ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*, Vol.20, No.3, March 2009, pp.713−723
doi: 10.3724/SP.J.1001.2009.03304

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 支持外观属性保持的三维网格模型简化*

卢 威+, 曾定浩, 潘金贵

(计算机软件新技术国家重点实验室(南京大学),江苏 南京 210093)

## Mesh Simplification for 3D Models with Feature-Preserving

LU Wei+, ZENG Ding-Hao, PAN Jin-Gui

(State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China)

+ Corresponding author: E-mail: ddhansh@gmail.com

**Lu W, Zeng DH, Pan JG. Mesh simplification for 3D models with feature-preserving. *Journal of Software*, 2009,20(3):713−723.** http://www.jos.org.cn/1000-9825/3304.htm

**Abstract**: This paper analyzes current mesh simplification methods, and proposes an algorithm based on the quadric error metric (QEM) for feature preserving. It adopts a Half-edge collapse method for mesh simplification and modifies QEM to remove the discontinuities of appearance attributes. By analyzing the relationships between vertices and the discrete appearance seam, a new formula is obtained which enables the edge contraction to postpone the appearance; meanwhile a proper replacer is selected for the wedge in the triangle that has been affected by half-edge collapsing operation to avoid material distortion. Experimental results demonstrate that author's algorithm achieves a similar high efficiency as QEM with desirable geometry and feature-preserving.

**Key words**: mesh simplification; half-edge collapse; progressive mesh; the quadric error metric; feature preserving

**摘 要**: 对已有的三维网格简化技术进行分析,利用半边折叠操作对 QEM(quadric error metric)算法进行改进,提出了一种基于二次误差测度(QEM)的网格简化算法,解决了非连续外观属性在简化过程中的畸变问题.通过分析顶点与非连续外观接缝的关系,得出了一个新的边折叠代价公式,使得外观畸变在简化过程中尽可能地推迟;并且在执行半边折叠时给受影响的三角形找到了合适的替换 wedge,避免外观畸变的发生.实验结果表明,该算法保持了 QEM 算法的高效性,同时在几何属性和外观属性上都取得了令人满意的简化效果.

**关键词**: 网格简化;半边折叠;累进网格;二次误差测度;外观属性保持

**中图法分类号**: TP391 **文献标识码**: A

在计算机图形学中,三维几何模型通常用多边形网格进行描述[1].由于多边形网格在数学上的简单性,大多数图形系统(包括硬件和软件)都内置了针对多边形网格的简单而高效的渲染算法[2,3].因此,多边形网格通常被称为标准图形对象[3],被广泛应用于虚拟现实、计算机辅助设计、地理信息系统以及医学图像系统等领域.三维空间中任意 3 个点可以确定一个平面,因此三角形成为多边形网格表示中最常用到的多边形.同时,三角形在简化过程中具有灵活性和高效性,因此许多应用程序,尤其是网络应用程序,都将三角形作为网格模型的基本组成

* Supported by the National Natural Science Foundation of China under Grant Nos.60473113, 60533080 (国家自然科学基金)
Received 2007-10-19; Accepted 2008-03-14

单元[4].

在实际应用中,为了使三维空间中的物体具有更高的真实感和层次感,需要使用复杂的、细节化的网格模型进行表示.针对不同的应用场合,考虑到计算机储存容量、处理速度、渲染速度、传输速率等因素的平衡折衷,可以选择一个合适的低分辨率模型来代替原始模型.这种通过对原始模型进行自动简化处理,生成一系列不同分辨率、不同精细度的模型以供渲染时使用的技术,被称为多细节层次(level of detail,简称 LoD)技术[5],生成的模型称为多细节层次模型或多分辨率模型.

现有大量的网格模型简化算法可以生成连续的、多分辨率的模型,但都无法保证模型中的非连续外观属性不出现畸变.因此,我们对网格模型上顶点和非连续外观接缝的关系进行分析,推导出新的折叠边代价计算公式,使得此类畸变的发生尽可能地推迟,保持绝大多数模型外观属性不变形.

## 1 相关工作

已有的大部分针对三角形网格模型的简化算法[6]都以边折叠(edge collapse)和点分裂(边折叠的逆操作)作为基本的网格简化操作[7−9].半边折叠(half edge collapse)作为边折叠的退化版本,在执行折叠操作时不引入新的顶点,而是以折叠边上的一个端点作为折叠的结果[10],优点在于减少了内存占用,且有利于构建渲染系统可以直接处理的数据结构,提高了渲染效率.因此,本文采用半边折叠作为基础迭代操作,且定义一个折叠过程为 $v_s \to v_t$,其中 $v_s$ 为折叠起点,$v_t$ 为折叠终点.

边折叠关键要解决两个问题,即如何选取折叠边和如何确定代替边上新顶点的位置.Hoppe 采用能量优化的方法来选择边并确定新顶点的位置[8].该算法要求建立和求解复杂的全局能量优化方程,计算量大,很难满足实时要求,但是生成模型的效果却是所有简化算法中最好的[11].Garland 和 Heckbert 提出的二次误差测度(quadric error metric,简称 QEM)解决了 Hoppe 方法计算量大的问题,该算法使用二次误差度量来控制简化过程,在速度、保真度、健壮性上获得了很好的平衡[12].针对基于几何元素删除型算法的相似性,Kobbelt 等人抽象出基于QEM算法的一般框架,并从算法分析的角度分析了各种基于几何元素删除的算法,认为这些算法的本质上是贪婪算法,不同之处在于删除的元素以及选取待删除元素的标准[10].

事实上,网格模型的顶点除了几何坐标外,还具有外观属性,如颜色、纹理坐标、法向量等.Garland 在文献[13]中扩展了 QEM 算法,使其支持带有外观属性模型的简化.Hoppe 也以 QEM 为基础提出了处理顶点外观属性的方法,其储存空间比 Garland 的方法更少[9].但是,这两种方法都基于一个假设:网格模型顶点上的外观属性是连续的,都无法处理突变顶点的外观属性,例如纹理的接缝.因此,我们提出一种新的基于 QEM 的网格简化算法并定义新的折叠代价计算公式,有效地解决了外观属性出现的突变问题.实验结果表明,对于具有非连续外观属性的模型,该算法在保持了 QEM 算法高效率的同时,也获得了很好的保真度.

