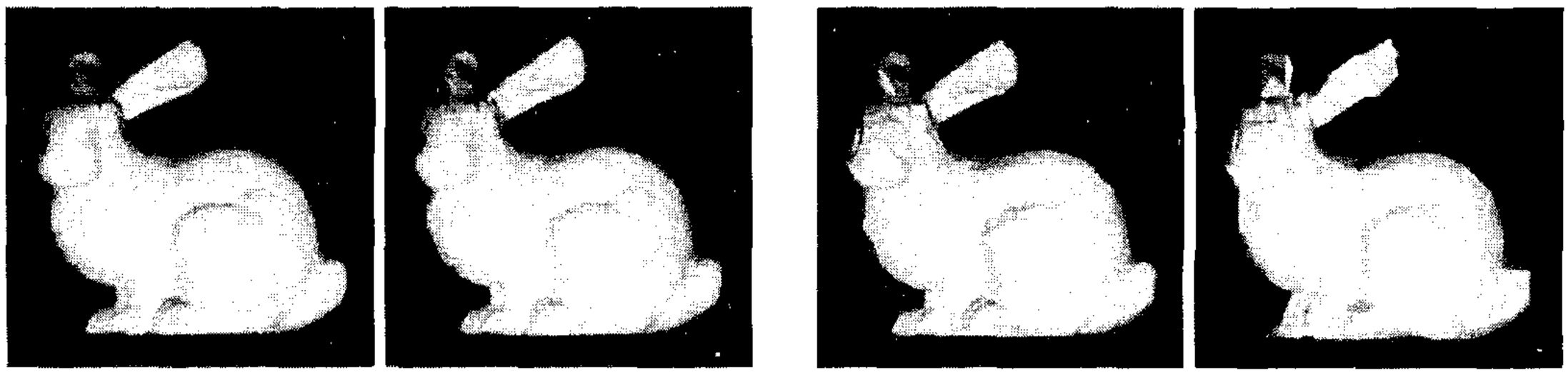
由于算法要处理的网格含有大量的三角形,因此要设计合理的数据结构,使得算法能处理数据量大的模型,而且要有较快的速度. 本算法所用到的数据结构有:(a)顶点表;(b)三角形表;(c)划分网格模型的八叉树. 对于每次聚类得到的新点坐标可以直接放入原顶点表,覆盖原来的表项,节省空间.

3.2 实验结果

该算法已在 IBM Power PC 上实现,图4,5,6是一组实例. 实践表明,该算法对较光滑和不光滑的模型数据都是比较有效的. 在图4中,兔子的模型在简化84.7%之后与原模型还很接近,当简化率为98.8%时,所得到的简化模型还是可接受的. 图5中的人头网格模型,最大简化率只有76.6%,这是由于原始网格数据量不大的缘故. 另外,本文算法也特别适用于三角化的地形模型,如图6. 图6(a)给出的是一个地形的原网格模型,图6(b),(c),(d)分别是其简化49.5%、74.8%、93.9%后的简化模型,注意到地形的边界特征保持得很好.

