第34卷 第20期 计 算 机 工 程 2008年10月
Vol.34 No.20 Computer Engineering October 2008
·图形图像处理· 文章编号：1000—3428(2008)20—0238—03 文献标识码：A 中图分类号：TP391

基于PM算法的网格简化改进算法

陈立潮，夏少芳，成洪静，刘 佳
(太原科技大学计算机科学与技术学院，太原 030024)

摘 要：针对传统网格简化算法在对边界顶点和边界边、累进网格二义性以及网格拓扑关系有效保持等的处理所存在的不足进行了相应的改进，改进的网格简化算法能有效保持网格模型的形体特征，消除累进网格的二义性，提高网格简化质量。针对折叠误差进行排序问题，采用最小堆算法，提高算法的时间效率。实验结果表明，该算法能产生高质量的网格，具有较高的执行效率。

关键词：边折叠；点分裂；二次误差测度；网格简化

Improved Mesh Simplification Algorithm Based on PM Algorithm

CHEN Li-chao, XIA Shao-fang, CHENG Hong-jing, LIU Jia
(Institute of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024)

【Abstract】Aiming at the deficiencies of the traditional mesh simplification algorithms in the process of boundary vertexes and edges, ambiguity of progressive and keep mesh's topological properties, the improvements are made. The improved algorithm can maintain the mesh physical characteristic effectively, eliminate the ambiguity of progressive mesh and improve the quality of mesh simplification. The use of the smallest stack to rank the folded error improves the time efficiency of the algorithm. Experimental results show that the improved algorithm can produce high-quality mesh and have very high efficiency in the implementation.

【Key words】edge collapse; vertex split; quadric error metrics; mesh simplification

1 概述

随着科学技术的进步，在计算机图形学、虚拟现实、计算机辅助设计技术、地理信息系统、医学图像系统等领域所构造和使用的图形模型越来越精细、复杂，这些复杂的模型动辄就产生数以百万计的面片，对计算机的存储容量、处理速度、绘制速度、传输效率等提出了很高的要求。然而在很多情况下，高分辨率的模型并非总是必要的，模型的准确度以及需要处理的时间也要有一个折中，因此，必须用一些相对简单的模型代替原始模型，即模型的简化。简化对于几何模型的存储、传输、处理，特别是对实时绘制有着重要的意义[1]。

网格的简化由静态简化逐步发展为动态简化，其中，顶点聚类算法、区域合并算法、重新布点法、逐步求精法以及几何元素删除算法都属于静态简化算法[2-3]。这些算法只考虑模型自身的信息，不能恢复原来模型的信息，有些算法在简化过程中破坏了网格原来的拓扑特征。动态简化算法的出现则很好地解决了以上问题，其中最著名且被人们采用最多的算法是Hoppe 在SIGGRAPH'96 会议上提出的渐进网格(Progressive Mesh, PM)算法，但该算法也存在一些不足之处，本文对此提出了一种改进的PM算法。

2 PM算法

2.1 定义

定义1 边界边：该边所关联的2个三角形中有1个是不存在的。

定义2 点的相关边：所有以点v_i为顶点的边的集合称为点v_i的相关边。

定义3 点的相关三角形：所有以点v_i为顶点的三角形所组成的集合称为v_i的相关三角形。

定义4 边界点：如果一个顶点的一条相关边在边界上，称其为边界点。

定义5 半边界边：满足以下2种情况之一的即可称作是半边界边：(1)该边的2个顶点都为边界点，且2点间没有共同的相邻边界边；(2)该边的2个顶点都为边界点，且2个点有共同的相邻边界顶点。

2.2 PM算法的基本思想

PM 算法以边折叠(edge collaps)和点分裂(vertex split)为基本操作，通过不断地进行边折叠操作达到网格简化的目的，同时在简化过程中记录原顶点和新顶点位置以及顶点间连接关系的变动信息，从而生成一个由原始模型的最简化模型和一系列简化信息组成的PM表示模式。PM可以把任意拓扑网格表示为一种高效、无损且具有连续分辨率的编码。在实时绘制时，PM 算法通过逆向跟踪简化信息序列，对每条简化信息执行点分裂逆操作，可以逐步恢复所删除的模型细节，实时得到原始模型的连续精度的简化模型，是一种多分辨率几何简化的优秀算法。

2.3 PM算法主体描述

PM算法是通过边折叠和点分裂这 2 个互逆的操作来实现的。基网格M^n，也就是未简化的原始网格，通过一系列的边折叠操作就可以简化为所需的最简网格M^0，每进行一次边折叠操作会简化掉1个顶点和2个三角形。同时记录下边折

