网格的渐进几何压缩*

秦绪佳[1], 刘新国[2], 鲍虎军[1], 彭群生[1]

[1](浙江大学 CAD&CG 国家重点实验室,浙江 杭州 310027);
[2](微软亚洲研究院,北京 100080)
E-mail: {qinxj,bao,peng}@cad.zju.edu.cn
http://www.zju.edu.cn

摘要: 提出一种渐进几何压缩算法.通过对简化算法的改进,网格模型由基网格及多组顶点分裂操作序列表达.当从一层网格向下一层精网格细化时,该组顶点分裂操作序列中的分裂操作顺序是任意的.因此,改进的渐进网格表示可改变每组顶点分裂操作的排序,实现高效率编码.设计了 Laplacian 几何预测器,通过相邻顶点来预测新增顶点位置,并对位置校正值进行量化及 Huffman 编码.实验结果表明,该算法可获得高压缩比,适合几何模型的网络渐进传输.

关 键 词: 三角网格;几何压缩;流形曲面;渐进网格
中图法分类号: TP391 文献标识码: A

尽管自由曲面普遍用于 CAD 和计算机动画系统中,但多边形网格,尤其是三角形网格在原型制造、虚拟现实和可视化中得到了广泛应用.与自由曲面相比,网格模型需要大量的信息来记录顶点间的连接关系及顶点几何坐标,对于复杂模型更需要大量的存储空间,且传输较慢.复杂网格模型在网络上传输速度慢的问题已越来越突出.

为了减少存储空间、加快传输速度,网格必须像视频、音频等媒体文件那样得到有效的压缩.近年来,一些学者在这方面做了大量研究[1~11],提出了一些普遍框架.比如,将模型分割成简单的三角网格片,并对其拓扑信息进行压缩编码.采用预测方法对新增顶点进行位置的估算实现位置压缩.其他属性数据,如法向、颜色等均可用同样的方法进行压缩编码.Taubin[3],Touma[4]和 Bajaj[10]分别给出了对单分辨率网格压缩的高效编码策略.

由于互联网的发展,人们通过网络对共享信息的需求变得更为迫切,因此,通过有限带宽网络实现大规模网格的传输变得越来越重要.单分辨率的几何压缩方式不能满足网格传输的要求,因为用户在网格数据传送完毕之前,不能看到网格的任何信息.理想的方法是,采用由基网格渐进地更新恢复出原始网格的传送方式,即先将基网格传送到远程客户端,然后不断传送细化信息,使基网格不断得到精化,逼近原始网格.即使网络中断,客户端也能获得原始网格的一个简化版本.因此,首选的压缩和传输网格模型是采用多分辨率来表示模式.

有两种多分辨率表示模式.一种模式是基于细分连续性的[12],其多分辨率网格通过基网格不断细分得到.在该模式中,顶点连接性及位置的编码直观、简单、有效.另一种模式是基于连续简化,如 Hoppe 的渐进网格(progressive mesh,简称 PM)[13].PM 表达适合任意网格的渐进传输,其网格表达为基网格与一系列的顶点分裂操作.PM 模式可以达到平均一个顶点 $(5.3+\log_2 n)$ 比特的紧凑表示.Taubin 的渐进森林分裂(progressive forest split,简称 PFS)模式[14]给出了一种更为复杂的分裂操作,称为森林分裂,以构造网格的多分辨率表示.森林分裂

* 收稿日期: 2002-03-01; 修改日期: 2002-06-13
基金项目: 国家自然科学基金资助项目(60133020;60021201;69925204)
作者简介: 秦绪佳(1968－),男,广西桂林人,博士,讲师,主要研究领域为计算机图形学,几何造型;刘新国(1972－),男,江西九江人,博士,副研究员,主要研究领域为计算机图形学,虚拟现实;鲍虎军(1966－),男,浙江乐清人,博士,研究员,博士生导师,主要研究领域为计算机图形学,虚拟现实,几何造型;彭群生(1947－),男,湖南新化人,博士,教授,博士生导师,主要研究领域为计算机图形学,虚拟现实,红外仿真成像技术.

