ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.4, April 2008, pp.1016−1025
DOI: 10.3724/SP.J.1001.2008.01016

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

基于可变模版的三角网格拓扑压缩*

刘 迎[1,2+], 刘学慧[1,2], 吴恩华[1,2,3]

[1](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100080)

[2](中国科学院 研究生院,北京 100049)

[3](澳门大学 科学技术学院 电脑与资讯科学系,澳门)

Variable-Code-Mode-Based Connectivity Compression for Triangular Meshes

LIU Ying[1,2+], LIU Xue-Hui[1,2], WU En-Hua[1,2,3]

[1](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

[3](Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macao, China)

+ Corresponding author: E-mail: liuyingbj@yahoo.com

Liu Y, Liu XH, Wu EH. Variable-Code-Mode-Based connectivity compression for triangular meshes. *Journal of Software*, 2008,19(4):1016−1025. http://www.jos.org.cn/1000-9825/19/1016.htm

Abstract: This paper presents an efficient algorithm for encoding the connectivity information of triangular meshes. In the previous algorithms, Huffman or arithmetic coding method is directly used to encode operator series, but in comparison in this method, it can efficiently improve the compression ratio of connectivity information by predicting correctly the operator currently being encoded. By the method, all triangles are traversed first to obtain operator series. Then an arithmetic coder based on variable code-mode is applied to encode the operator series. According to the operator last encoded, the property of triangular mesh and the method of mesh traversal, a code-mode is calculated for each operator currently being encoded, where the operator with higher prediction probability is given a shorter binary strand. Then the binary strand can be obtained according to its code-mode and encode every bit of this binary strand by adaptive arithmetic coding method. The algorithm is a face-based method and also a single-resolution lossless compression method for manifold triangular mesh. Testing results show that the compression ratio of the algorithm is very high and even higher than the compression ratio by using TG algorithm, which is commonly regarded as one of the best in terms of compression ratio.

Key words: code mode; arithmetic coder; mesh; connectivity compression; encode; decode

摘 要: 针对三角网格模型的拓扑信息,提出了一种高效压缩方法.不同于以往的单纯利用算术编码或霍夫曼编码对遍历三角网格生成的拓扑流进行编码压缩,根据三角网格模型(特别是规则三角网格模型)的特点,自适应地提高编码过程中对当前编码字符发生的预测准确率,实现对三角网格模型的拓扑信息的高效压缩.算法首先遍历三角

* Supported by the National Natural Science Foundation of China under Grant Nos.60373051, 60173022 (国家自然科学基金); the National Basic Research Program of China under Grant No.2002CB312102 (国家重点基础研究发展计划(973)); the Research Grant of University of Macau of China (澳门大学研究基金)

Received 2005-12-30; Accepted 2006-06-01

网格模型,得到操作符序列;然后对得到的操作符序列的每个操作符作模版可变的自适应算术编码.在编码过程中,根据当前编码字符的前一个操作符、三角网格模型的特点以及网格遍历方法为当前编码操作符计算一个模版,在这个模版中,预测准确率高的操作符用较短的二进制串表示.根据当前编码操作符的可变模版,可以得到该操作符的二进制表示,并对这个二进制表示的每个比特作自适应算术编码.该方法是针对流形三角网格模型的拓扑信息作单分辨率的基于面的无损压缩,可以得到很好的三角网格拓扑信息的压缩结果,其压缩比甚至比拓扑压缩领域压缩比方面最好的 TG 算法的压缩比还要好.

关键词: 模版;算术编码;网格;拓扑压缩;编码;解码

中图法分类号: TP391 **文献标识码**: A

随着三维扫描技术和遥感技术的发展,大规模三维网格数据在各应用领域,如电子商务、医疗、科学计算可视化、工程分析和游戏等领域随处可见.具有庞大数据量的三维网格模型无疑为数据的交互显示和网络传输带来困难.如果没有有效的压缩算法,对于庞大的三维信息数据,其在网络上的传输时间是令人难以接受的.

网格由 3 种基本元素组成:顶点、边、面.基本元素所表达的信息分为拓扑信息和几何信息.几何信息用于描述网格顶点的三维坐标;拓扑信息用来描述这些顶点是如何形成一个个面、面又如何构成三维网格模型表面.三维网格压缩算法针对不同特性有几种不同的分类方法,如几何压缩与拓扑压缩、渐进(多分辨率)压缩与非渐进(单分辨率)压缩、三角网格压缩与非三角网格压缩等(有兴趣的读者请阅读文献[1,2]).网格压缩算法要分别或同时压缩两种数据类型(几何和拓扑),本文研究的是单分辨率的拓扑压缩算法,对于网格的几何压缩算法,请参考文献[3–7].

三角网格的拓扑压缩可以追述到图形函数语言中三角带的概念.随着大规模三角网格模型的普及,人们将这一概念进一步扩展,提出了基于区域的网格拓扑信息压缩方法.根据压缩过程中主要操作单元类型的不同,分为基于顶点[4,8]、基于边[9]、基于面[10−15]等不同类型.所有这些算法最后都生成具有一定信息的字符序列,并利用各种编码方法对该字符序列进行编码压缩,较早算法中多应用霍夫曼编码[16],而最近几年的算法多应用算术编码[17],因为算术编码的压缩结果可以达到一个序列的熵值,好于霍夫曼的压缩结果.