## 2 基于 QEM 的改进网格模型简化算法(M-QEM)

### 2.1 二次误差测度的基本框架

在三维空间中,一个平面的向量方程为 $n^Tv+d=0$,其中 $n=[a,b,c]^T$ 且 $a^2+b^2+c^2=1$,为平面的单位法向量;$d$ 为距离常量.当一个边折叠操作完成后,折叠边上的两个顶点变成了一个新的顶点,该顶点的相关平面集是两个旧顶点相关平面集的并集.对于新顶点所引起的误差,Garland 改进了 Plane-Based error metric 算法[14],提出 quadric 方法[12],将其定义为该顶点到相关平面距离的平方之和.因此,任一平面的 quadric $Q=(A,b,c)=(nn^T,dn,d^2)$,其中 $A$ 为一个 3×3 的对称矩阵,$b$ 为一个 3×1 的向量,$c$ 是距离常量;任意顶点 $v$ 到平面的距离的平方可以用 quadric 方式表示为 $Q(v)=D^2(v)=v^TAv+2b^Tv+c$.由于 $Q(v)$的加法满足分配率[12],因此一个平面集的 quadric 可以通过将各个平面的 quadric 相加而得.

根据以上关于 quadric 的定义和性质,我们在计算某个折叠边的代价时,无须为新顶点计算其到一个平面集的距离的平方和,而只需为每个顶点保存一个 quadric 即可.由于本文采用半边折叠的方法,新顶点 $v_n$ 位置的选

择相对简单很多:设 $Q=Q_s+Q_t$,当 $Q(v_s)>Q(v_t)$时,$v_n$ 即为 $v_t$;反之,则为 $v_s$.半边折叠 $v_s\rightarrow v_t$ 的折叠代价 $Cost(v_s,v_t)$为

$$Cost(v_s,v_t)=(Q_s+Q_t)v_t \tag{1}$$

### 2.2 顶点外观属性的处理

#### 2.2.1 预备知识

具有外观属性的顶点可能对应不止一个属性值,例如一个顶点在不同的三角形中具有不同的颜色,因此,Hoppe 提出了 wedge 的概念来标识顶点[15].wedge 是一个 $n$ 元组$(a_0,a_1,a_2,\ldots,a_{n-1})$,元素 $a_i$ 为顶点的属性,且 $n>0$;其中 $a_0$ 为顶点坐标属性,记为 *coord*,其他属性根据网格模型各异.在大多数模型中,wedge 为一个四元组(*coord*,*normal*,*color*,*texCoord*):$coord\in R^3$,为顶点坐标;$normal\in R^3$,为顶点的法向量;$color\in RGB$,为顶点的颜色;$texCoord\in R^2$,为顶点的纹理坐标.一个顶点可能对应多个 wedge,而一个 wedge 只能对应一个顶点.当某个顶点对应多个 wedge 时,这些 wedge 的 *coord* 属性必须是一样的.当 $n=1$ 时,模型的 wedge 只有 *coord* 属性,退化成为基本形式的顶点.

具有外观接缝的模型,其接缝处的顶点通常对应两个或以上的 wedge,每个 wedge 上的外观属性均不同.我们定义接缝处的顶点为接缝点,接缝处的边为接缝边;不在接缝处的顶点为内部点,两个端点皆为内部点的边,为内部边.根据纹理映射机制,三角形面片内的纹理坐标是对 3 个顶点的纹理坐标进行插值而得,所以内部点之间的折叠不会引起可察觉的视觉差异.因此,对于内部点本文将沿用基本的 QEM 算法,不进行外观属性扩展.实验结果证实该想法是正确的.

#### 2.2.2 计算边折叠代价

对于带有接缝点的半边折叠 $v_s\rightarrow v_t$,我们需要解决两个问题:一是如何计算边折叠代价,二是如何选择折叠后受影响三角形的顶点的外观属性.根据 $v_s$ 和 $v_t$ 是否为接缝点,可以分为以下几种情况进行讨论:

第 1 种情况,起点 $v_s$ 为接缝点,终点 $v_t$ 为内部点,如图 1 所示,$(v_l,v_s)$和$(v_s,v_r)$均为接缝边,$v_l$,$v_s$ 和 $v_r$ 为接缝点,各对应两个纹理坐标.在执行半边折叠操作后,三角形$(v_l,v_s,v_t)$和$(v_s,v_r,v_t)$被删除,受影响的三角形$(v_l,v_r,v_s)$上的顶点 $v_s$ 被替换为 $v_t$.由图 1 可知,此时 $v_t$ 的纹理坐标保持不变,唯一地对应于(0.5,1),但与 $v_s$ 原本的纹理坐标(0.5,0.25)不连续,因此接缝处的纹理会出现外观畸变.在实际应用中,纹理坐标更为复杂,如果任由这种半边折叠操作发生,那么在网格模型的接缝处可能会出现畸变的外观.图 2 所示的原始矿石模型具有 5 000 个三角形,在不约束此类半边折叠操作的情况下,当简化到分辨率为 460 个三角形时(如图 3 所示),在模型接缝处就明显出现了畸变.即使我们放宽半边折叠的限制,引入新的 wedge 以保持折叠后的纹理不变,但接缝却变形了.这在有些情况下是不可接受的,比如接缝作为模型表面分界线的时候.

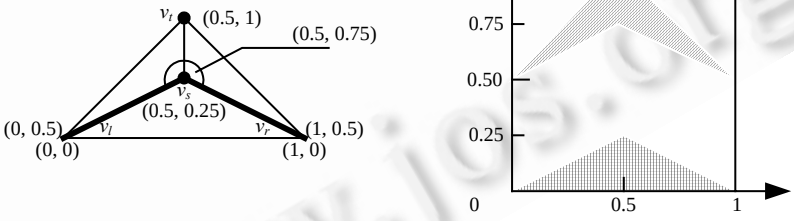



Fig.1 Texture coordinates of a triangle

图 1 三角形顶点的纹理坐标

Fig.2 An emerald model with textural seams

图 2 一个带有纹理接缝的绿柱石模型

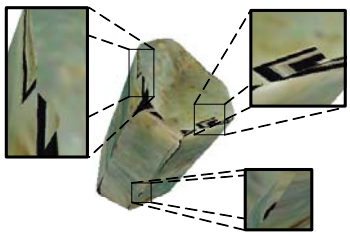


Fig.3 Aberrances on the emerald model

图 3 绿柱石模型在接缝处的异常外观

因此,在模型简化时,我们应尽量避免这种起点为接缝点而终点为内部点的半边折叠操作.为此,我们加大折叠代价,使此类操作尽量靠后.式(1)计算半边折叠代价的公式被重新定义为

$$Cost(v_s,v_t)=Dist(v_s,v_t)+\alpha\cdot IsSeam(v_s)\cdot(1-IsSeam(v_t)) \quad (2)$$

其中,$Dist(v_s,v_t)=(Q_s+Q_t)v_t$,函数 $IsSeam(v)$定义为

$$IsSeam(v)=\begin{cases}1,\text{若}v\text{为接缝点}\\0,\text{若}v\text{为内部点}\end{cases},$$

$\alpha$是一个足够大的数,与 $Dist(v_s,v_t)$相比具有绝对优势,保证接缝点到内部点的折叠不会过早执行.