操作是分裂网格的森林边(forest edge),可视为推广的 PM 边分裂操作.森林分裂会产生许多多边形洞,将它们三角化并进行编码.PFS 方法可以得到平均每个网格顶点占 4.0~5.0 比特的压缩率.Bajaj[15]提出另一种渐进压缩模式,他将原始网格分解为一系列的层边界轮廓,多分辨率表达为建立简化和重构轮廓的编码.尽管这种模式能够得到与 PSF 模式相似的压缩结果,但基网格的质量往往较差,并且中间层次的网格也不能较好地逼近初始网格.

本文提出一种新的基于 PM 的渐进几何压缩模式.首先对 PM 作简要介绍,然后论述生成 PM 表达的高效渐进压缩的简化算法,并给出编码模式,最后给出实验结果并对全文进行总结.

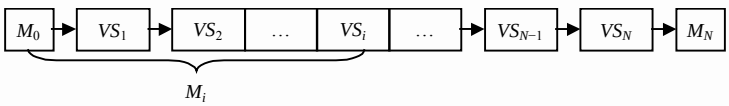
1 渐进网格(PM)与顶点分裂

文献[13]提出的渐进网格是任意网格的连续多分辨率表示.它通过递归地对网格施以边折叠操作实现,如图 1 所示.

EC: edge collapse①. M_N is the original model, M_i is the simplified model when EC_i is performed, and M_0 is the coarsed base mesh②.
①*EC*:边折叠,②M_N 是初始网格,M_i 是经 EC_i 操作后的简化网格,M_0 是基网格.

Fig.1 Progressive mesh construction
图 1 渐进网格的构造

边折叠是可逆的,其逆操作是顶点分裂操作,原始网格可由简单的基网格 M_0 经过一系列的顶点分裂操作恢复,如图 2 所示.这是网格简化的逆过程.因此,原始网格可以表示为基网格与一系列的顶点分裂操作,可直接用于渐进传输.如图 3 所示,3 个参数(u,*first*,*last*)可惟一地定义一个顶点分裂操作,u 为待分裂顶点,*first* 和 *last* 是共享由 u 分裂出的边$\langle u,v\rangle$的两个三角形的顶点.根据这 3 个参数,新顶点 v 可以严格地恢复出来.



VS: vertex split①. M_N is the original model; M_i is the refined model when VS_i is performed; M_0 is the coarsed base mesh②.
①*VS*:顶点分裂,②M_N 是初始网格;M_i 是经 VS_i 操作后的细化网格;M_0 是基网格.

Fig.2 Progressive mesh refinement
图 2 渐进网格恢复

在 PM 表达中,基网格往往较为简单,主要存储消耗为记录分裂顶点的索引及位置,因此渐进压缩的关键在于如何对顶点分裂进行有效的编码.我们的策略基于下面的事实:如果将分裂顶点序列按索引增序排列,可大大减少分裂顶点序列所占的比特数.然而,对于 PM 表达,顶点分裂操作序列是网格简化操作的严格逆序.因此,必须设计一种新的简化算法来构造渐进网格表达.此算法记录特殊的顶点分裂顺序,并具有高压缩比.为了满足这一要求,简化算法应具有以下优点:

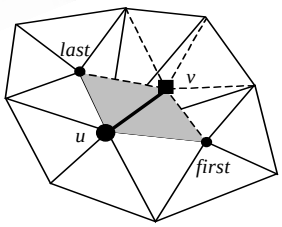
(1) 能快速实现网格简化;

(2) 简化网格能较好地逼近原始网格;

(3) 所得到的多分辨表达能进行高效的编码;

(4) 对每一个分裂顶点,分裂得到的新顶点的位置能尽量接近初始网格的对应顶点.

下一节将讨论具有上述特性的新简化算法.



v is the decimated vertex, u is the split vertex, and $\langle u,v\rangle$ is the collapsed edge①.
①v 为删除顶点,u 为分裂顶点,$\langle u,v\rangle$为折叠边.

Fig.3 Edge collapse and vertex split
图 3 边折叠和顶点分裂

2 渐进压缩的简化模式

我们知道,最优逼近原始网格的方法是将边按误差递增排序得到折叠边优先队列[13].由于误差估计是很费时的,难以满足我们的需要.在本文的改进算法中,采用两步策略来选择折叠边.

第 1 步,采用顶点排序的边长准则.删除顶点的优先级按下式权值 $priority(v)$ 增序排序.

$$priority(v)=\alpha\cdot\min\_edge(v)+\beta\cdot\max\_edge(v),\tag{1}$$

其中, $\min\_edge(x)$ 和 $\max\_edge(x)$ 分别是顶点 v 的邻边之中最短、最长的边长.