基于顶点的拓扑压缩算法:Touma等人[4]提出的针对三角网格拓扑信息作单分辨率无损压缩的算法,是目前为止压缩效果非常好的算法之一.该算法沿螺旋顶点树遍历编码每个顶点的邻接三角形个数,分支处需记录额外的信息,最后用算术编码压缩上述遍历结果序列.对规则的三角网格,该算法压缩效果极好.

基于边的拓扑压缩算法:该类算法需记录对两种操作对象的各种不同操作——一类是对面的操作,如F或H用来表示增加一个面片;一类是对边的操作,如R,L,S,M等用来确定新的当前边.这类算法与基于面的拓扑压缩方法具有很大的相似性.Isenburg等人[9]提出的算法就是这类算法.

基于面的拓扑压缩算法:Rossignac等人[15]提出的Edgebreaker算法针对三角网格拓扑信息作单分辨率压缩,是一种经典的算法.该算法提出后,有多种算法[11−14]对其作了改进.EdgeBreaker算法通过遍历三角网格模型的每个三角形后,获得由CLERS这 5 个字符组成的字符串序列,然后用霍夫曼编码方法编码这个字符串序列.Isenburg等人[13]其后针对Edgebreaker算法提出了一种新的解码方法,该方法避免了Edgebreaker算法解码时的二次遍历问题.但算法[13]仍然存在缺陷,它必须在接受全部编码信息后才能够开始解码,因此不适于网络在线传输的解码应用.Jong等人改进了Edgebreaker算法的压缩过程[14](简称JBS算法),用“J”替代很影响压缩结果的“S”操作,使编码字符由CLERS改变为QCRLJ,并且使得解码简单而且是一次遍历.该算法用来压缩字符序列的方法由霍夫曼编码方法改为对 5 个基本字符组成的序列直接作自适应算术编码,极大地提高了压缩比.

观察到JBS算法[14]遍历三角网格模型得到的操作符之间固有的相关性以及三角网格模型的特点,本文在JBS算法基础上利用这种相关性以及三角网格模型的特点,根据前一个操作符尽量准确地预测下一个操作符,然后使用基于上下文的算术编码来充分地利用这种高准确率的预测以达到提高压缩比的目的.与JBS算法相比,本文算法得到的压缩比比JBS算法得到的压缩比有很大提高.在我们的实验模型中,最高可提高到 79%,大部分模型的压缩比提高超过 30%(见第 3 节实验结果部分).与JBS算法一样,本文算法一次遍历解码,适合在线应

用.而且到目前为止,JBS算法无论在压缩比方面,还是在编码/解码方面都是一种效果非常好的算法.

本文第 1 节介绍与本文算法相关的工作,我们将详细讲解 Jong 在 2005 年提出的 JBS 算法以及算术编码概念.第 2 节给出本文基于可变模版的网格拓扑压缩算法.第 3 节给出实验结果.第 4 节为结论.

1 相关工作

Jong等人提出的JBS算法[14]是对Edgebreaker算法的改进,也是基于面的针对三角网格拓扑信息的单分辨率压缩算法.这类基于面的算法是基于区域增长原理的:在遍历网格过程中要始终维持一个由边组成的有向边界,这个边界把网格分成已遍历部分和未遍历部分,然后每遍历编码一个多边形,则输出一个该多边形和边界的拓扑关系操作(符),并把该多边形归入已编码部分.为了下面叙述方便,我们把边界上正在处理的边定义为当前边.对于三角形网格,与当前边相邻的未遍历三角形定义为当前三角形,其不在当前边上的顶点定义为第三顶点.

Edgebreaker 算法和 JBS 算法的边界都采用逆时针方向.Edgebreaker 算法使用 S 操作把图形分成两部分,同时需要用额外的偏移或其他操作记录分支信息.JBS 算法引用新的操作 J 直接把当前边在当前边界上的下一条边作为新的当前边,同时不编码当前三角形.该方法避免了把图形分割成两部分这种现象.JBS 算法使用 5 个字符 Q,C,R,L,J 表示在编码和解码三角网格模型过程中对三角面片的不同的操作方法.其中,CRL 操作来自 Edgebreaker 算法:C 操作表示第三顶点不在边界上的拓扑情况;L,R 操作表示第三顶点在边界上且当前三角形除了当前边外还有一条边(e)在边界上,L,R 分别表示 e 在当前边的不同方向;Q 操作合并了 Edgebreaker 算法的 CR 两个操作;J 操作替代了 S 操作,与 S 操作有很大区别:J 操作表示当前边在当前边界上的下一条边为新的当前边,执行了“跳”操作.J 操作中的当前三角形不被编码(跳过),以后再遇到这个被跳过的三角形时再编码它.图 1 示例了 QCRLJ 操作,其中灰色部分为网格已遍历部分,白色部分为网格未遍历部分,灰色和白色的分界线为边界;边界上带箭头的粗实线为当前边;灰色区域外的三角形为当前三角形,其被遍历后归入灰色部分;实心大圆点为第三顶点;不在边界上的带箭头的粗实线为下一个当前边.图 2(来自文献[14])为该算法对示例图形的压缩过程,细箭头线表示压缩顺序,最后得到 27 个操作字符:CQQJRLRCJQ QRRLLLRQQQ RRLLRLR.

JBS 算法对遍历网格得到的操作字符序列直接采用自适应算术编码压缩,较充分地利用了模型局部的相似特征,相比霍夫曼编码压缩方法(Edgebreaker 算法使用的压缩操作符序列的方法),极大地提高了压缩比.该算法可以一次遍历解码,解码与编码顺序相同.