以上讨论表明,在执行一次半边折叠操作后,对于受影响的三角形中的 wedge,如果有连续的 wedge 与之替换,那么接缝处的外观属性就不会发生畸变;反之则很可能产生类似图 3 的奇怪外观.如果无法找到合适的 wedge 来替代受影响的三角形中消失的 wedge,那么该半边折叠操作的代价应当尽量地大.

第 2 种情况,起点 $v_s$ 为内部点,终点 $v_t$ 为接缝点,由于接缝点保持不动,所以不会出现接缝形变.在对受影响的三角形进行顶点替换时,如果三角形顶点上的 wedge 均找到了与之连续的 wedge 进行替换,那么接缝的外观就不会发生畸变,因此,问题转变成如何选择合适的 wedge.如图 4 所示,接缝边$(v_l,v_t)$和$(v_t,v_r)$将模型表面分成两部分:包含 $v_s$ 的内部和不包含 $v_s$ 的外部,三角形也分为内部三角形和外部三角形.在执行 $v_s\to v_t$ 之后,由于 $w_s$ 与 $w_0$ 是连续的,所以受影响的 3 个三角形上的 $v_s$ 所对应的 $w_s$ 均被替换为 $w_0$.

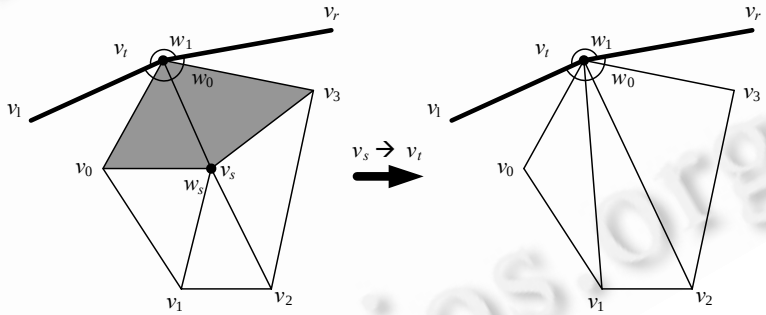



Fig.4 Half-Edge collapsing from an internal point to a seam point

图 4 内部点到接缝点的折叠

那么如何判断 $v_t$ 对应的两个 wedge,哪个是 $w_0$,哪个是 $w_1$?我们必须从网格的几何结构入手,找出与 $w_s$ 连续的 wedge.由于三角形$(v_3,v_t,v_s)$与折叠顶点 $v_s$ 和 $v_t$ 都相邻,且在未发生畸变之前三角形上的 wedge 是连续的,所以其上的顶点 $v_t$ 所对应的 wedge 为 $w_0$.

第 3 种情况,起点 $v_s$ 和终点 $v_t$ 均为接缝点.这种情况较为复杂,可细分为 4 个子类型:

(1) $v_s$ 和 $v_t$ 在同一条接缝上,如图 5(a)所示.这种类型与第 2 种情况类似,关键问题也是如何选择合适的替换 wedge.由图可知,$v_s$ 对应 $w_0$ 和 $w_1$,$v_t$ 对应 $w_2$ 和 $w_3$,其中 $w_0$ 和 $w_2$ 同属上半部分,是连续的;$w_1$ 和 $w_3$ 同属下半部分,也是连续的.执行 $v_s\to v_t$ 后,$w_0$ 替换为 $w_2$,$w_1$ 替换为 $w_3$,即可确保不发生外观畸变.

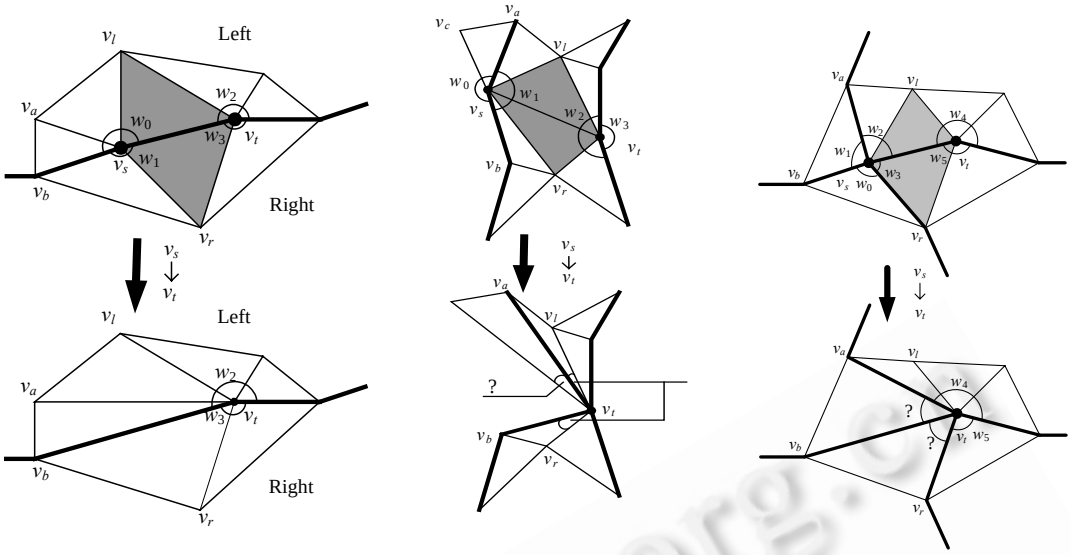



Fig.5 Half-Edge collapsing from a seam point to a seam point

图 5 接缝点到接缝点的折叠

(2) $v_s$ 和 $v_t$ 分属两条不同的接缝,均为接缝点,但是折叠边$(v_s,v_t)$不是接缝边.如图 5(b)所示,$v_s$ 对应的 wedge 为 $w_0$ 和 $w_1$,$v_t$ 对应的 wedge 为 $w_2$ 和 $w_3$,且只有 $w_1$ 和 $w_2$ 是连续的.在执行 $v_s \to v_t$ 后,三角形$(v_s,v_l,v_a)$和$(v_s,v_b,v_r)$上的顶点 $v_s$ 对应的 $w_1$ 可被替换为 $w_2$,但是三角形$(v_s,v_a,v_c)$上的 $v_s$ 对应的 $w_0$ 找不到合适的 wedge 与之替换.因此,接缝 $v_a$-$v_s$-$v_b$ 处的外观属性会出现畸变,而且即使该接缝处的外观属性没有发生畸变,而其几何形状也变形了,所以,此类折叠操作应尽量靠后.我们以式(2)为基础将折叠代价重新定义为