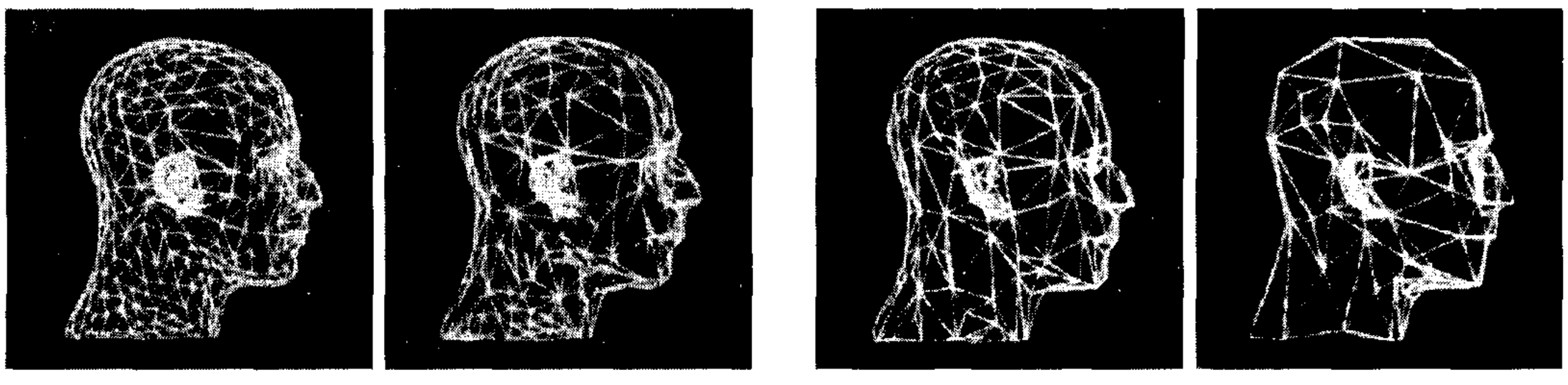
图7是 Rossignac[4]算法对兔子模型所做的简化,注意到图7(b)的简化模型已有很大的变形. 另外,本文算法中的误差控制方法与 Garland[7]所给出二次误差类似. Garland[7]提出的算法对网格进行简化的基本操作是顶点对折叠,即让原网格中构成一条边的顶点对或者距离较近的顶点对合并成一个顶点. 实际上,这是本文算法的一个特例,我们只要对网格模型的八叉树划分做更严格的限制,同时对相邻的八叉树节点也进行聚类操作就可实现 Garland[7]算法. 与该算法相比,本文算法的速度要快得多,而效果相当. 图8是用

Garland[7]算法对地形数据所做的简化. 表1给出了三种算法的速度对比. 所有的数据都是在 IBM Power PC 上测得的.



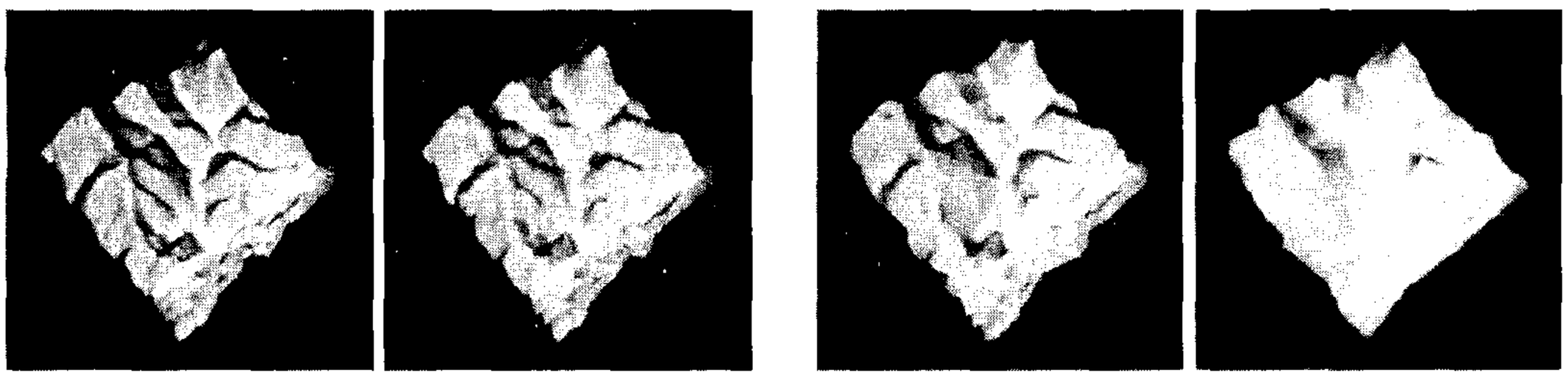
(a)原始模型 (69473个三角形)　(b)简化模型1 (10609个三角形,简化84.7%)　(c)简化模型2 (2682个三角形,简化96.1%)　(d)简化模型3 (852个三角形,简化98.8%)