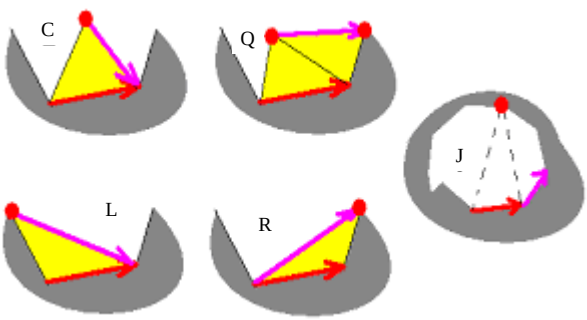

Fig.1 The operators: QCRLJ

图 1 QCRLJ 操作示例图

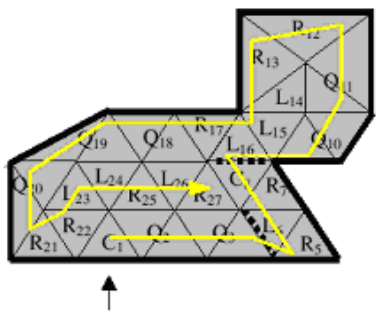

Fig.2 The compression process of JBS

图 2 JBS 算法压缩过程示例

本文提出的算法改进了 JBS 算法的压缩算法.本文算法获得的操作序列与 JBS 算法是完全一样的,不同的只是对操作序列的压缩方法.对遍历网格后得到操作序列中的每个操作,我们根据其前一个操作以及遍历三角网格的固有属性来计算一个新的模版.根据这个新的模版,我们确定当前操作符的二进制表示,然后对二进制表示的每个比特作自适应算术编码.本文算法的压缩结果好于 JBS 算法.本文算法提供的解码方法是一次遍历方

法,解码顺序与编码顺序相同.

2 基于可变模版的网格拓扑压缩算法

本文提出的算法充分利用了相邻操作符之间的相关性和三角网格模型的共性,提出了一种针对流形三角网格的拓扑压缩算法.该算法分两步完成:1) 利用区域增长方法遍历网格的所有三角形,记录遍历过程中的每个操作(符).遍历完成后,得到操作符序列;2) 对操作符序列的每个操作符作基于可变模版的自适应算术编码:在编码每个操作符之前,先计算模版;然后根据模版得到要编码操作符的二进制表示序列;对二进制表示序列的每个比特作自适应算术编码.

2.1 获取操作符序列

本文直接利用了JBS算法[14]的三角网格遍历方法来获取操作符序列.这种遍历方法有很多优点:一次遍历编码和解码;解码方向与编码方向相同,处理机制简单,可以解码一个三角形后,马上抛弃它,适用于在线传输和解码的实时与交互应用场合;解码过程中占用内存少;线性时间复杂度等.本文的编码算法不破坏这种遍历方法,也就同样具备了这些优点.

JBS 的网格遍历可简述如下:

1) 对于一个封闭的网格模型,可任选一个三角形形成有向初始边界(对于未遍历网格部分来说,边界为顺时针),并且指定边界上的一条边为当前边.该面片标记为已压缩.

2) 寻找包含当前边、但未被压缩的三角形,并对该面片作压缩操作(标记该面片为已压缩,记录操作符为 QCLR 中的一个——见本文相关工作部分)或跳过该面片(继续标记该面片为未压缩,记录操作符为 J),同时更新边界和当前边.

3) 重复步骤 2),直到所有面片都标记为已压缩,算法遍历网格过程结束,此时得到 JBS 算法的操作符系列.

接下来是熵编码部分.本文算法采用基于可变模版的自适应算术编码,该编码方法与 JBS 的编码压缩算法有很大的不同,应用本文算法得到的压缩比相比于 JBS 算法得到的压缩比有很大提高.

2.2 操作符之间的相关性

首先遍历网格模型得到操作符序列,然后直接采用算术编码(早几年采用霍夫曼编码,其压缩效果在绝大多数情况下不如算术编码的压缩效果好)压缩操作符序列,这是近几年有关拓扑压缩算法的最常用的过程,其得到的压缩结果被认为已经是(及其接近)熵理论值,所以这几年对拓扑压缩算法的改进多集中在遍历网格获取操作符序列部分.然而,我们观察到,操作符之间具有一定的相关性,且三角网格模型本身具有一定的规律.如果能够利用具体某种遍历网格算法本身的特征,找出操作符之间的相关性,提高编码过程中预测下一个操作符的准确率,我们就可以提高(甚至很大幅度地提高)压缩比.

通过分析 JBS 算法中的遍历三角网格方法和三角网格模型本身,我们发现了以下一些特点可以被用来有效地提高压缩比:

1) 对于封闭的流形三角网格模型,每个顶点的邻接面个数平均约为 6.对于规则的网格,绝大多数顶点的邻接面个数为 6;对非常不规则的网格模型,顶点的邻接面个数有很大的变化范围.而在我们常用的网格模型中,非常不规则的网格模型所占比例非常小,大部分介于规则和半规则之间.

