基于 Marching Cubes 重组的外存模型渐进压缩

刘 迎[1)] 蔡康颖[1)] 王文成[1)] 吴恩华[1),2)]

[1)](中国科学院软件研究所计算机科学重点实验室 北京 100080)

[2)](澳门大学科学技术学院电脑与资讯科学系 澳门)

摘 要 外存模型是指其规模远远超出内存容量的海量模型.为提高其存储、传输、显示等操作的效率,对外存模型进行渐进式的压缩是非常重要的.但当前已有的外存模型压缩算法都是单一层次的,不能做到渐进压缩.为此,该文提出一种针对外存模型的渐进压缩方法,能高效地压缩外存模型,并进行多分辨率的传输和显示.该方法首先将外存模型的包围盒空间按照八叉树形式进行剖分和层次化组织,使得最精细层次的各个立方块空间中的局部模型都能完全装入内存进行处理;然后,在各个立方块中对局部的模型进行基于 Marching Cubes 方式的重新拟合,并在此基础上建立各个局部的自适应八叉树;最后,基于各个局部自适应的八叉树,由粗至细渐进地遍历全局自适应八叉树的各个节点,并利用对内存模型能高效渐进压缩编码的先进方法进行编码压缩.实验表明,该方法对外存模型的压缩比达到了与处理内存模型相似的压缩比,高于目前的外存模型压缩方法,是第一个能渐进压缩外存模型的方法.

关键词 外存模型;渐进压缩;自适应八叉树;算术编码;Marching Cubes

中图法分类号 TP391

Progressive Out-of-Core Compression Based on Reconstruction with Marching Cubes

LIU Ying[1)] CAI Kang-Ying[1)] WANG Wen-Cheng[1)] WU En-Hua[1),2)]

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

[2)](*Department of Computer and Information Science, Faculty of Science and Technology, University of Macao, Macao*)

Abstract Out-of-Core models are the massive models that cannot be loaded into the memory as a whole. For improving the efficiency of storing, transmitting and rendering such models, it is very important to progressively compress the models. However, to our knowledge, all the existing out-of-core compression algorithms are single rate, which cannot perform progressive compression. With regard to this, this paper proposes a method that can progressively compress out-of-core models in high efficiency and transmit and render the models in multi-resolutions. At first, the method uniformly divides the bounding box of the out-of-core model into sub-boxes for the local model in every sub-box to be able to process in core, and manages the sub-boxes hierarchically in an octree. Afterwards, the local model in every sub-box is reconstructed with Marching Cubes and an adaptable sub-octree is constructed for the reconstructed local model. Finally, based on the sub-octrees, the nodes of the octree of the whole model can be traversed progressively from coarse to fine and compressed with an advanced compression method for handling in-core models. Experimental results show that the new method can compress the out-of-core models in similar compression ratios as the advanced method to compress the in-core models, it is also superior to the existing methods for compressing out-of-core models and is the first method for performing progressive compression on out-of-core models.

收稿日期:2004-06-09;修改稿收到日期:2004-09-21.本课题得到国家自然科学基金(60373051,60173022)、国家"九七三"重点基础研究发展规划项目基金(2002CB312102)和澳门大学研究基金资助.**刘 迎**,女,1970 年生,博士研究生,研究方向为大规模数据可视化.E-mail: liuyingbj@yahoo.com. **蔡康颖**,女,1977 年生,博士研究生,研究方向为大规模数据压缩. **王文成**,男,1967 年生,博士,研究员,博士生导师,研究方向为科学计算可视化. **吴恩华**,男,1947 年生,博士,研究员,博士生导师,研究方向为真实感图形生成和虚拟现实.

Keywords out-of-core models; progressive compression; adaptive octrees; arithmetic encoding; Marching Cubes

1 引 言

随着计算机处理能力的提高和数据采集能力的增强,大规模数据模型在实践中越来越多,也越来越重要.目前,由几百万面片组成的大规模数据集已经非常普遍[1,2].然而,尽管计算机硬件技术发展非常快,其发展速度还是远远不能满足处理大规模数据的需求.所以,大规模数据在存储、传输、渐进显示等方面的效率还是较差.为此,要对大规模数据模型进行高效的简化和压缩,这是目前国际上计算机图形学的一个热点研究问题.