$$Cost(v_s,v_t)=Dist(v_s,v_t)+\alpha\cdot IsSeam(v_s)\cdot(1-IsSeam(v_t))+\beta\cdot IsSeam(v_s)\cdot IsSeam(v_t)\cdot(1-IsSeam(v_s,v_t)) \quad (3)$$

参数$\beta$和$\alpha$一样,是一个比 $Dist(v_s,v_t)$大很多的数.函数 $IsSeam(v_s,v_t)$定义为

$$IsSeam(v_s,v_t)=\begin{cases}1, 若(v_s,v_t)为接缝边\\ 0, 若(v_s,v_t)为非接缝边\end{cases}.$$

边$(v_s,v_t)$是否为接缝边可以通过以下方法进行判断:首先找出与边$(v_s,v_t)$相邻的三角形,设其集合为 $Shared(v_s,v_t)$.对于流形表面而言,此类三角形不会多于两个.若$|Shared(v_s,v_t)|<2$,则$(v_s,v_t)$为边界,是接缝的一种特殊情况.若$|Shared(v_s,v_t)|=2$,则设相邻两个三角形为$(v_s,v_t,v_l)$和$(v_s,v_r,v_t)$,如果前者中的 $v_s$ 对应的 wedge 与后者中的 $v_s$ 对应的 wedge 不同,则$(v_s,v_t)$为接缝边,反之则否.

(3) $v_s$ 和 $v_t$ 分属两条接缝,均为接缝点,且$(v_s,v_t)$为接缝边.以图 5(c)为例,由于 $v_s$ 同时在接缝 $v_b$-$v_s$-$v_t$ 和 $v_a$-$v_s$-$v_r$ 上,我们称其为两条接缝的交叉点.此时,顶点 $v_s$ 对应 4 个 wedge:$w_0$,$w_1$,$w_2$ 和 $w_3$,顶点 $v_t$ 对应两个 wedge:$w_4$ 和 $w_5$,其中 $w_2$ 和 $w_4$ 连续,$w_3$ 和 $w_5$ 连续.执行 $v_s \to v_t$ 后,受影响的三角形$(v_a,v_s,v_l)$上的 $w_2$ 被替换为 $w_4$.$w_3$ 不为任何受影响的三角形所有,不必替换;即使 $w_3$ 属于某个受影响的三角形,也可以被替换为 $w_5$.另外,两个受影响的三角形$(v_a,v_b,v_s)$和$(v_b,v_r,v_s)$,其上的 $w_0$ 和 $w_1$ 在顶点 $v_t$ 处找不到连续的 wedge 与之替换,将导致接缝处模型的外观发生畸变,也会造成接缝 $v_a$-$v_s$-$v_r$ 形变.因此,对于此类折叠操作,我们也加大了折叠代价,使模型外观畸变不会过早发生.式(3)被重新定义为

$$\begin{aligned}Cost(v_s,v_t)=&Dist(v_s,v_t)+\alpha\cdot IsSeam(v_s)\cdot\big(1-IsSeam(v_s)\big)+\\&\beta\cdot IsSeam(v_s)\cdot IsSeam(v_t)\cdot\big(1-IsSeam(v_s,v_t)\big)+\gamma\cdot IsCross(v_s)\cdot IsSeam(v_t)\end{aligned} \quad (4)$$

与$\alpha,\beta$一样,$\gamma$也是一个比 $Dist(v_s,v_t)$大很多的数.函数 $IsCross(v)$定义为

$$IsCross(v)=\begin{cases}1, 若v为交叉点\\ 0, 若v为非交叉点\end{cases}.$$

(4) 从非交叉接缝点到交叉点的折叠,恰好与图 5(c)所示相反,为 $v_t \to v_s$.由于 $w_4$ 和 $w_2$ 是连续的,受影响的三角形上的 $w_4$ 均可以替换为 $w_2$.同理,$w_5$ 可以替换为 $w_3$,因此,折叠代价计算方法与内部点折叠操作相同.

### 2.3 计算惩罚量

根据以上分析,我们知道惩罚量$\alpha,\beta,\gamma$与 $Dist(v_s,v_t)$相比必须足够大,这样才能使可能产生外观畸变的半边折叠代价相对较大.下面我们将讨论这 3 个惩罚量的取值.

首先讨论$\alpha$的取值.假设某次半边折叠操作 $v_s \to v_t$,其中 $v_s$ 为接缝点,$v_t$ 为内部点,则由式(2)可知,其折叠代价为 $Cost(v_s,v_t)=Dist(v_s,v_t)+\alpha$,且必须大于其余不会产生畸变的折叠操作的代价,即

$$Cost(v_s,v_t) > \max\{Dist(v_i,v_j) \mid v_i \to v_j \in HEC(M^n)\} = \max\{(Q_i+Q_j)v_j \mid v_i \to v_j \in HEC(M^n)\} \tag{5}$$

其中,集合 $HEC(M^n)$表示简化模型 $M^n$ 中所有可折叠的边,$Dist(v_i,v_j)$表示 $v_j$ 到与 $v_i,v_j$ 相关的三角形的带权距离平方和,权值为三角形面积,即

$$Dist(v_i,v_j)=\sum_{t\in Tri(v_i)} Area(t)\cdot Dist(v_j,t)+\sum_{t\in Tri(v_j)} Area(t)\cdot Dist(v_j,t) \tag{6}$$

其中,$Area(t)$为三角形 $t$ 的面积,$Dist(v,v)$表示顶点 $v$ 到三角形 $t$ 的距离,$Tri(v)$表示与顶点 $v$ 相关的三角形集合.$HEC(M^n)$中的半边折叠 $v_i \to v_j$,其二次误差 $Dist(v_i,v_j)$的数量级与 $Dist(v_s,v_t)$相同.事实表明,绝大多数的 $Dist(v_i,v_j)$不会大于两倍的 $Dist(v_s,v_t)$(特殊情况将在下文讨论),因此,为了简化计算可放宽式(5)的约束,令 $Cost(v_s,v_t)\geq 2Dist(v_s,v_t)$且 $Cost(v_s,v_t)=Dist(v_s,v_t)+\alpha$,则$\alpha \geq Dist(v_s,v_t)$.