2) 根据 JBS 算法的遍历网格方法工作原理,有些操作符是不可能前后邻接的(或者出现的概率极低),如 C 后面不可能邻接 L 或 R(CR 即为 Q 操作符).J 或 Q 的后面不可能或非常少地出现 L.示例如图 3 所示,其中边界上的粗线为当前边;非边界上的粗线为下一个当前边;粗虚线为不可能出现的下一个要遍历的三角形;黑色细虚线为当前三角形,但暂不编码它,以后的遍历还会遇到它,此时只记录 J 操作符,同时改变当前边.

在 JBS 算法遍历三角形的过程中,我们可以根据前一个操作符以及特点 1)中的属性来预测随后的操作符(关于如何预测,见第 2.3.2 节).当然,预测不可能百分之百地准确.根据我们的实验,对于这种操作符的预测正确

率,绝大部分网格模型可达到60%以上,大部分网格模型在80%左右,高的甚至达到97%.预测准确率具体数值见第3节实验结果部分.

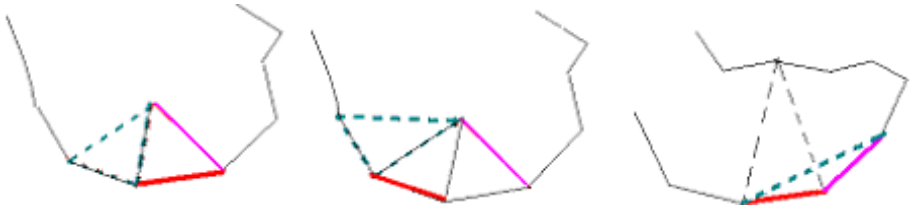

Fig.3 The impossible neighboring operators

图3 操作符间不可能出现的情况示例

2.3 基于可变模版的自适应算术编码

2.3.1 本文压缩算法思想的来源

遍历得到的操作符序列的每个操作符有各自的概率分布属性.对于给定具体序列,采用某种优化编码方法对其编码后,每个操作符平均占用的比特数定义为熵.熵可以为小数.这个熵值有一个理论值[1]: $-\sum_{i=1}^{n} p_i \cdot \log_2 p_i$,其中,n为操作符种类,P_i为第i种操作符的概率, $\sum_{i=1}^{n} p_i = 1$.如果只有具体序列,没有任何其他信息,则上述熵理论值是能够达到的最好的压缩结果.整数方法实现的算术编码[17]的压缩结果可以达到这个熵理论值[1].

一个序列有一个熵值,如果只有序列的本身信息,则熵是能够达到的最好的压缩比数值.所以,如果要提高压缩比,除了操作符序列本身以外,还需要利用其他信息.这也是本文算法的核心内容:充分利用遍历三角网格算法本身的属性和相邻操作符之间的相关性来更高效地压缩操作符序列.充分利用这些特点,为要压缩的每个操作符制订了一个可变模版.

2.3.2 可变模版的计算

利用三角网格模型的共性和前一个操作符,我们可以为每个要编码的操作符做一个预测,预测操作符按可能的概率从大到小排序.各种预测结果见表 1.其中,pre为前一个操作符,f为预测的当前边的轴顶点(pivot是指有方向的当前边,其方向由轴顶点指向非轴顶点)[1]的邻接面个数;f_0,f_1分别为当前边的非轴顶点、轴顶点已经编码的邻接面个数.在表 1“Other condition”列中用了“<=”而不是“=”,是因为f是轴点的预测值,而不是实际的邻接面个数.

Table 1 All predicted codes

表 1 预测编码枚举

Pre	Other condition	The predicted operators sorting by the probability from maximum to minimum					Impossible operators	
Q	$f-f_1=2$	Q	C	R	J		L	
	$f-f_1<=1$	R	Q	C	J		L	
	$f-f_1>2$	C	Q	R	J		L	
L or R	$f-f_0<=1$	L	The others are as same as the following three rows according to f and f1					
	$f-f_1=2$	Q	C	R	L	J		
	$f-f_1<=1$	R	Q	C	L	J		
	$f-f_1>2$	C	Q	R	L	J		
J	$f-f_1=2$	J	Q	C	R		L	
	$f-f_1<=1$	J	R	Q	C		L	
	$f-f_1>2$	J	C	Q	R		L	
C	$f-f_1=2$	Q	C	J			R	L
	$f-f_1>2$	C	Q	J			R	L

表 1 中的 f 值是一个预测值,我们采用了两种方法计算 f 值:1) f 采用固定值 6;2) f 值动态变化,等于在遍历三角形过程中,刚刚由边界上的顶点转变成内部顶点的那个点的邻接面个数(对于第 2 种动态预测方法,我们是根据网格模型的局部相似性直观上得到的).这两种 f 值的不同计算方法会导致最后的压缩比不同.对这两种方法,我们都作了实验,结果见第 3 节.

图 4 给出前一个操作符是Q时,根据f与f_1的关系得出的下一个可能性最大的操作符的示例.图 4 中,f_1为轴顶点的已编码邻接面个数,图示f_1=4;f_0为当前边非轴顶点已编码邻接面个数,图示f_0=2;Q是刚刚编码的前一个操作符;每个小图中右侧的操作符是预测的下一个可能性最大的操作符;黑色的细实线是当前边界;粗实线是当前边;细虚线是已编码的三角形的边;粗虚线为预测的可能性最大的三角形拓扑情况.