大规模数据的特点是无法一次把所有数据全部装入内存来处理,这使得以往的内存压缩算法不再适用,必须研究针对外存模型的压缩算法.就我们所知,目前已有的外存模型压缩算法都是单一层次的,尚无法做到渐进压缩.而对于大规模模型,渐进的传输和显示(基于渐进的压缩)有助于读者能尽快地理解模型的全局以提高工作效率.为此,本文提出一种渐进压缩外存模型的算法.

新方法避免了已有的分块处理方法要对各个块进行拼接的复杂工作,是对模型进行整体的编码压缩.它的压缩效率达到了目前内存模型的最高压缩效率,高于目前各种外存模型的压缩效率,且它的压缩是由粗到细渐进进行的.

在压缩过程中,新算法只需顺序遍历外存模型一次,再逐个遍历所有分块数据(能够完全装入内存)各一次.因此,它的运算速度也是很快的.

本文第 2 节介绍与本文算法相关的工作;第 3 节给出本文渐进压缩编码算法;第 4 节给出试验结果;第 5 节为结束语.

2 相关工作

已往的模型压缩算法往往只能处理中等规模的模型,不能直接用来处理外存模型.当前针对外存模型的压缩算法还不是很多,可以分为两大类:分块压缩和流式压缩.分块压缩算法[3]将模型分为一个个的小块,每一个小块都足够小,可以被调入内存用内存压缩算法进行压缩,同时还要记录一些额外信息以便在解压缩的时候把各个小块的压缩结果正确地拼接起来.分块压缩算法从概念上讲非常简单直接,但是由于分块和拼接带来的额外开销使得其效率较低,并且相对于整体压缩的方法而言其压缩比比较低,因为它难以利用模型的一些整体特征进行压缩.流式压缩[4]则是首先重新整理原模型成为一种依照原模型拓扑的流式结构,可以使用户透明地完成随机任意的查询.然后顺序地读取流式结构进行整体的压缩.由于流式结构克服了目前外存模型的存储杂乱无序的情况,能较好地利用相邻三角形在空间和拓扑上的关联性.这种方法的解压缩速度较快,压缩比也较高.

Marching Cubes 算法是目前三维数据场等值面生成中最常用的方法[5,6].该方法实际上是一种分而治之的方法,即把等值面的抽取分布于每个体素中分别进行.该方法基于如下假设:即等值面穿过体素单元的方式是非常有限的,因此可根据每个体素顶点值是大于或小于等值面的值的二元状态列举出所有可能出现的拓扑情况;于是,对每种情况建立其等值面的连接模式并存储在索引表中,就可通过索引快速求得该体素内的等值面多边形生成模式.

目前内存模型压缩效果最好的算法是 2003 年 Siggraph'2003 大会 Lee 等人提出的一种渐进压缩方法[7].该方法对三维均匀数据场中由 Maching Cubes 方法生成的等值面采用八叉树进行层次化的组织,其中最精细体素构成完全八叉树最底层节点.然后,以广度优先遍历的方式处理八叉树的各个节点,同时输出反映模型拓扑特征的两个数据流:"标志流"和"叶子流".由于体素的顶点的值只能是小于或不小于等值面的值这两种情况,这些顶点可标志为 0(表示小于)或 1(表示不小于)."标志流"记录的是按照一种固定顺序访问一个节点对应的 8 个体素顶点的 1 或 0 的标志;如果一个节点对应的 8 个体素顶点都为 1 或都为 0,则在"叶子流"中记录它是不是一个叶子节点.当遍历至最精细层次,就完成了模型的拓扑信息的编码,此时,就对"标志流"和"叶子流"进行算术编码以压缩.该方法只在最精细层次进行几何压缩,即对模型的顶点进行几何位置的编码和压缩.此时,它在每个体素内部计算出一个对偶点,并根据这些对偶点的分布情况建立一个优化的坐标系,使得在该坐标系下每个对偶点的坐标值比较小,然后就对这些坐标值进行编码压缩.由于该方法利用了 Marching Cubes 对拓扑关系的隐含处理,它的拓扑压缩效率非常高,而由于几何压缩都是在各个最精细的体素中进行的,各个点的坐标值都较小,使得它的几何压缩效率也非常好.但该方法只能

处理内存模型,因为它遍历八叉树时要求所有节点均可访问.