假设网格模型包围盒的长、宽、高分别为 $l,w,h$,则模型中任意两点距离的平方都不会大于 $l^2+w^2+h^2$.令 $\mathrm{Max}=\sqrt{l^2+w^2+h^2}$,则

$$\begin{aligned}Dist(v_s,v_t)&=\sum_{t\in Tri(v_s)} Area(t)\cdot Dist(v_t,t)+\sum_{t\in Tri(v_t)} Area(t)\cdot Dist(v_t,t)\leq \sum_{t\in Tri(v_s)} Area(t)\cdot \mathrm{Max}Dist+\sum_{t\in Tri(v_t)} Area(t)\cdot \mathrm{Max}Dist\\ &=\mathrm{Max}Dist\cdot\left(\sum_{t\in Tri(v_s)} Area(t)+\sum_{t\in Tri(v_t)} Area(t)\right)=\mathrm{Max}Dist\cdot\left(TriArea(v_s)+TriArea(v_t)\right)\end{aligned} \tag{7}$$

其中,$TriArea=\sum_{t\in Tri(v_s)} Area(t)$,是与顶点$v$相关的三角形的面积之和.我们令$\alpha$为 $\mathrm{Max}Dist\cdot(TriArea(v_s)+TriArea(v_t))$且$\alpha \geq Dist(v_s,v_t)$,从而有

$$Cost(v_s,v_t)\geq 2Dist(v_s,v_t)\geq Dist(v_i,v_j),$$

$(v_i,v_j)$代表绝大部分顶点对.少数不会产生外观畸变但 $Dist(v_i,v_j)$大于 $2\cdot Dist(v_s,v_t)$的半边折叠操作,其几何误差已经足够大,会严重影响视觉效果.因此,其折叠代价大于 $Cost(v_s,v_t)$是合理的,实验结果也验证了我们的想法.

$\beta$是从接缝点折叠到接缝点时,我们增加的惩罚量.这种情况与接缝点折叠到内部点所产生的畸变类似,所以$\beta=\alpha$.通常,交叉点是模型表面特征比较明显的顶点,因此,$\gamma$作为交叉点折叠到接缝点的惩罚量,应具有更大的值,令其为 $2\alpha$.最终 $v_s \to v_t$ 的折叠代价为

$$\begin{aligned}Cost(v_s,v_t)=&Dist(v_s,v_t)+\alpha\cdot IsSeam(v_s)\cdot(1-IsSeam(v_t))+\alpha\cdot IsSeam(v_s)\cdot IsSeam(v_t)\cdot(1-IsSeam(v_s,v_t))+\\ &2\alpha\cdot IsCross(v_s)\cdot IsSeam(v_t)\end{aligned} \tag{8}$$

其中,$\alpha=\mathrm{Max}Dist\cdot(TriArea(v_s)+TriArea(v_t))$.$\mathrm{Max}Dist$ 由网格模型的包围盒决定,在模型读入时计算.在原始模型中,$TriArea(v)$等于与顶点相邻的三角形面积之和.在执行 $v_s \to v_t$ 后,与顶点 $v_s$ 相关的三角形集并入到与 $v_t$ 相关的三角形集中,因此,与 $v_t$ 相关的三角形面积之和为$(TriArea(v_s)+TriArea(v_t))$.

## 3 算法描述

### 3.1 数据结构

算法的输入为顶点序列 VertexData 和索引序列 IndexData,前者代表网格模型中每个顶点的属性集合,其中

每个单元是一个 wedge,包含了坐标及其他外观属性,如纹理坐标 *texCoord*、法向量 *normal*、颜色 *color*;后者代表网格模型中顶点的连接关系,其中每个单元是一个 VertexData 序列的索引,用来检索 VertexData 中的 wedge.IndexData 中相邻的 3 个单元为一组,每组代表一个三角形,其中每个单元代表三角形的一个角.以图 6(a)所示的模型为例,其对应的 VertexData 和 IndexData 可以表示为图 6(b)中的结构.

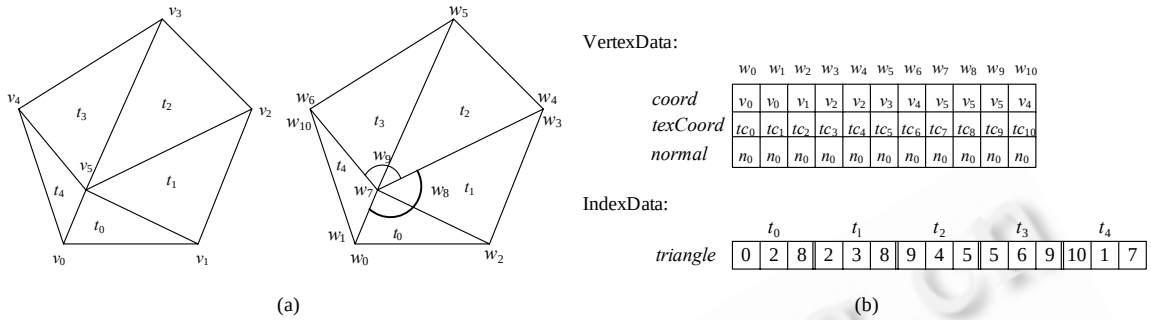



Fig.6 VertexData and IndexData of a mesh model

图 6 网格模型对应的 VertexData 和 IndexData

VertexData 和 IndexData 结构不仅表示网格模型的顶点属性和顶点之间的连接关系,同时也隐式地包含了网格模型中顶点之间以及顶点与三角形之间的邻接关系.为了描述和实现网格简化算法,我们引入了 3 个中间数据结构 PMVertex,PMWedge 和 PMTriangle 来显式地表示这些几何关系.