这一准则保证了简化网格的三角形的大小、形态均匀,有利于后续的几何预测.

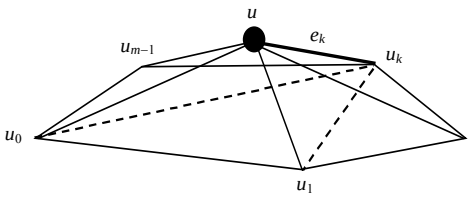


Fig.4 Volume-Preserving decimation cost

图 4 体积保持的顶点删除代价

第 2 步,在如何选择与删除顶点连接的边作为折叠边方面,传统的方法是以共享这条边的两三角形的夹角作为评判标准.这里,我们提出用局部体积准则来评价边折叠代价.

设 u 为待删除顶点, $u_k(k=0,1,...,m-1)$ (m 为其 1-环邻点数)为 u 的 1-环邻点,我们采用 u 及 u_k 形成的锥形体积来度量边折叠的局部体积变化.当共享边 $e_k=\langle u,u_k\rangle$ 被折叠时,如图 4 所示,锥形的侧面由三角形 $\Delta uu_ju_{j+1}(j=0,1,...,m-1)$ 围成,底面由三角形 $\Delta u_ju_{j+1}u_k(j=0,1,...,m-1)$ 构成,局部体积变化可用锥形体积来度量:

$$collapsed\_volume(e_k)=\sum_j((u_j-u_k)\otimes(u_{j+1}-u_k))\cdot(u-u_k).\tag{2}$$

显然,当 u 及其 1-环邻点共面时,锥形体积等于 0.我们发现,锥形体积大小直接反映了简化网格对原始网格的逼近程度,锥形体积越小,逼近程度越高.在我们的算法中,边折叠的代价可表示为

$$cost(e_k)=\|collapsed\_volume(e_k)\|.\tag{3}$$

相应地,具有最小代价的边 $e_{\text{collapsed}}$ 选作折叠边:

$$cost(e_{\text{collapsed}})=\min_{0\le k<m}cost(e_k).\tag{4}$$

最后,为控制全局逼近误差,还要进一步检测所选取的折叠边 $e_{\text{collapsed}}$ 在进行折叠操作后其简化误差是否小于事先给定的阈值 E,如果是,则可沿该边折叠.在具体实现中,采用原模型的顶点到简化模型的距离来估计简化误差.实验结果(如图 5 所示)表明,局部体积误差度量准则可使简化模型保持原始模型的特征.

(a) Level 5 (a) 层次 5 | (b) Level 7 (b) 层次 7 | (c) Level 10 (c) 层次 10 | (d) Level 14 (d) 层次 14 | (e) Level 18 (e) 层次 18 | (f) Level 23 (the original) (f) 层次 23(原始模型)

Fig.5 Six levels of the multiresolution representation of the bunny model with 24 levels (α=1.0,β=0.01)

图 5 Bunny 模型多分辨率表示的 24 个层次简化模型中的 6 个(α=1.0,β=0.01)

上述的基本简化算法可获得高质量的简化网格,但还不能达到表达渐进压缩的目的.如前所述,我们引入“独立点集”对折叠边序列加以约束.在每一简化层次上,当一个顶点被删除时,暂时锁定该顶点的 1-环邻点为不可删除顶点,直至该层简化结束.所有被删除的顶点均取自独立点集.显然,顶点在独立点集中的顺序是任意的,即简化过程的边折叠顺序是相互独立的.这样,我们可以按任意顺序对删除点进行传送,从而可对顶点分裂进行高效压缩.同时,一个顶点分裂后,新顶点的位置可通过独立点集来预测,因为它的所有 1-环邻点都可在独立点集中方便地检索到.实际上,每个独立点集记录每一层次分裂顶点序列,所有独立点集构成如图 6 所示的层次结构.

在编码模式上,属于同一独立点集的顶点分裂操作分在一组,并一起进行编码.因此,在简化过程中独立点集可能会很大,一般地,对于一个顶点规模为 n 的网格,独立点集可能达到 $n/4$ 个顶点.为满足这一要求,本文采用下面的策略来构造 PM 表达.