3　外存模型的渐进压缩算法

如同文献[8]所讨论的,外存模型往往是过采样生成的,对它进行比较规则的重采样不会影响外存模型的质量,但却可以很好地提高其存储、传输、显示、压缩等操作的效率.为此,我们将外存模型进行基于 Marching Cubes 的重组.这样,在保证重组的模型与原模型相差不大的情况,能够方便地利用文献[7]高效压缩内存模型的方法对外存模型进行渐进的压缩.

由于文献[7]的方法只能处理内存模型,我们将外存模型进行分块,即:将外存模型的包围盒均匀分成许多块,使得每块中的局部模型都能进行内存方式的处理.在每个块中,对模型进行基于 Marching Cubes 的重组,并建立局部的八叉树以管理 Marching Cubes 重组时的网格格点.在各个块的局部八叉树基础上,能获得模型全局的八叉树.这样,就可以仿照文献[7]的方法对全局的八叉树节点进行广度优先的遍历,并输出反映模型拓扑特征的"标志流"和"叶子流".当遍历至八叉树最精细的层次时,就可以对各个体素中的对偶点进行几何编码.由于对偶点的编码都是在体素中局部进行的,因此,几何编码完全可以在各个块中分别完成,最后顺序输出进行算术压缩.

这种基于 Marching Cubes 重组的分块操作,与已有的分块压缩方法不同.它只是利用各个块的局部编码来获得全局的编码,实际上是进行外存模型的整体压缩,不必处理边界缝合的问题.而已有的外存模型的分块处理方法[1,9,10]中分块之间要采用特殊的处理方法以保证拼接的连续性,并且各个块的压缩往往难以高效利用全局特征提高压缩效率.

本文算法的主要步骤如下:

1.将外存模型的包围盒均匀剖分成许多块,使得各个块中的局部模型都能完全装入内存处理,并为这些块建立八叉树的层次化结构,形成块八叉树,即全局八叉树最靠近根结点的几层;

2.对每个块中的局部模型进行基于均匀数据场的 Marching Cubes 的重组,并对各个网格点进行八叉树的组织,形成局部八叉树;

3.基于局部八叉树和块八叉树,可以整合成外存模型的基于 Marching Cubes 重组的全局八叉树;

4.仿照文献[7]的方法,对全局八叉树的节点进行广度优先的遍历,并输出"标志流"和"叶子流".此时,"标志流"和"叶子流"的信息都是从局部八叉树中获知,并不需要实际生成全局八叉树;

5.当对全局八叉树的遍历至最精细层次时,就完成了"标志流"和"叶子流"的形成,并将它们以文献[7]中的方法进行算术编码的压缩;

6.在全局八叉树的最精细层次,对各个体素中的对偶点进行几何压缩.几何压缩完全在各个块中以文献[7]中相应的方法进行处理.

由于本文算法中的几何压缩是完全按照文献[7]中方法进行的,本文下面将不讨论几何压缩,而着重讨论拓扑压缩,特别是如何由局部八叉树和块八叉树得到全局的八叉树.

3.1　分块计算

由于要对模型包围盒的分块进行八叉树的组织,我们将采用包围盒均匀分块的方式,既满足建立完全八叉树的要求,又能使各个块的局部模型能完全在内存中处理.

设块八叉树有 n 层且树的根节点为第 1 层,则此八叉树的最底层有 $8^{(n-1)}$ 个节点.于是,包围盒要剖分为 $8^{(n-1)}$ 个块,且沿 3 个坐标轴方向上都是 $2^{(n-1)}$ 块.

通过计算第 n 层的块的大小,可知为一个块建立局部八叉树所需的空间是否在内存中能得到满足.如果不能满足,就继续剖分直至能得到满足.此时,记录包围盒的分块的层数,并为这些分块建立块八叉树.

然后,读取外存模型一遍,并根据各个三角形在空间的位置将它们分类到各自所属的块中,各个块分别用一个文件记录它有哪些三角形.如果一个三角形与多个块的空间均有重叠,则它在这几个块的文件中都进行记录.