PMVertex 表示网格模型的一个顶点,是一个五元组(*geo*,*quadric*,*area*,*cost*,*to*)(如图 7 所示),其中 *cost* 和 *to* 分别表示最优折叠代价和最佳折叠终点,通过这两个值就能进行半边折叠操作.PMWedge 对应 VertexData 中的一个 wedge 单元,是一个二元组 (*realIndex*,*vertex*).每个假设 *w* 是一个 PMWedge,则 *w.realIndex* 是 *w* 在 VertexData 中的索引;*w.vertex* 是一个 PMVertex,为 *w* 对应的顶点.PMTriangle 表示网格模型上的一个三角形,是一个三元组 ($w_0$,$w_1$,$w_2$),3 个单元均为 PMWedge,以逆时针顺序分别对应三角形上的 3 个 wedge.

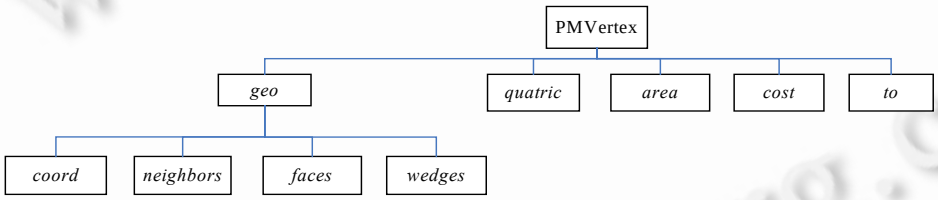



Fig.7 Structure of PMVertex

图 7 PMVertex 结构

在执行折叠操作之前,我们将 VertexData 和 IndexData 转化成这 3 个数据结构的表示形式:PMVertex 序列 *vertices*,PMWedge 序列 *wedges* 以及 PMTriangle 序列 *triangles*.

### 3.2 算法简介

在计算半边折叠 $u\rightarrow v$ 的折叠代价时,我们首先根据折叠边两个端点的 quadric 计算折叠操作的二次距离,然后通过判断两个端点与接缝的关系增加惩罚量.在每次迭代过程中,我们从 *vertices* 集合中未删除的 PMVertex 里找出折叠代价最小的顶点 *u*,执行 $u\rightarrow u.to$,直到达到某个期望的简化率为止.对于具有 *n* 个顶点的网格模型,如果通过顺序搜索 *vertices* 序列中的元素来获得 *cost* 最小的顶点,那么每次迭代过程的时间复杂度为 $O(n)$,这对折叠过程来说是非常可观的.为了降低搜索的复杂度,我们利用最小堆来获得 *cost* 最小的顶点[16],将每次搜索的时间复杂度降低到 $O(\lg n)$.

**算法 1**. ComputeCollpaseCost(*u*,*v*)伪代码.

输入:折叠边的起始点 $u$ 和终点 $v$;

输出:对应边的折叠代价.

1. $Q := u.quadric + v.quadric$
2. $dist := \boldsymbol{Q}(v)$
3. If IsSeam($u$) then
4. $\alpha := (u.area + v.area) * \text{Max}Dist$
5. If IsCross($u$) then
6. $cost := dist + 2\alpha$
7. Else if !IsSeam($v$) then
8. $cost := dist + \alpha$
9. Else if IsSeam($v$) and !IsSeam($u$, $v$) then
10. $cost := dist + \alpha$
11. End if
12. End if
13. Return *cost*

一次半边折叠过程 $u \rightarrow v$ 分为 8 个步骤:

1. 保存折叠起点 $u$ 的邻接顶点,因为这些顶点的最优折叠代价会在折叠后改变,需要重新计算;
2. 将 $u$ 的邻接三角形分成两类:要被删除的 *removal* 集合;其上的 wedge 要被替换的 *replacement* 集合;
3. 为要被删除的 wedge 寻找合适的替代者,即替换后不会引起视觉畸变;
4. 将 *removal* 包含的 PMTriangle 删除,并通知相关顶点,将其从邻接三角形集合中删除;
5. 为 *replacement* 替换相应的 PMWedge,如果没有合适的,则从 *v.geo.wedges* 中随机选一个;
6. 将被替换的 PMWedge 从 *wedges* 中删除;将 *replacement* 中的 PMTriangle 添加到顶点 $v$ 的邻接三角形中,并更新所有受影响的顶点的连接关系;
7. 将顶点 $u$ 的 quadric 及相关三角形面积累加到 $v$ 上;
8. 重新为受影响的顶点计算最优折叠代价和最佳折叠终点,并更新它们在最小堆中的位置.

## 4 实验结果和效率

利用改进网格模型简化算法 M-QEM,我们对几个用三维扫描仪取得的虚拟博物馆中的藏品模型进行简化,并与 PM 和 QEM 算法进行比较.另外,M-QEM 算法适用于任意规模的网格模型,因此我们从 Stanford 大学的网络资源中下载了更为复杂的网格模型[17]执行简化操作.实验结果是在一台普通消费级 PC 机上获得,系统配置为 Intel Pentium IV 2.94 GHz 的 CPU 和 1GB 内存.

表 1 比较了 M-QEM 与 PM,QEM 算法在网格模型构建上所花费的时间,显然,M-QEM 在速度上远胜于 PM 算法.由于我们对 QEM 算法进行了扩展,因此无法达到 QEM 的速度.但由表 1 可得,M-QEM 的速度与 QEM 属于同一数量级,算法依旧保持了很高的简化效率.

我们利用 Metro[18]工具对简化后的模型进行几何相似度分析.图 8 和图 9 比较了 M-QEM 和 PM 算法所生成的一系列不同分辨率的绿柱石模型的几何相似度.结果显示,M-QEM 具有非常理想的效果,其平均误差在绝大多数情况下都小于 PM 算法所得到的结果,但在某些分辨率下的最大误差比 PM 的要大.这是因为我们对模型的纹理接缝进行了特殊处理,导致相关顶点的偏移量较大;PM 算法则没有对这些顶点进行处理.

在实验中我们发现,只有当三角形数在 2 000~500 之间并从特定角度进行观察时,M-QEM 所得的纹理误差才会比 PM 的大,如图 10 所示.这同样是因为我们对纹理接缝进行了特殊处理,因此,从接缝可见的角度观察会得到更大的纹理误差.除此之外,M-QEM 都得到了更为理想的结果.当三角形数在 500~12 范围内时,PM 生成的模型已产生很大的误差和外观畸变,而 M-QEM 算法生成的模型依旧保持了正确的外观属性.