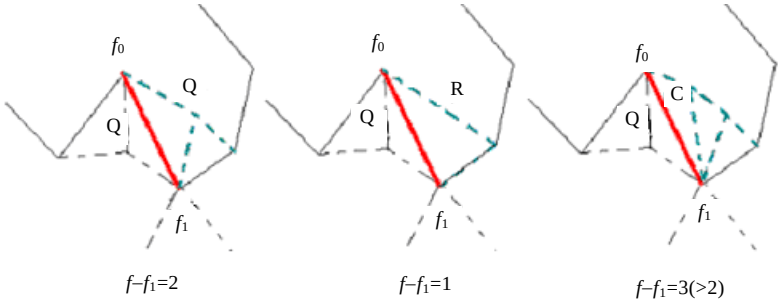

Fig.4 Illustration of the prediction of an operator that its previous operator is Q

图 4 前一个操作符是 Q 时,预测下一个操作符的示例图

1) 如果 f 值采用 6(对于规则的流形三角网格,绝大多数顶点的邻接面个数等于 6),则图 4 中各种情况预测完全正确,其左、中、右图分别对应表 1 中第 3 列的第 2~第 4 行的情况.
2) 考虑到轴顶点的所有邻接面个数可能不是 6(等于 6 的可能性最大),其为 5(图 4 中间图)或 7(图 4 最右边图)的可能性仅次于 6.此时,虽然$f-f_1$仍然等于 2,但其实际的操作符分别为R和C.所以,对于表 1 第 2 行的情况,Q可能性最大,其次是R或C.至于J操作符,对于大多数模型来说出现概率不大,在我们的模版中,一般都把J排在后面,仅位于不可能的操作符的前面.所以,针对表 1 第 2 行的情况,本文采用的模版为QCRJL.
3) 对于$f-f_1$=1 的情况(对应表 1 第 3 行),毫无疑问,如果f值(预测值)等于该轴顶点的实际邻接面个数,则操作符必然为R;如果f值不等于该轴顶点的实际邻接面个数,则其与实际值相差为 1 的可能性最大(即于$f-f_1$=2,对应图 4 最左边的图),再次为相差 2 及以上(即于$f-f_1$>=3,对应图 4 最右边的图).所以,对于表 1 第 3 行的情况,本文采用的模版为RQCJL.
4) 对于$f-f_1$>2 的情况(对应表 1 第 4 行),同样,如果f值(预测值)等于实际值,则操作符必然为C;如果f值不等于实际值,则f与实际值相差为 1(于$f-f_1$=2,对应图 4 最左边的图)的可能性最大,其次是相差为 2(于$f-f_1$=1,对应图 4 中间图).所以,对于表 1 第 4 行的情况,本文采用的模版为CQRJL.
5) 对于表 1 的其他行模版,本文采用类似的方法计算得到.

对于上述方法得到的模版,可以保证:即使第一个预测操作符不等于编码操作符 Q,我们也能尽量地使排在第 2 位的预测操作符和编码操作符一致,排在第三、第四的操作符意义也是如此,总之,排号越靠前的操作符,其与被编码操作符一致的可能性越高.

表 1 中根据前一个操作符,对下一个操作符的每种可能性都作了枚举.所有情况下,一个操作符最多 5 种可能,最少 3 种可能.我们对最多 5 种可能的情况按可能性从大到小为每个预测操作符分配二进制串,分别为 0/100/101/110/111(至于为何采用这样的二进制串来表示操作符,我们将在第 2.3.3 节中提到).例如:对于前一个操作符是Q,且$f-f_1$=2 的情况(表 1 第 2 行),我们分配各操作符的二进制串分别为Q(0),C(100),R(101),J(110),L(111).由此,每个操作符在对其编码之前,都可根据表 1 中的某一行为其确定一个模版.根据这个模版,我们随后对这个操作

符作自适应算术编码.

2.3.3 自适应算术编码

在编码一个操作符之前,我们通过上面提到的方法为其计算了一个模版.根据这个模版,我们可以很容易地知道要编码操作符的二进制表示.如我们要编码 Q,对上面提到的模版,其二进制表示为 0(则 C 为 100,其他略).而我们要编码的实际上是这个二进制表示,是对其每个比特作算术编码.

算术编码过程涉及到的具体内容有:

1) 对每个比特作自适应算术编码.

2) 自适应算术编码采用整数实现方法[17].

在这种算术编码中,下一个操作符预测正确与否直接决定了最后的压缩结果.如果预测正确,我们只对该操作符编码 1 个比特 0;预测错误,我们就要对该操作符编码 3 个比特,包括 0 和 1.因为我们始终要尽量使被编码的比特为 0,所以,如果预测得不准确,可能性排第 2 位的操作符的二进制表示应该采用尽可能多的 0 和尽可能少的 1,模版的其他操作符以此类推.又因为本文算法对下一个操作符的预测准确率大多情况下很高,所以对预测正确的操作符我们优先采用了最简单的表示方式——只有一个 0 比特.综上所述,我们根据预测准确率的概率由大到小排序各操作符后,其相应的二进制表示分别为 0/100/101/110/111.

我们的模版尽量采用 0 比特作为要编码的原始值,且尽量采用更少的位数,再加上我们的对下一个操作符的高准确率预测(见第 3 节),所以我们实际上是对大部分(绝大部分)0 比特在作算术编码,0 所占的比例和操作符的预测准确率相差不多,对大部分网格模型来说,其值在 80%左右(见第 3 节).对这样的基于可变模版得到的序列作自适应算术编码,编码结果接近这个序列的熵值,远高于操作符序列的熵值.