3.2　块内重组

对于每个块的空间,将它进行均匀的网格剖分,如同 3.1 节所讨论的,网格剖分的个数也必须满足建立完全八叉树的条件.

然后,对于每个块,将它所记录的三角形调出来,与它的均匀网格剖分的网格线进行求交计算.根据网格线与模型的面片相交的情况,可知块内网格格点是否在模型的内部或外部,并对内部的网格点标记为 1,外部的网格点标记为 0.这种相交的计算,请见文献[11].

为正确处理外存模型的均匀网格剖分中的网格格点的标记,我们将从最外层的块开始处理,因为最外层块的网格线上的格点内/外性质容易判定,然后逐步地处理相邻的块,就能获得网格点的正确标记.如图 1 中 2 维例子所示,粗黑的线条表示包围盒分成的块,折线表示一个外存模型,虚线表示均匀网格

线.如果先求块 2 中网格点 D 和 E 的内/外性质时,则难以处理;但是,先处理块 1 中的网格点,由于网格线总是从模型外进入模型内,可知 A 在外,B,C,D 在内,这样就知道这条网格线在进入块 2 时是在模型内,于是,在处理完块 1 后再处理块 2 就易知 E 在内.

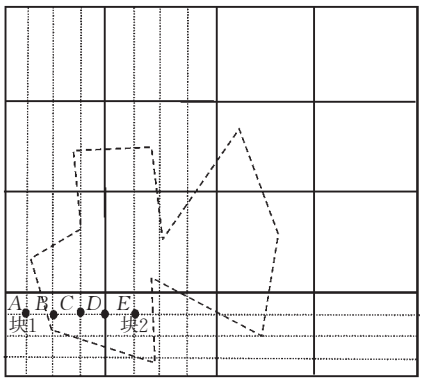


图 1 外存模型的均匀网格剖分及网格格点的内/外标记

在每个块中得到了各个网格点的内/外标记后,就完成了基于 Marching Cubes 的重组.因为,根据 Marching Cubes 方法和网格线上与模型的交点,就能生成与原模型逼近的 3 维模型.显然,网格剖分越细致,则重组模型就越接近原模型.此时的网格线与模型的交点,将用作几何压缩.

然后,就对每个块中的网格剖分形成局部八叉树.

3.3 遍历全局八叉树方法

在得到了各个块的局部八叉树后,就能与块八叉树相结合,完成全局八叉树的遍历并输出全局八叉树的"标志流"和"叶子流".显然,各个块的局部八叉树的根节点就是块八叉树的对应叶节点.

根据文献[7]的方法从根节点开始遍历全局八叉树时,全局八叉树中深度大于块八叉树高度的层次中的节点是位于不同的块中,因此为了生成全局八叉树的这些层次的"标志流"和"叶子流"必须要遍历所有局部八叉树的相应层次.为了保证线性地而不是随机地访问存放在外存上的数据(即保存在外存上的各个局部八叉树),我们按照块八叉树叶子结点的字典序广度优先遍历顺序访问存储各个局部八叉树的文件.由于全局八叉树的遍历顺序按照字典序的广度优先顺序,一定是所有局部八叉树中同一深度的结点都被处理完以后,下一个深度的节点才会被处理,即所有局部八叉树中同一深度的节点在全局八叉树中也位于同一层.

基于上面的讨论,遍历全局八叉树输出"标志流"和"叶子流"的步骤按以下方式进行:

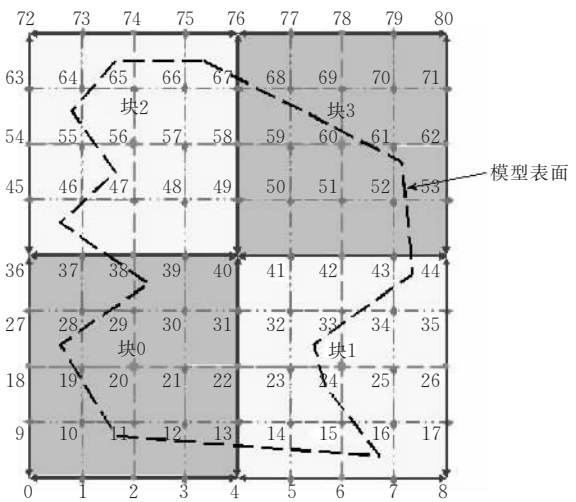
1.按照字典序的广度优先顺序遍历块八叉树,形成全局八叉树最上面几层(等同于块八叉树高度)的"标志流"和"叶子流",由此,各个局部八叉树的根结点的数据已经被包括进去了;