基金项目：山西省自然科学基金资助项目(20051044)
作者简介：陈立潮(1961－)，男，教授、博士，主研方向：模式识别，人工智能，数据挖掘；夏少芳、成洪静、刘 佳，硕士研究生
收稿日期：2007-12-04 **E-mail：**ftivy@163.com

叠过程的信息，以便进行点分裂操作来细化到所要的状态。上面的过程可以表述如下：

$$M^n \underset{vsplit_{n-1}}{\overset{ecol_{n-1}}{\rightleftarrows}} M^{n-1} \underset{vsplit_{n-2}}{\overset{ecol_{n-2}}{\rightleftarrows}} \cdots \underset{vsplit_0}{\overset{ecol_0}{\rightleftarrows}} M^0 \quad (1)$$

图1给出了网格简化过程中一次边折叠和点分裂的过程。

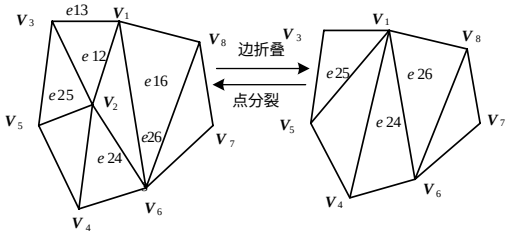


图 1　边折叠和点分裂示意图

在选择折叠边的时候，Hoppe 采用以下能量函数计算边折叠的代价并排序，然后按照排序后的折叠代价选择边的折叠，即先折叠代价小的边。Hoppe 的能量函数表示如下：

$$E(k,v) = Edist(k,v) + Erep(v) + Espring(v) \quad (2)$$

该能量方程虽能很好地保证简化后网格的质量，但其计算相当复杂，时间效率很低，对于有十几万甚至上百万个面片的网格简化来说，所需的时间是不可忍受的。同时，在边折叠时有可能破坏网格的拓扑关系，对于点分裂的过程，某些情况下也会产生二义性。

3　改进的 PM 算法

3.1　折叠误差的计算和新顶点的确定

在边折叠过程中有 2 个关键的问题：(1)边折叠操作的代价是多少，即这个操作带来的误差是多少；(2)如何选择新顶点$\boldsymbol{v}_{i0}$的位置，使得折叠操作误差尽可能小。因此折叠边的选择直接影响算法的效率和质量。本文采用Garland和Heckbert提出的二次误差测度方法进行边折叠误差的计算以及新顶点位置的确定，因为该方法简化的质量仅次于Hoppe的能量方程，却具有很高的计算速度和较小的内存消耗。

设e是待折叠的边，$\boldsymbol{v}_1$, $\boldsymbol{v}_2$分别为e的 2 个端点，$P(e)$为顶点$\boldsymbol{v}_1$和顶点$\boldsymbol{v}_2$所有相关三角形所在平面构成的平面集的并集，将e折叠到空间某一新位置$\boldsymbol{v}_{i0}=[x_{i0}, y_{i0}, z_{i0}, 1]$之后产生的　误差$Q_e(\boldsymbol{v}_{i0})$定义为新顶点$\boldsymbol{v}_{i0}$到 $P(e)$中诸平面的距离之平方和，即

$$Q_e(\boldsymbol{v}_{i0}) = \sum_{P\in P(e)} \left(d_p\right)^2 \quad (3)$$

其中，d_p为新顶点$\boldsymbol{v}_{i0}$到平面 p 的距离。设$\boldsymbol{v}_{i0}=[x_{i0}, y_{i0}, z_{i0}, 1]$是$\boldsymbol{v}_{i0}$的齐次坐标表示形式，$\boldsymbol{p}=[a,b,c,d]$表示三角形所在平面的平面方程：$ax+by+cz+d=0$，其中，$a^2+b^2+c^2=1$。由此式(3)可以表示为

$$Q_e(\boldsymbol{v}_{i0}) = \sum_{p\in p(v)} \left(\left(\boldsymbol{v}_{i0}^{\mathrm{T}} p\right)\left(\boldsymbol{p}^{\mathrm{T}} v_{i0}\right)\right) = \sum_{p\in p(v)} \left(\boldsymbol{v}_{i0}^{\mathrm{T}}\left(\boldsymbol{p}\boldsymbol{p}^{\mathrm{T}}\right)\boldsymbol{v}_{i0}\right) = \boldsymbol{v}_{i0}^{\mathrm{T}} \sum_{p\in p(v)} \left(M_p\right)\boldsymbol{v}_{i0} \quad (4)$$

其中，$\boldsymbol{M}_p$称为三角形的误差矩阵

$$\boldsymbol{M}_p = \boldsymbol{p}\boldsymbol{p}^{\mathrm{T}} = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix} \quad (5)$$