**Table 1** Multi-Resolution construction time

**表 1** 网格模型构建时间比较

| Models | Number of vertices | Number of triangles | Running time (millisecond) | | |
|---|---|---|---|---|---|
| | | | M-QEM algorithm | PM algorithm | QEM algorithm |
| Emerald | 2 502 | 5 000 | 66 | 754 | 31 |
| Agate | 25 002 | 50 000 | 793 | 7 484 | 496 |
| Cyrtospirifer rudkiensis | 30 861 | 61 718 | 952 | 9 210 | 629 |
| Bunny | 362 272 | 725 000 | 8 138 | 99 599 | 5 422 |
| Armadillo | 3 390 515 | 7 500 000 | 98 192 | 1 108 884 | 76 109 |
| Happy Buddha | 4 586 124 | 9 200 000 | 141 848 | 1 372 290 | 103 721 |

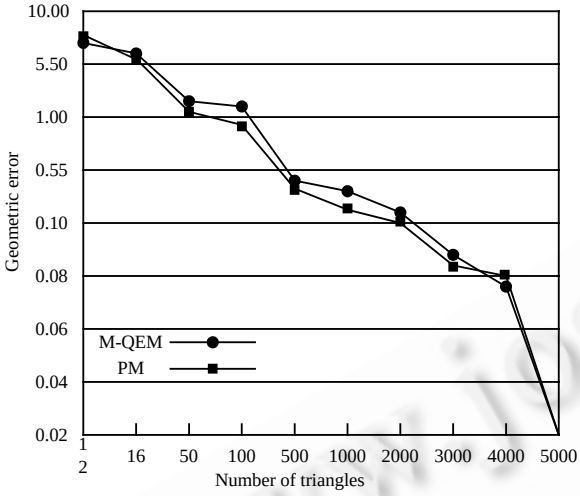



Fig.8 Max geometric errors

图 8 最大几何误差

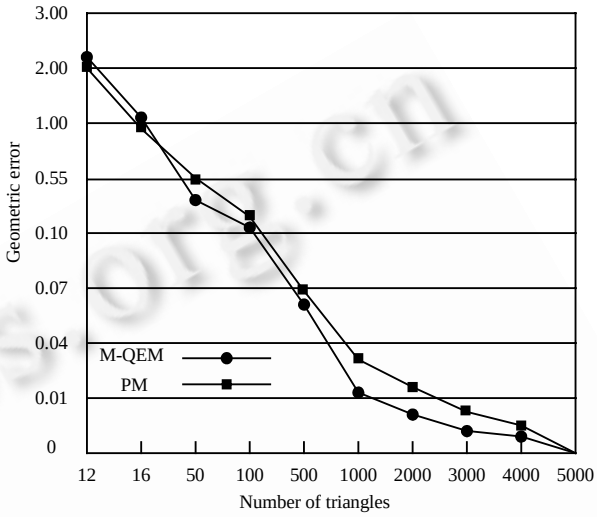



Fig.9 Mean geometric errors

图 9 平均几何误差

图 11 展示了 M-QEM 算法简化得到的模型,最左侧一列是绿柱石、玛瑙、鳍德克弓石燕的原始模型,分别具有 5 000,50 000 和 61 718 个三角形,后两列则是三角形数为 1 000 和 150 的简化模型.由图可知,简化模型的几何形状保持了很高的相似度,特别是在模型的细节特征部分,如棱边、沟壑等都完整地保留了下来.对于纹理的连续外观属性,简化模型与原始模型几乎没有差别,这就证实之前我们没有使用扩展 QEM 是正确的.如果使用扩展的 QEM,效果应该会更理想,但是会花费更多的储存空间和计算时间.

我们对绿柱石模型进行进一步简化,如图 12 所示,从左至右,三角形数依次递减,分别为 100,30,12,10.当三角形数为 100 时,模型的几何形状和外观都具有良好的保真度,而且纹理接缝也没有发生畸变.当模型仅有 30 个三角形时,其几何形状与原始模型有了可觉察的差别,从图中即可看出,绿柱石模型顶端突起的部分变平,整个模型呈六棱柱状,与原始模型的整体轮廓一致,模型纹理也没有出现畸变.当网格模型的三角形数简化到 12 时,模型的几何形状类似于长方体,与原始模型相比有了明显差别.尽管如此,模型的纹理依然没有发生畸变,这说明半边折叠操作的执行次序是完全正确的.根据 M-QEM 定义的折叠代价计算方法,以交叉点为起点的半边折叠具有最高的折叠代价,所以具有 12 个三角形的简化模型完整地保留了 8 个交叉点.当模型进一步简化时,由于剩下的顶点都是交叉点,我们只能以交叉点为起点进行半边折叠.因此,如图 12 中最右侧的简化模型所示,其三角形数减少到了 10,纹理便发生了畸变.

在半边折叠代价计算公式中,我们并没有局限于某种特定的纹理坐标或其他属性值,而是将属性抽象成 wedge,所以顶点的各种外观属性都可以应用 M-QEM 算法.

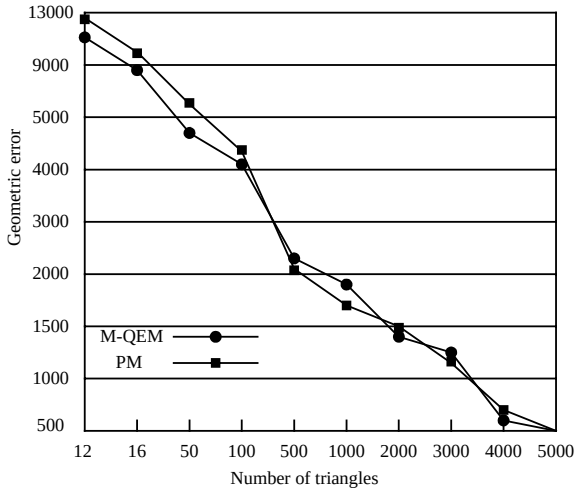



Fig.10 Textural errors

图 10 纹理误差

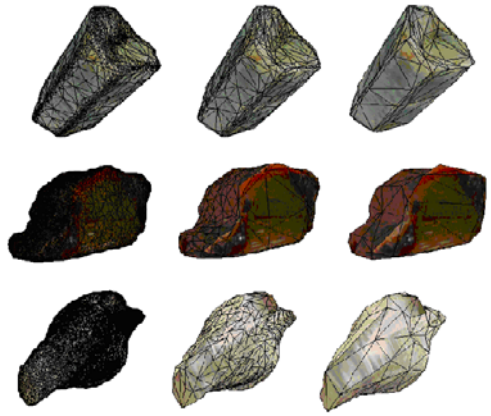


Fig.11 Sequence of approximations of the emerald, agate and Cyrtospirifer rudkiensis