对事先给定的全局简化误差 E (可根据网格的包围盒自动设定),设 R 为每一简化层次中删除顶点数目的最小百分率,通常 R 在区间 $\left[\frac{1}{5},\frac{1}{4}\right]$ 上选取.由于误差是随简化过程积累的,为控制误差,我们将全局误差 E 分割

成 S 步误差的累加,即 $\left\{E_k=\frac{k}{S}E \mid k=1,2,...,S\right\}$,以自适应地控制生成每一层次简化网格的渐进简化过程.考察由网格 M_{i+1} 简化至网格 M_i,设 E_k 为当前的简化误差阈值,在独立点集及阈值 E_k 的控制下对网格 M_{i+1} 进行边折叠操作,直至没有可折叠的边.如果独立顶点集小于 $R\cdot\|M_{i+1}\|$ ($\|M_{i+1}\|$ 是 M_{i+1} 的顶点数),则锁定点集中的所有顶点并进入下一层次(M_i)的简化;否则,用 E_{k+1} 替换当前误差阈值 E_k,继续边折叠操作.当最大累积误差达到 E 时,整个简化过程结束,所得到的简化网格序列 $\{M_i, i=N,N-1,...,0\}$ 满足:

$$D(M_{i+1},M_N)\le D(M_i,M_N)\le E,\ i=0,1,...,N-1,$$
$$\frac{\|M_i\|}{\|M_{i+1}\|}<1-R, \tag{5}$$

其中 $D(\cdot,\cdot)$ 为两网格的距离函数.

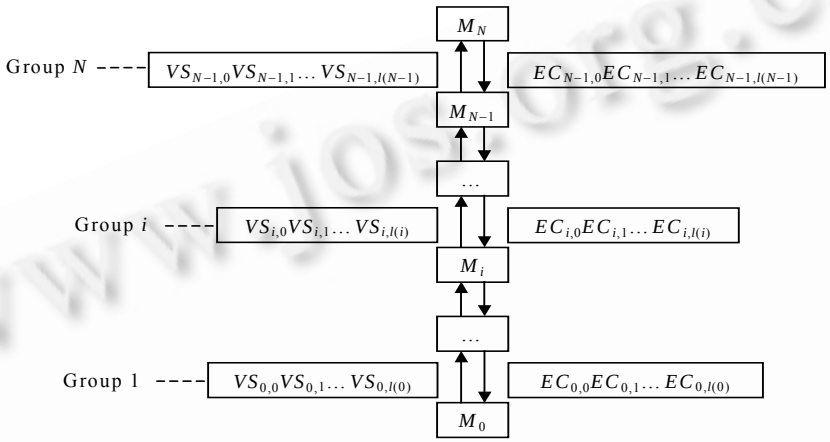


Fig.6 Hierarchical PM representation. M_i is the simplified model at level i

图 6 层次 PM 表达.M_i 是第 i 层次的简化模型

参数 S 直接影响网格的简化质量及速度,一般地,S 越小,简化网格的质量越高,但简化速度越慢.由于顶点为 n 的网格其最大独立点集的顶点个数可能达到 $n/4$,因此可用下式来估算 S:

$$S=\frac{\log\frac{\|M_N\|}{\|M_0\|}}{\log\frac{3}{4}}, \tag{6}$$

其中 $\|M_N\|$ 和 $\|M_0\|$ 分别为原始网格和基网格的顶点数.

兔子模型的多分辨率表示在图 5 中已经给出.图 7 为 24 层简化模型中的 3 层,整个简化算法过程在 Pentium III 500 上的运行时间为 4.84 秒,简化网格能较好地保持原模型的特征.图 7 中深颜色的三角形为边折叠中删除的三角形.由此可见,改进算法在每一层次上的折叠边/分裂顶点分布均匀,这为后面的顶点位置压缩提供了良好的条件.

3 渐进几何压缩

由图 6 可以看出,渐进简化算法从原始网格 M_N 开始,网格 M_{i+1} 经过一组边序列折叠操作得到简化网格 $M_i(i=N-1,...,1,0)$,重复简化过程直至基网格 M_0.我们的渐进压缩模式是上述简化过程的逆过程,从基网格 M_0 开始,网格 M_i 经过一组顶点序列的分裂操作得到网格 $M_{i+1}(i=0,1,...,N-1)$,设 $P_i=\{(u_j^i,L_j^i,F_j^i)\,|\,j=0,...,l(i)\}$ 为由网格 M_i 到 M_{i+1} 的第 i 组分裂顶点序列,由于所有由第 i 组分裂顶点序列(独立顶点集)中的顶点分裂得到的新顶点均在网格 M_{i+1} 上,所以由 M_i 恢复 M_{i+1} 时可在 P_i 中采用任意的操作顺序.不失一般性,可假设 $u_j^i\le u_{j+1}^i$.在这一节中,我们讨论如何对顶点 P_i 进行连接性编码和位置编码.