从式(4)可以看出，$Q_e(\boldsymbol{v}_{i0})$的计算依赖于新顶点 v_{i0} 的确定。本文选择折叠边$e(v_1, v_2)$2 个顶点中使 $Q_e(\boldsymbol{v}_{i0})$ 最小的其中一个顶点作为e被折叠后的新顶点 $\boldsymbol{v}_{i0}$，这样做有 2 个好处：简化了算法的复杂度，提高了效率；虽然在简化精度上做出了一点牺牲，但消除了累进过程的二义性，避免了多次点分裂时的判断。

3.2　边界边和边界顶点的改进

传统二次误差测度方法对于开放边界的网格会产生很大的误差，所以，本文对边界边和边界顶点提出以下改进方案：

(1)一个顶点是边界点的边的折叠。对此本文直接将边界点作为折叠后的新顶点的位置。这种方法既简单又能很好地保留网格的整个形体特征。

(2)边界边的折叠。对此文献[4-5]都作了考虑，却都选择了不折叠的方式，但是采用这种方式对于拥有边界比较明显的网格，如地形模型的简化，就会严重影响网格的简化程度。

考虑到边界边能够较好地保存网格的形体特征并且拥有较多的细节特征，本文提出一种“延迟简化”的方式，即为网格边界边的误差值加上一个阈值C_0，该阈值的选择使得边界边的折叠迟于其他边的折叠。这样既提高了网格简化的程度又使边界特征能有效保留。

3.3　拓扑结构的处理

对于上述半边界边的情况，可以看出当折叠边(v_1,v_2)后网格的拓扑关系将产生错误。如图 2 所示 2 种情况，折叠后新顶点会拥有 4 条边界边，发生拓扑错误。

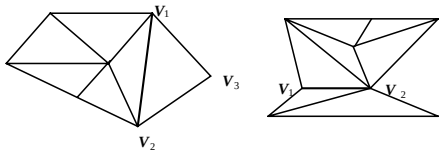


图 2　相邻边界边规则的半边界

以图 3 为例，图 3 右边所示网格内部有空洞特征的情况，当直接进行边折叠后会如图 3 左边所示，空洞消失。

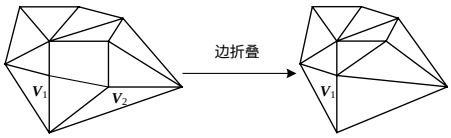


图 3　相邻边界顶点规则的半边界

对于以上 2 种折叠后会产生拓扑错误的情况，本文采用不折叠的方式，这样处理既保证了网格拓扑关系的正确性，又避免了复杂的判断。边折叠作为简化的基本单元操作，如果处理很复杂，会大大降低简化的效率。

在边折叠过程中还经常遇到一种情况：如图 4 所示，当顶点v_1和v_2拥有 3 个或 3 个以上的共同相邻顶点时，边e折叠以后，顶点v_2折叠到v_1，面f_1和f_2，f_3和f_4就会重合在一起，边e_1折叠到e_2上，面f_3 f_4 f_7 f_8共用一条边e_2。而对面片拓扑结构的要求是一条边最多只能和 2 个面相邻，因此，这样的折叠会使网格的拓扑关系产生错误。对于这种情况的处理，本文采用类似文献[4]提出的方法，即保证图 5 中虚线的部分比外面实线部分先折叠。

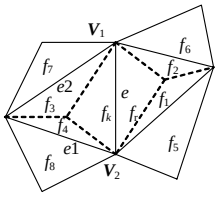


图 4　有 3 个或 3 个以上相邻顶点的情况

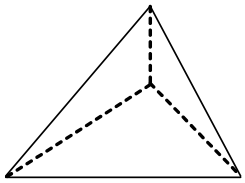


图 5　多个相邻点拓扑关系解决方案

3.4　折叠误差的排序

折叠误差的排序问题也是影响算法效率的关键因素之一，虽然链表可以完成排序、存储、按大小顺序插入和删除操作等任务，但是算法的效率很差，一般完成这些操作需要遍历整个链表，时间复杂度最坏情况为 $O(n^2)$，因此，本文采用最小堆来实现对折叠误差的排序以及插入和删除等操作。最小堆是节点间具有层次次序关系的完全二叉树，构造最小堆的原则是：双亲值小于或等于孩子的值，堆顶即树根，具有最小的数据值，如图 6 所示。

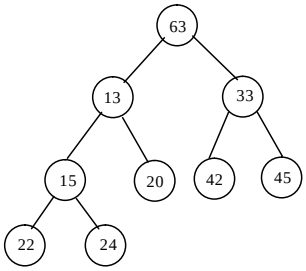


图 6　最小堆示意图

最小堆允许插入和删除操作，插入数据的过程就是依次与各个节点的数值进行比较定位的过程，删除操作总是删除堆顶的最小值，删除堆顶数据后为了保证最小堆原则，需要进行反复筛选重新调整为最小堆，这个过程称为最小堆排序。本文通过将元素插入到最小堆中并通过反复删除根节点进行折叠操作，然后重新计算与新顶点相关的所有边的折叠误差，并插入到堆中对它们进行排序。使用最小堆的具体的操作内容有：