算术编码结束后,就得到了本文算法的最终压缩结果,见第 3 节.实际上,本文采用的这个自适应算术编码也可以这样描述:每个操作符根据其各自的模版先输出二进制表示,这样就可以得到所有操作符的二进制表示流,再对这个二进制表示流作自适应算术编码.在这里,用另外的语言再描述一遍是为了读者能够更好地理解本文算法的核心内容——基于模版的算术编码部分.

2.4 解 码

解码过程由编码过程决定,所以 JBS 编码过程所决定的其解码的优点,本文算法也都具备.这里特别需要提及的是,我们的基于可变模版的自适应算术编码并不影响 JBS 解码过程,只是有些不同而已.

在压缩某一个操作符时,我们要根据前一个操作符、当前边两个顶点已经编码(压缩)的邻接面个数以及预测的当前边轴点的所有邻接面个数 f(两种预测方法中任选其一)计算一个模版.而计算模版的这些信息对于解码器来说也是已知的,解码时可计算同样的模版.

算术编码/解码时得到的是一个个 0 或 1 比特.因为编码时只编码了 5 个固定的二进制串(0/100/101/110/111),所以当解码一个比特时,如果是 0,则直接根据当时的模版得到相应的操作符;如果是 1,则接着再解码两个比特,连同前面的 1 组成 3 个比特的二进制串,再比照当时的模版,找到对应的操作符.此时,根据出现的操作符,可以直接利用 JBS 算法的解码方法,解码一个三角形或只改变当前边,同时还要根据这个新解码的操作符来计算下一个模版.以此类推,可以很容易地正确解码整个操作符序列.解码算法具有一次遍历、所用空间少、解码快、适合在线应用等优点.

2.5 时间和空间复杂度分析

本文算法具有线性时间和空间复杂度.

遍历网格后得到的操作符个数一般情况下会少于面片个数,但二者相差不多,在我们的实验模型中,相差最多不超过两倍(具体数值参见第 3 节实验结果部分).每个操作符最多 3 个二进制数值(大部分为 1 个二进制数值 0),每个二进制数值的自适应算术编码消耗的时间是一个常数,并且压缩每个操作符前的模版的计算(见表 1)也耗费了常量时间.所以,本文的基于可变模版的自适应算术编码的时间复杂度是 $O(f)$,其中,f 为网格模型的三角形数量.

本文算法模版的计算利用了前一个操作符、已编码操作符的二进制表示的 0/1 统计数据,这两部分所用空间为很小的常量.计算模版和遍历网格都需要在内存维持一个链表,一般情况下,这个链表与网格模型结构和大小有关,但链表长度会远远小于网格的顶点个数.本文算法最主要的内存消耗就是要把网格模型整个地装入内存,包括所有顶点和面片.所以本文算法的空间复杂度为 $O(v+f)$,其中,f 为网格模型的三角形数量,v 为网格模型的顶点数量.

3 实验结果

我们在普通 PC 机上实现了 JBS 算法和本文算法,实验用的部分图形如图 5 所示.实验结果见表 2,其中,#*V* 为网格顶点个数,#*F* 为网格三角形个数,#*T* 为所有操作符⟨QLRJC⟩个数,%*T* 为操作符⟨QLRJC⟩预测正确百分比,#*E* 为本文算法算术编码前 01 比特流个数,%*E* 为本文算法算术编码 01 流时 0 所占百分比,LY 为本文算法的压缩结果,%*H* 为本文压缩结果 LY 比算法 JBS 压缩结果提高的百分比.其中,各符号后缀 0 表示 f=6 作为预测值(表中列名称最后字是 0 的列),而后缀 1 表示 f 等于刚刚转换成内部点的顶点的邻接面个数作预测值(表中列名称最后字是 1 的列).此外,本文实验模型是随机选择的常规三角网格模型,没有特别选择规则模型进行测试,所以没有压缩比非常高的实验模型,比如没有压缩比小于 0.10bpv 的实验模型.

Table 2 Compression result of test meshes (b/v)

表 2 拓扑压缩实验结果 (b/v)