(a) Level 16 (b) Level 11 (c) Level 8
(a) 层次 16 (b) 层次 11 (c) 层次 8

Fig.7 The distribution of the collapsed triangles
图 7 折叠三角形的分布

3.1 连接性(拓扑)编码

如前所述,每个顶点分裂操作由 3 个参数表征.第 1 个参数为分裂顶点,新顶点由该顶点分裂得到,同一组顶点分裂操作按顶点 u_j^i 的单调非减序列排序:

$$u_1^i \le u_2^i \le \ldots \le u_{l(i)}^i .$$

这一特性可保证分裂顶点实现高效编码.这里不直接采用 u_j^i 序列 $\{u_j^i \mid j=0,1,2,\ldots,l(i)\}$ 编码成比特流,而是对 $u_j^i - u_{j-1}^i$ 序列 $\{u_j^i - u_{j-1}^i \mid j=1,2,\ldots,l(i), u_0^i = 0\}$ 进行压缩编码.由于 $\{u_j^i\}$ 是单调非减序列,显然, $u_j^i - u_{j-1}^i \ge 0$.根据简化约束,有

$$l(i) > R \cdot \|M_{i+1}\| = R \cdot (l(i) + \|M_i\|), \tag{7}$$

$$l(i) > \frac{R}{1-R}\|M_i\|. \tag{8}$$

因为

$$\sum (u_j^i - u_{j-1}^i) = u_{l(i)}^i - u_0^i = u_{l(i)}^i < \|M_i\|, \tag{9}$$

不同序列的平均值满足

$$\frac{\sum (u_j^i - u_{j-1}^i)}{l(i)} < \frac{1-R}{R} = \frac{1}{R} - 1, \tag{10}$$

因为 $1/5 \le R \le 1/4$,所以

$$\frac{\sum (u_j^i - u_{j-1}^i)}{l(i)} < 4 . \tag{11}$$

这表明采用 Huffman 编码,平均每个单元的编码只占 2 比特.

顶点分裂操作的第二参数与分裂顶点 u_j^i 的入度有关.设 v_j^i 为顶点分裂操作 (u_j^i, L_j^i, F_j^i) 产生的新顶点,顶点分裂操作的第三参数与新顶点 v_j^i 的入度有关.因此有

$$d_j^i = \begin{cases} L_j^i - F_j^i + 2, & \text{当} u_j^i \text{为内部顶点时} \\ L_j^i - F_j^i + 1, & \text{当} u_j^i \text{为边界顶点时} \end{cases}. \tag{12}$$

同样地,可采用对 d_j^i $\{d_j^i = L_j^i - F_j^i \mid j=1,2,\ldots,l(i)\}$ 的编码来替代对 L_j^i 和 F_j^i 的编码.Huffman 编码同样可用于对 d_j^i 的编码,即对 P_i 的后两部分的编码.由于网格的平均入度为 6,因此对 P_i 后两部分的编码 5 比特就足够了.所以,采用我们的编码模式,每个顶点的连接性编码只需 7 比特.

3.2 顶点位置压缩

相对于顶点连接信息,顶点的位置信息是占用三角网格主要的存储空间.渐进压缩模式的关键是对顶点的几何坐标进行有效压缩.我们采用图像压缩中的预测-校正方法和 Huffman 编码对顶点位置进行压缩.