(1)按折叠误差大小顺序建立最小堆。

(2)选择堆顶部折叠误差最小的边进行折叠，并生成新的边的折叠误差，按大小顺序插入到堆中。

(3)在堆中删除与折叠边的一个点相关的所有记录，更新所有记录，即对堆中的数值按最小堆形成原则重新排序，为下一次折叠做准备。

利用最小堆来实现 n 个元素的排序，时间复杂度仅仅为 $O(n\mathrm{lb}n)$，大大提高了算法的效率。

3.5　算法的主要步骤描述

改进算法的主要步骤描述如下：

(1)初始化操作

1)读入待简化的 delauny 三角网的相关数据；

2)求出每个顶点与其相邻的所有顶点的编号；

3)求出与每个顶点相关的所有三角形；

4)求出每个顶点的相关边的编号。

(2)计算原始网格中每个三角形的误差矩阵。

(3)计算每条边的折叠误差，并判断该边是否为边界边，若是，则将其折叠误差 $Q_e(\boldsymbol{v}_{i0})$ 加上阈值 C_0，否则不变。

(4)将每条边的折叠误差、折叠后的顶点坐标插入到最小堆中，同时记录下每条边相应 2 个顶点的序号。

(5)选择堆顶部折叠误差最小的边进行折叠。

(6)判断折叠该边后是否产生拓扑错误，若是，判断是哪种拓扑错误，并进行相应的操作，然后转步骤(5)，否则直接转步骤(7)。

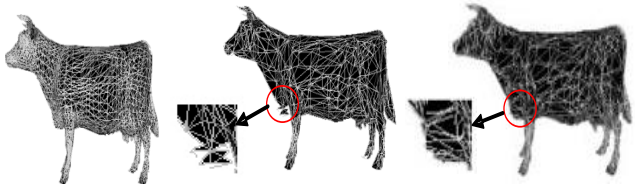
(7)对该边进行折叠操作，将折叠边的其中一个顶点折叠到误差较小的另一个上，作为新顶点，并记录折叠后的对应关系。

(8)对边进行折叠后，将与该边相关联的边都从堆中删除，重新计算与新顶点相关的边的折叠误差和坐标并插入到堆中进行排序。

(9)若边的序列为空或误差已经达到用户要求则算法结束，否则转步骤(5)。

4　实验结果

为了验证本文算法的有效性，实验由 VC++6.0 结合 OpenGL 编程，在 Intel Pentium 4 3.0 GHz CPU 环境下实现。图 7(a)给出了 cow 模型的原始网格模型；图 7(b)是直接利用 Garland 的折叠误差函数来实现的 PM 算法将原始网格简化 80%时的情形；图 7(c)是本文算法将原始网格简化 80%的情形。可以明显看出，本算法能够很好地保证模型的外形特征，而 Garland 的算法在牛的头部、胸部和尾部出现了明显的锯齿状，如局部被放大的部分所示。



(a)原始模型　(b)用 Garland 算法简化　(c)用本文算法简化

图 7　cow 三角网格模型的简化对比

由于本文对折叠边的误差排序、新顶点的选择以及拓扑关系处理进行了改进，因此大大提高了算法的效率，数据越大时效果越明显。表 1 示出了分别用 Garland 和本文算法实现的网格简化的时间消耗情况。

表 1　2 种算法时间消耗的比较

模型名称	面片相应变化	面片简化数	本算法简化时间/s	Garland 算法简化时间/s
cow	5 804~906	4 898	3.362	4.812
big_porsche	10 474~1 594	8 880	6.821	10.656
bunny	69 451~6 806	62 645	48.722	92.088
地形模型	36 820~1 245	35 575	26.282	50.516

5　结束语

本文在网格简化的过程中采用改进的二次误差测度方法进行边折叠误差的计算并进行新顶点位置的确定，消除了累进过程的二义性，提高了效率。对于边界边的处理在误差的计算过程中引入了新的阈值，实行“延迟简化”，有效地保持了网格的形体特征。对可能存在的拓扑错误进行了判断和处理，使得网格的拓扑保持特性得以维持。对折叠误差采用最小堆结构进行快速排序，提高了算法的时间效率。

实验结果表明，该方法简单易行，在提高网格简化效率的同时，保证了网格简化的效果和质量，做到了模型简化精度和处理时间上的很好折中。该算法可进一步运用到相关的简化中。

(下转第 246 页)