Graph	#*V*	#*F*	#*T*	%*T*0	%*T*1	En-Tropy	JBS	*LY*0	*LY*1	%*H*0	%*H*1	#*E*0	#*E*1	%*E*0	%*E*1	TG	%*TG*
AnimalPerson	6 540	12 584	7 676	96	95	1.23	1.24	0.43	0.46	65	63	8 372	8 476	94	93	0.49	13
Apple	867	1 704	964	84	90	1.22	1.24	0.93	0.56	25	55	1 266	1 152	84	91	0.81	31
Bolt	328	652	519	67	56	3.36	3.42	2.37	2.76	31	19	859	973	69	66	2.69	12
Bunny	34 835	69 474	46 075	78	70	2.28	2.28	1.45	1.75	36	23	66 629	73 701	78	74	1.82	20
Cow	2 897	5 504	4 675	70	66	3.37	3.38	2.22	2.38	34	30	7 485	7 837	72	70	2.59	14
Dice	2 253	4 502	3 490	65	67	3.23	3.25	2.37	2.15	27	34	5 942	5 786	69	73	1.88	-13
Eve-head	2 748	5 424	4 229	65	57	3.19	3.20	2.31	2.61	28	18	7 187	7 829	70	67	2.81	18
Fish	1 234	2 460	1 561	90	86	2.02	2.04	0.87	1.03	57	50	1 889	2 005	87	84	1.04	17
Fish01	19 457	36 701	30 119	72	67	3.11	3.11	1.96	2.15	37	31	47 175	50 017	75	73	2.33	16
Foot	2 154	4 204	2 792	73	73	2.33	2.34	1.69	1.59	28	32	4 320	4 278	73	76	1.91	16
Gobo	31 856	63 684	39 444	77	73	1.89	1.85	1.34	1.45	28	22	57 258	60 492	78	76	1.49	10
Gourd	326	648	497	64	55	3.14	3.19	2.36	2.72	26	15	851	947	69	65	2.43	3
Iceberg	1 726	3 106	2 499	90	90	1.27	1.28	0.75	0.76	41	41	2 987	2 995	87	87	0.88	15
Muskox	8 639	15 523	10 811	87	86	2.21	2.21	0.97	1.06	56	52	13 625	13 877	85	84	0.94	–3
Myhead	510	923	965	57	54	4.08	4.11	3.31	3.47	19	16	1 787	1 861	65	63	3.48	5
Output	14 039	20 870	20 685	78	84	2.64	2.64	1.61	1.24	39	53	29 913	27 241	78	84	1.42	13
Padlock	1 736	2 798	3 068	74	74	3.33	3.35	2.19	2.21	35	34	4 696	4 692	75	75	2.54	14
Poli_game	1 004	1 998	1 756	48	43	3.89	3.90	3.44	3.63	12	7	3 576	3 744	61	60	3.71	7
Rhino	229	454	432	51	52	4.26	4.33	3.63	3.60	16	17	854	850	61	63	3.35	–7
Surface	660	1 184	1 043	85	85	2.76	2.79	1.41	1.42	49	49	1 351	1 357	82	82	1.40	–0
Trashbin	2 801	5 594	8 298	80	81	3.08	3.09	2.06	2.03	33	34	11 550	11 496	78	78	1.67	–18
Triceratops	2 835	5 660	4 596	55	48	3.48	3.49	2.86	3.15	18	10	8 712	9 402	65	63	2.88	1
Tw_goku	4 549	8 604	7 623	68	60	3.59	3.59	2.43	2.74	32	24	12 567	13 705	70	68	2.71	10
Vase	46 335	92 263	51 830	94	94	1.15	1.13	0.43	0.45	62	60	57 614	58 536	93	93	0.49	11
Winegoble	502	1 000	563	96	97	1.15	1.20	0.40	0.25	67	79	611	597	94	97	0.35	28

实验中,我们给出了根据操作符序列计算得到的理论熵值(表 2 左数第 7 列“JBS 熵”).由表 2 可以看出,直接使用自适应算术编码来编码操作符序列得到的压缩结果(表 2 左数第 8 列“JBS”)与理论熵值极其接近,一般误差不超过 2%.如果不利用操作符序列本身以外的信息,已经不能提高压缩结果了.

我们的算法采用可变模版计算操作符序列的二进制表示流,然后对这个流作算术编码.根据我们的可变模版,预测正确的操作符只编码 1 个 0 比特,而我们对操作符预测正确的比例(表 2 中左数第 5 列、第 6 列“%*T*0”,“%*T*1”)非常高,所以,我们最后被压缩的 0/1 比特流个数(#*E*0,#*E*1,右边数第 4 列、第 3 列)比操作符个数(#*T*,左边数第 4 列)没有多多少,在我们的实验模型中,最多的也没有达到操作符个数的两倍,且大部分为 0 比特(0 所占百分比就是表 2 中最右边的两列);最少(Winegoble 模型)压缩的比特流个数只比操作符个数多 6%,且约 97%是 0,只有约 3%为 1.由此可见,对这样的 01 序列作编码压缩,自然压缩比会非常高.如果采用算术编码压缩这样的序

列,压缩结果会接近序列的熵理论值(误差一般不会超过 2%).也正因如此,本文算法采用算术编码压缩这个 01 流序列.

采用本文压缩方法得到的压缩结果(表 2 中右数第 7 列、第 8 列“*LY*0”,“*LY*1”)比 JBS 算法得到的压缩结果(表 2 中左数第 8 列“JBS”)要高很多,对我们实验的大部分模型来说,压缩比提高超过 30%(表 2 中右数第 5 列、第 6 列“%*H*0”,“%*H*1”,取二者中较大的数据),最高达到 79%.

对于 f 的两种不同预测方法,实验结果表明,两种方法得到的结果(表 2 中列名称后缀分别为 0 和 1)是有差别的.对不同的网格模型,总有一种预测方法更合适.如果想要较好的压缩结果,可以实验两种预测方法,选择压缩结果好的那一种.同时,在编码文件头增加一个比特来记录使用的预测方法.