注意到预测-校正效果主要取决于预测结果,因此必须寻找有效的预测器用于我们的算法.如前所述,本文的渐进表达是由基网格经过一系列的顶点分裂操作实现的,分裂顶点及其 1-环邻点均是已知的,新顶点位置可以由分裂顶点及其 1-环邻点预测.考察分裂顶点 u,设 v 为 u 分裂生成的新顶点,我们采用 Laplacian 算子组合来

构造预测器[12,16~18]:

$$w=\frac{1}{m}\sum_{j=1}^{m}\left(u_j+L(u_j)\right), \tag{13}$$

其中 m 为顶点 v 的入度,u_j 为 u 的 1-环邻点,$L(u_j)$是顶点 u_j 的 Laplacian 算子,$L(u_j)$定义为

$$L(u_j)=\frac{1}{m_j}\sum_{k=1}^{m_j}(u_{j,k}-u_j)\,. \tag{14}$$

这里,$u_{j,k}$表示 m_j 个 u_j 的 1-环邻点.显然,只需将校正后的矢量 ***v*–*w*** 写入到比特流中.为了提高压缩效率,校正矢量根据用户指定精度进行量化及 Huffman 编码.显然,Huffman 码表是进行编码的关键,图 8 给出了标准兔子模型顶点预测值的校正量幅值和量化后的分布直方图.

(a) Distribution of the quantized correction coordinates (b) Histogram of the corrections in (a)
(a) 量化校正值的分布 (b) 对(a)校正后的直方图

Fig.8 Quantized vertex position correction of the bunny model (10 bits quantization precision)
图 8 兔子模型顶点位置校正值量化(10 比特量化精度)

有两种以传统压缩模式处理 Huffman 表的方法.一种方法是由用户通过对很多网格的分析统计来定义 Huffman 表,Huffman 表不写入压缩比特流中,编码效率不是最优,特别是在待编码数据量很大时.此方法只适合小规模数据.另一种方法非常复杂,首先频率函数源于待编码数据,Huffman 表由频率函数构造.在本文的压缩模式中,采用第 2 种方法来优化编码效率.然而,将 Huffman 表写入压缩比特流要耗费存储空间,为此,我们基于以下几点设计紧凑表达的 Huffman 表:

(1) 如图 8 所示,顶点位置校正直方图是一个粗糙的单调递减函数;

(2) 对 Huffman 表 H_0,可构造惟一码长的表 H_1,且与 H_0 有相同的编码效率;

(3) 如前所述,对单调递增整数序列,采用对其差分序列编码效率更高.

第(1)点说明,分裂操作顶点的位置校正在 Huffman 表中的码长是大致单调上升的.第(2)点说明,只要存储所有位置校正的 Huffman 码长.鉴于第(3)点所具有的优点,可以大大减少顶点位置校正在 Huffman 表中的存储量.设 d 为当前顶点分裂操作与前一操作的码长差,我们将差值 d 按下面方式写入比特流中:

$$\begin{cases} 0 & \text{if } d=0;\\ 1\,00\ldots0\,1 & \text{if } d>0, \text{ the number of bit 0 is } d;\\ 1\,11\ldots1\,0 & \text{if } d<0, \text{ the number of 1 is } d \text{ except the first bit 1.}\end{cases}$$

根据此编码策略,Huffman 表可以高效地编码.在解压缩时,恢复的码长序列可以方便地转换成 Huffman 表.

4 码本结构与解压缩

本文用 PM 表达的码本结构组织如图 9 所示.在实现中,基网格直接写入比特流中,因为基网格一般较简单,规模较小.必要时也可对基网格按单分辨率的几何压缩模式进行压缩编码.

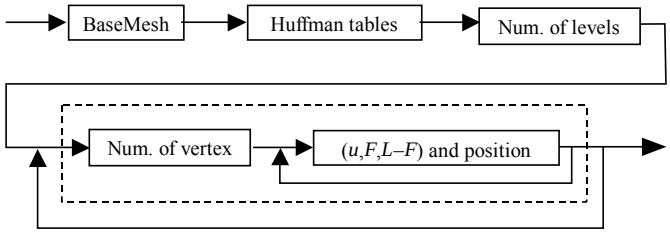


Fig.9 Structure of the codebook
图 9 码本结构

解压缩是压缩的逆过程,我们首先构造基网格及 Huffman 表.基网格将经过以下步骤一层一层地不断得到精化:

(1) 用编码时的顶点排序方法对当前网格的顶点进行重排序;

(2) 读取顶点分裂参数(u,F,L–F)及校正信息;

(3) 从最后的分裂顶点开始,用(u,F,L)决定下一分裂顶点;

(4) 从比特流中读取新顶点的预测及校正信息,恢复出新顶点的位置;

(5) 分裂出新顶点并修改网格在分裂顶点的连接性.

当所有的分裂顶点都被处理以后,原始网格将完整地恢复出来.