图 4　本文算法对兔子模型的简化



(a)原始模型 (1355个三角形)　(b)简化模型1 (900个三角形,简化33.6%)　(c)简化模型2 (585个三角形,简化56.8%)　(d)简化模型3 (317个三角形,简化76.6%)

图 5　算法对人头模型的简化



(a)原始模型 (8192个三角形)　(b)简化模型1 (4134个三角形,简化49.5%)　(c)简化模型2 (2057个三角形,简化74.8%)　(d)简化模型3 (503个三角形,简化93.9%)

图 6　本文算法对地形模型的简化(采用了 Gouraud 明暗处理)

(a)简化模型1 (12505个三角形,简化82.0%)　(b)简化模型2 (2898个三角形,简化95.8%)

图7　Rossignac 算法对兔子模型的简化

(a)简化模型1 (1874个三角形,简化77.1%)　(b)简化模型2 (543个三角形,简化93.4%)

图 8　Garland 算法对地形模型的简化(采用了 Gourand 明暗处理)

表1 三种算法运行时间对比

原模型（三角形数）	简化模型（三角形数）	Rossignac 算法所用时间(s)	Garland 算法所用时间(s)	本文算法所用时间(s)
人头(1355)	597	0.31	1.48	0.52
地形(8192)	2057	2.33	31.02	4.56
兔子(69473)	2772	20.60	1738.87	44.02

4 结束语

本文提出的新的基于顶点聚类的网格简化算法简单明了、速度快且效果好.当网格模型为多边形模型时,该算法不受影响.因而该算法可用于虚拟现实和交互式可视化中的多细节层次场景的自动生成.

本文算法比起文献[4]中的算法,给出了一种误差控制方法从而能有效的控制简化网格与原始网格之间的误差,同时能对网格模型进行有效的自适应性划分;具有简化率高,简化效果好且速度相当的优点.

今后的研究主要是两个方面:1)研究更有效的误差控制方法;2)研究顶点聚类操作的逆操作(顶点分裂),以构造递进网格.另外,我们还想把该算法与视点和绘制算法结合起来,形成与视点相关的实时简化算法.因为该算法对网格模型建立了八叉树划分,我们可以根据视点的位置,对网格模型的不同部分采用不同的简化误差,得到更快的处理速度.

参 考 文 献

1 Schroeder W J,Zarge J A *et al*. Decimation of triangle meshes. *Computer Graphics*,1992,**26**(2):65—70

2 Turk G. Re-tiling polygonal surface. *Computer Graphics*,1992,**26**(2):55—64

3 Hoppe H,DeRose T *et al*. Mesh optimization. *Computer Graphics*(*SIGGRAPH*'93),1993,**27**:19—26

4 Rossignac J,Borrel P. Multi-resolution 3D approximation for rendering complex scenes. In:Falcidieno B,Kunii T. editors,Geometric Modeling in Computer Graphics. New York:Springer Verlag,1993:455—465

5 Eck M,DeRose T *et al*. Multi-resolution analysis of arbirtary meshes. *Computer Graphics*(*SIGGRAPH*'96),1996,**30**:173—182

6 Cohen J,Varshney A,Manocha D *et al*. Simplification envelopes. *Computer Graphics*(*SIGGRAPH*'96),1996,**30**:119—128

7 Garland M,Heckbert P S. Surface simplification using quadric error metrics. *Computer Graphics*(*SIGGRAPH*'97),1997,**31**:209—216

8 潘志庚,马小虎,石教英.虚拟环境中多细节层次模型自动生成算法.软件学报,1996,**7**(9):526—531

9 周晓云,刘慎权.基于特征角准则的多面体模型简化方法.计算机学报(增刊),1996,**19**(9):217—223

10 陶志良,潘志庚,石教英.基于能量优化的网格简化算法及其应用.软件学报,1997,**8**(12):881—888

周 昆 男,1977年出生,现为浙江大学 CAD & CG 国家重点实验室硕士研究生.研究方向为虚拟现实、科学计算可视化.

潘志庚 男,1965年出生,博士,现为浙江大学 CAD & CG 国家重点实验室研究员.研究方向为分布式计算、虚拟现实、多媒体技术.