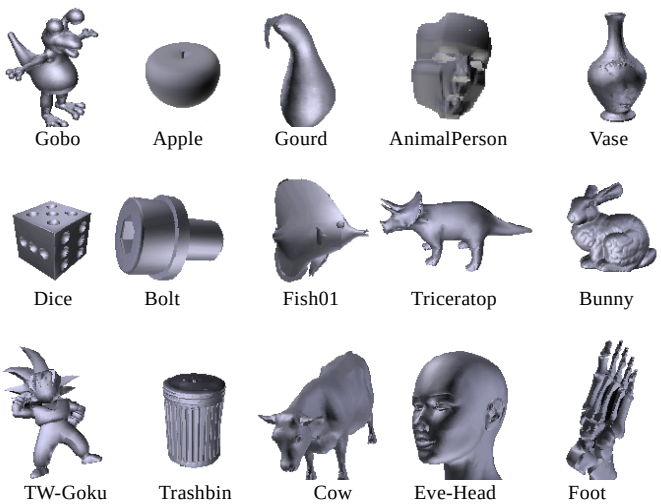

Fig.5 Partial test models

图 5 实验用部分模型

4 结 论

本文算法对 JBS 算法操作符序列的熵编码部分作了改进.新算法利用三角网格模型的特点和遍历三角网格方法(JBS 算法)所固有的属性来尽可能准确地预测被编码操作符,并按照预测概率的大小排序可能的操作符,从而得到被编码操作符的模版.根据模版,可以计算被编码操作符的二进制表示,并采用算术编码来压缩这个二进制表示的每个比特.本文算法相比 JBS 算法,在绝大部分情况下得到的压缩比能得到极大的改善.对于本文的大部分实验模型,采用本文算法得到的压缩比甚至比在网格拓扑压缩领域压缩比最好的算法之一——TG 算法的压缩比还要高.

我们的算法还有一些有待改进的地方,特别是要改进应用于顶点的邻接面个数有很大变化的非常不规则的三角网格模型的情况.另外,为其他网格遍历方法设计有效模版方面也需要加以研究.

References:

[1] Gotsman C, Gumhold S, Kobbelt L. Simplification and compression of 3D meshes. In: Iske A, Quak E, Floater MS, eds. Proc. of the Tutorials on Multiresolution in Geometric Modelling. New York: Springer-Verlag. 2002. 319−361.

[2] Peng JL, Kim CS, Kuo CCJ. Technologies for 3D mesh compression: A survey. ELSEVIER Journal of Visual Communication and Image Representation, 2005,16(6):688−733.

[3] Taubin G, Horn W, Lazarus F, Rossignac J. Geometry coding and VRML. Proc. of the IEEE, 1998,86(6):1228−1243.

[4] Touma C, Gotsman C. Triangle mesh compression. In: Proc. of the Graphics Interface. New York: ACM Press, 1998. 26−34. http://www.graphicsinterface.org/proceedings/1998

[5] Liu XG, Bao HJ, Peng QS. Incremental geometric compression. Journal of Software, 2000,11(9):1167−1175 (in Chinese with English abstract).

[6] Qin XJ, Liu XG, Bao HJ, Peng QS. Progressive geometry compression for meshes. Journal of Software, 2002,13(9):1804−1812 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/1804.pdf

[7] Deering M. Geometry compression. In: Robert C, ed. Proc. of the Siggraph'95. Los Angeles: ACM Press, 1995. 13−20.

[8] Alliez P, Desbrun M. Valence-Driven connectivity encoding for 3D meshes. In: Magnenat-Thalmann N, ThalMann D, eds. Proc. of the Eurographics. Manchester: Springer-Verlag, 2001. 480−489.

[9] Isenburg M, Snoeyink J. Face fixer: Compressing polygon meshes with properties. In: Proc. of the Annual Conf. Series, ACM SIGGRAPH 2000. New Orleans, 2000. 263−270.

[10] Gumhold S, Strasser W. Real time compression of triangle mesh connectivity. In: Proc. of the Annual Conf. Series, ACM SIGGRAPH 1998. New York: ACM Press, 1998. 133−140.

[11] Rossignac J, Lopes H, Safanova A, Tavares G, Szymczak A. Edgebreaker: A simple compression for surface with handles. Computers & Graphics Int'l Journal, 2003,27(4):553−567.

[12] Szymczak D, King D, Rossignac J. An EdgeBreaker based efficient compression scheme for regular meshes. In: Proc. of the 12th Canadian Conf. on Computational Geometry. Fredericton, 2000. 53−68.

[13] Isenburg M, Snoeyink J. Spirale reversi: Reverse decoding of EdgeBreaker encoding. In: Proc. of the 12th Canadian Conf. on Computational Geometry. Fredericton, 2000. 247−256.

[14] Jong BS, Yang WH, Tseng JL, Lin TW. An efficient connectivity compression for triangular meshes. In: Proc. of the 4th Annual ACIS Int'l Conf. on Computer and Information Science (ICIS 2005). Las Vegas: IEEE Computer Society, 2005. 583−588.

[15] Rossignac J. EdgeBreaker: Connectivity compression for triangle meshes. IEEE Trans. on Visualization and Computer Graphics, 1999,5(1):47−61.

[16] Huffman DA. A method for the construction of minimum-redundancy codes. Proc. of the IRE, 1952,40(9):1098−1101.

[17] Witten IH, Neal RM, Cleary JG. Arithmetic coding for data compression. Communications of the ACM, 1987,30(6):520−540.

附中文参考文献:

[5] 刘新国,鲍虎军,彭群生.增量几何压缩.软件学报,2000,11(9):1167−1175.

[6] 秦绪佳,刘新国,鲍虎军,彭群生.网格的渐进几何压缩.软件学报,2002,13(9):1804−1812. http://www.jos.org.cn/1000-9825/13/1804.pdf

刘迎(1970－),女,辽宁锦州人,博士,副研究员,主要研究领域为网格压缩.

刘学慧(1968－),女,博士,副研究员,主要研究领域为计算机图形学.

吴恩华(1947－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为计算机图形学,科学可视化.