5 实验与结果

本文的渐进压缩算法在 PIII 500 MHz 的 PC 机上用 C++实现.我们对该算法进行了充分的测试.测试图例如图 5 和图 10 所示.本算法的压缩速度约为每秒 20 000 个三角形,统计结果见表 1~表 4.在表中,dV 和 dT 分别表示顶点和三角形个数的增量,C/dT,G/dV 分别为每个三角形和顶点所占的平均比特数,C+G/dT 为每个三角形所占的总 bit 数.

在本文的算法中,每个三角形顶点连接信息所占的平均存储量小于 3.5 bit,小于 Taubin 等人提出的 PFS 模式的最小占用量.测试结果也优于 Bajaj 提出的渐进压缩模式的结果,顶点位置编码取决于模型的平滑度、顶点稠密度、顶点分布及指定的量化精度,经测试统计,我们采用 10 bit 的量化精度.以标准兔子模型为例,本文的压缩算法的压缩效率远优于 Bajaj 提出的算法.

(a) Deerhead model (a) 鹿头模型 (b) Horse Model (b) 马模型 (c) Dino Model (c) 恐龙模型

Fig.10 Some testing models
图 10 实验测试模型

Table 1 Compression results of Dino model
表 **1** Dino 模型的压缩结果统计

Dino	0	1	2	3	4	5	6	7	8
DV	131	48	65	90	131	179	253	354	475
DT	258	96	130	180	262	358	506	708	950
C/dT	–	3.35	3.49	3.45	3.42	3.49	3.39	3.36	3.48
G/dT	–	15.1	14.3	13.1	12.9	11.8	10.7	9.54	8.85
C+G/dT	–	18.5	17.8	16.6	16.3	15.2	14.1	12.9	12.3
Dino	9	10	11	12	13	14	15	Average	
DV	689	957	1 335	1 818	2 541	3 552	6 438	–	
DT	1 378	1 914	2 670	3 636	5 082	7 106	12 876	–	
C/dT	3.39	3.40	3.37	3.34	3.30	3.24	3.20	3.27	
G/dT	7.78	7.18	6.41	5.58	4.98	4.23	2.57	4.58	
C+G/dT	11.2	10.6	9.78	8.92	8.28	7.47	5.77	7.85	

Table 2 Compression results of Bunny model
表 2 Bunny 模型的压缩结果统计

Bunny	0	1	2	3	4	5	6	7	8	9
DV	127	31	46	67	97	143	189	269	353	501
DT	236	62	92	134	193	286	376	537	702	998
C/dT	–	3.69	3.46	3.63	3.53	3.58	3.44	3.57	3.51	3.52
G/dT	–	16.9	15.7	15.5	14.7	12.9	12.0	11.2	10.5	9.62
C+G/dT	–	20.6	19.2	19.1	18.2	16.5	15.4	14.8	14.0	13.1

Bunny	10	11	12	13	14	15	16	17	Average
dV	710	1 011	1 383	1 956	2 713	3 795	5 240	8 868	–
dT	1 412	2 016	2 760	3 903	5 411	7 573	10 460	17 680	–
C/dT	3.49	3.44	3.42	3.50	3.47	3.48	3.44	3.14	3.346
G/dT	8.89	8.26	7.62	7.00	6.46	5.86	5.32	4.09	5.670
C+G/dT	12.4	11.7	11.0	10.5	9.93	9.34	8.76	7.23	9.016

Table 3 Compression results of Horse model
表 3 Horse 模型的压缩结果统计

Horse	0	1	2	3	4	5	6
dV	249	73	111	146	206	316	425
dT	376	146	220	290	410	627	837
C/dT	–	3.70	3.52	3.56	3.44	3.33	3.45
G/dT	–	12.3	11.27	10.7	9.80	8.82	8.33
C+G/dT	–	16.0	14.8	14.3	12.2	12.2	11.8

Horse	7	8	9	10	11	12	Average
dV	610	847	1 181	1 596	2 248	2 904	–
dT	1 203	1 673	2 329	3 154	4 436	5723	–
C/dT	3.49	3.45	3.43	3.40	3.67	3.27	3.38
G/dT	7.49	6.83	6.33	5.71	5.16	4.74	6.04
C+G/dT	10.9	10.3	9.76	9.11	8.83	8.01	9.42