图 11 绿柱石、玛瑙和鳍德克弓石燕的简化模型

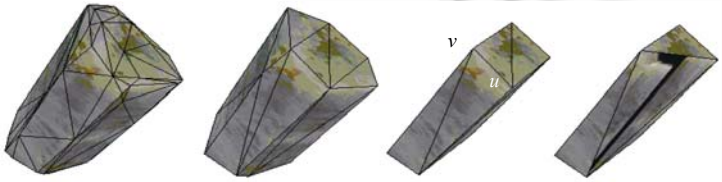



Fig.12 Further simplification of emerald model

图 12 对绿柱石模型的进一步简化

## 5 结论和进一步工作

本文针对具有非连续外观属性的模型在渲染效果和效率上的需求,以QEM算法为基础,实现了一个运行快速、保真度高的网格模型简化算法M-QEM.该算法具有如下特点:

第一,以 QEM 作为相似度评估方法指导折叠过程,具有很高的简化效率,并且在几何形状和外观特性上都保持了很高的相似度;

第二,针对涉及外观属性接缝的半边折叠,以 QEM 折叠代价为基础,合理地加上了与接缝有关的惩罚量,使得外观畸变在简化过程中尽可能地推迟;

第三,采用半边折叠操作作为简化方式,在模型简化过程中不引入新的顶点,而是重复利用已有的数据结构,减少了内存占用,并且有利于构建渲染系统可直接处理的数据结构,提高了渲染效率.

虽然我们使用 QEM 作为误差评估手段,保持了算法的高效性,但是由实验结果可知,简化一个具有 60 000 多个三角形面片的网格模型需要近 1s,无法满足实时应用.由于网格模型简化是一次性过程,我们将整个过程中的相关信息记录下来,以便在实时渲染时重建出所需的分辨率模型.我们需要一个适合于实时渲染的网格多分辨率模型表示方法,以记录网格模型的简化信息,并且能够在很短的时间内恢复出任意分辨率的模型.在下一步工作中,我们将利用全序网格结构[19]作为多分辨率模型的数据结构,着重研究如何根据该数据结构重建出网格模型.

**References**:

[1] Pan ZG, Pang MY. Survey for decimation of geometric meshes. Journal of Jiangsu University, 2005,26(1):67−71 (in Chinese with English abstract).

[2] Luebke D. A developer's survey of polygonal simplification algorithms. IEEE Computer Graphics and Applications, 2001,21(3):24−35.

[3] Hearn D, Baker MP. Computer Graphics. 2nd ed., Upper Saddle River: Prentice Hall/Pearson, 1996. 45−46.

[4] Dong WL, Li JK, Jay Kuo CC. Fast mesh simplification for progressive transmission. In: Akansu AN, ed. Proc. of the IEEE Int'l Conf. on Multimedia & Expo. New York: IEEE Computer Society Press, 2000. 1731−1734.

[5] Luebke D, Reddy M, Cohen J, Varshney A, Watson B, Huebner R. Level of Detail for 3D Graphics. San Francisco: Morgan Kaufmann Publishers, 2002. 431−467.

[6] Cignoni P, Montani C, Scopigno R. A comparison of mesh simplification algorithms. Computers and Graphics, 1998,22(1):37−54.

[7] Hoppe H. Progressive meshes. In: Rushmeier H, ed. Proc. of the SIGGRAPH. New Orleans: Addison-Wesley Professional, 1996. 99−108.

[8] Hoppe H, DeRose T, Tom Duchamp. Mesh optimization. In: Thomas JJ, ed. Proc. of the SIGGRAPH. Chicago: ACM Press, 1992. 19−26.

[9] Hoppe H. New quadric metric for simplifying meshes with appearance attributes. In: Ebert DS, Gross MH, Hamann B, eds. Proc. of the IEEE Visualization. Los Alamitos: IEEE Computer Society Press, 1999. 59−66.

[10] Kobbelt L, Campagna S, Seidel HP. A general framework for mesh decimation. In: Davis W, Booth K, Fournier A, eds. Proc. of the Graphics Interface. New York: ACM Press, 1998. 43−50.

[11] He HG, Tian J, Zhang XP, Zhao MC, Li GM. A survey of mesh simplification. Journal of Software, 2002,13(12):2215−2224 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/2215.htm

[12] Garland M, Heckbert P. Surface simplification using quadric error metric. In: Whitted T, ed. Proc. of the SIGGRAPH. Los Angeles: ACM Press, 1997. 209−216.

[13] Garland M, Heckbert PS. Simplifying surfaces with color and texture using quadric error metrics. In: Ebert DS, Rushmeier H, Hagen H, eds. Proc. of the IEEE Visualization. Washington: IEEE Computer Society Press, 1998. 263−270.

[14] Ronfard R, Rossignac J. Full-Range approximation of triangulated polyhedra. Computer Graphics Forum, 1996,15(3):67−76.

[15] Hoppe H. Effective implementation of progressive meshes. Computer & Graphics, 1998,22(1):27−36.

[16] Garland M. Quadric-Based polygonal surface simplification [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 1995.

[17] The Stanford 3D scanning repository. http://graphics.stanford.edu/data/3Dscanrep/

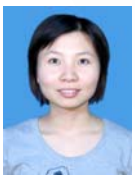
[18] Cignoni P, Rocchini C, Scopigno R. Metro: Measuring error on simplified surfaces. Computer Graphics Forum, 1998,17(2):167−174.

[19] Bouvier E, Gobbetti E. TOM: Totally ordered mesh-A multiresolution structure for time cirtical graphics applications. Int'l Journal of Image and Graphics, 2001,1(1):115−134.

**附中文参考文献**:

[1] 潘志庚,庞明勇.几何网格简化研究与进展.江苏大学学报,2005,26(1):67−71.

[11] 何晖光,田捷,张晓鹏,赵明昌,李光明.网格模型化简综述.软件学报,2002,13(12):2215−2224. http://www.jos.org.cn/1000-9825/13/2215.htm



**卢威**(1984－),女,江西九江人,博士生,主要研究领域为计算机图形学,虚拟现实技术.

**潘金贵**(1952－),男,教授,博士生导师,主要研究领域为多媒体信息处理技术,计算机图形学,远程网络教育.

**曾定浩**(1980－),男,硕士,主要研究领域为计算机图形学,分布式虚拟环境.