Table 4 Compression results of Deerhead model
表 4 Deerhead 模型的压缩结果统计

Deerhead	0	1	2	3	4	5	6	7	8	9	10	Average
dV	177	54	81	131	169	239	347	457	635	918	1205	–
dT	350	108	162	262	338	478	694	914	1270	1836	2410	–
C/dT	–	4.03	3.75	3.72	3.59	3.58	3.61	3.57	3.46	3.43	3.01	3.38
G/dT	–	11.3	11.2	9.80	8.89	8.43	7.93	6.94	6.56	6.00	5.31	6.68
C+G/dT	–	15.3	15.0	12.5	12.4	12.0	11.5	10.5	10.0	9.43	8.32	10.06

6 结 论

本文提出一种有效的渐进几何压缩算法,渐进网格(PM)的边折叠/顶点分裂序列在高效编码时被重新组织和排序.在每一个简化层次上,折叠边数目得到了自适应控制.基于独立点集,分裂顶点的分布尽量均匀,分裂顶点以递增序列有效编码.新顶点由分裂顶点及其邻点经 Laplacian 预测器预测.新顶点的校正经量化并用改进 Huffman 编码模式进行压缩编码.理论分析和实验结果表明,本文的算法可达到每个三角形 3.5 bit 的高压缩率.这一较好的特征保持网格简化保证了本文的压缩模式能很好地用于稠密网格在网络上实现渐进传送.

与单分辨率的网格压缩方法相比,多分辨率的压缩模式中需要额外的开销来存储各层次模型网格顶点的连接关系,尤其是在对整个模型进行传输时,这些开销是人们所不期望的.因此,如何减少这些额外存储开销值得进一步研究.网格几何信息是占用存储空间的主要因素,进一步的研究我们将注重更有效的对网格顶点几何坐标及属性数据的压缩.

References:

[1] Deering, M. Geometry compression. In: Robert, C., ed. Proceedings of the SIGGRAPH'95. Los Angeles: ACM Press, 1995. 13~20.

[2] Chow, M. Optimized geometry compression for real-time graphics. In: Proceedings of the IEEE Visualization'97. 1997. 346~354. http://home.earthlink.net/~mmchow/gcompiler/gcompiler.html.

[3] Taubin, G., Rossignac, J. Geometric compression through topological surgery. ACM Transactions on Graphics, 1998,17(2):84~115.

[4] Touma, C., Gotsman, C. Triangle mesh compression. In: Proceedings of the Graphics Interface'98. Vancouver, 1998. 26~34. http://www.graphicsinterface.org/proceedings/1998/.

[5] Taubin, G., Horn, W., Lazarus, F., *et al*. Geometric coding and VRML. In: Proceedings of IEEE Special Issue on Multimedia Signal Processing. 1998. 1228~1243. http://www.gvu.gatech.edu/~jarek/papers/vrml.pdf.

[6] Taubin, G., Rossignac, J. 3D geometric compression. In: SIGGRAPH'98 Course Notes 21. New York: ACM Press, 1998.

[7] Gumhold, S., Strasser, W. Real time compression of triangle mesh connectivity. In: Cohen, M., Zettzer, D., eds. Proceedings of the SIGGRAPH'98. New York: ACM Press, 1998. 133~140.

[8] Li, J., Kuo, C.C. A dual graph approach to 3D triangular mesh compression. In: Proceedings of the IEEE International Conference on Image Processing. 1998. http://www.cs.technion.ac.il/~ronir/courses/advancedTopics/pubs/.

[9] Rossignac, J. Edgebreaker: Connectivity compression for triangle meshes. IEEE Transactions on Visualization and Computer Graphics, 1999,5(1):47~61.

[10] Bajaj, C., Pascucci, V., Zhuang, G. Single resolution compression of arbitrary triangular meshes with properties. In: Proceedings of the Data Compression Conference. 1999. 247~256. http://www.ticam.utexas.edu/reports/1999/9905.ps.gz.

[11] Isenburg, M., Snoeyink, J. Mesh collapse compression. In: Proceedings of the SIBGRAPI'99—12th Brazilian Symposium on Computer Graphics and Image Processing. 1999. 27~28. http://www.cs.unc.edu/~isenburg/research/papers/.

[12] Kobbelt, L., Campanga, S., Vorsatz, J., *et al*. Interactive multi-resolution modeling on arbitrary meshes. In: Cohen, M., Zettzer, D., eds. Proceedings of th