2.为各个块建立局部八叉树,并形成各个局部八叉树的"标志流"和"叶子流"(即每个块的"标志流"和"叶子流").在此,也是按照字典序的广度优先顺序进行遍历;

3.根据按照字典序的广度优先顺序得到块八叉树叶子节点(也是各个分块的根结点)的遍历顺序,该顺序就是处理所有局部八叉树的顺序;

4.按照上一步得到的处理所有局部八叉树的顺序从各个局部八叉树的第二层开始由顶至底地逐层取出各个局部八叉树相应层次的"标志流"和"叶子流"就形成全局八叉树相应层次"标志流"和"叶子流".局部八叉树的取出的"标志流"和"叶子流"在生成时已经剔除了冗余的顶点,因此只需简单地顺序追加数据而不必考虑剔除重复数据的问题.

图 2 为一个 2 维示意图,以解释如何结合块八叉树和块内的局部八叉树形成全局八叉树的遍历(当然,二维图中的树结构都是四叉树).



说明:
全局八叉树高度:4; 分块数量:4; 块八叉树高度:2; 局部八叉树高度:3.
块八叉树的"标志流";
——第一层:0,8,72,80;
——第二层:4,36,40,44,76.
局部八叉树的"标志流":
——第一层:局部八叉树第一层的节点同时是块八叉树的第二层节点(叶节点),该层"标志流"由遍历"基八叉树"第二层得到.
——第二层:(块 0)2,18,20,22,38;(块 1)6,24,26,42;(块 2)54,56,58,74;(块 3)60,62,78.
——第三层:(块 0)1,9,10,11,19,3,12,13,21,27,28,29,37,30,31,39;(块 1)5,14,15,23,7,16,17,25,32,33,41,34,35,43;(块 2)45,46,47,55,48,49,57,63,64,65,73,66,67,75;(块 3)50,51,59,52,53,61,68,69,71,70,71,79;
全局八叉树"标志流"("+"代表顺序追加);
= 块八叉树的"标志流"+ 块 0 的第二层"标志流"+ 块 1 的第二层"标志流"+ 块 2 的第二层"标志流"+ 块 3 的第二层"标志流"+ 块 0 的第三层"标志流"+ 块 1 的第三层"标志流"+ 块 2 的第三层"标志流"+ 块 3 的第三层"标志流"
=0,8,72,80,4,36,40,44,76,2,18,20,22,38,6,24,26,42,54,56,58,74,60,62,78,1,9,10,11,19,3,12,13,21,27,28,29,37,30,31,39,5,14,15,23,7,16,17,25,32,33,41,34,35,43,45,46,47,55,48,49,57,63,64,65,73,66,67,75,50,51,59,52,53,61,68,69,71,70,71,79.

图 2 遍历全局八叉树输出全局"标志流"和"叶子流"的完全四叉树示意图(黑虚线表示全局数据场分成的 4 块;灰虚线表示各块的第二层划分界线;灰点划线表示各块的第三层划分界线)

4　实验结果

我们用 Visual C++ 6.0 实现了本文的新方法,并在一台微机上进行了实验.该机配置了一个 PentuimⅢ 866MHz 的 CPU、256M 内存、20G 的硬盘.

首先,我们检验本文方法的有效性.为此,我们将一些模型进行基于 Marching Cubes 重组后形成一个内存模型,再用本文的新方法和文献[7]的方法分别对它进行处理.这里,模型 Dragon8 是将模型 Dragon 重复 8 次得到的,模型 Dragon88 是将模型 Dragon 重复 8×8=64 次得到的.

图 3 是对于 Dragon 模型的一些处理结果图,a～h为 Dragon 的 8 个分块图形;i,j 为它们重组得到的图形(都是全局八叉树的第八层时的情况).这里,i,j 的图形与文献[7]中的图形完全一样.表 1 和图 3 实验结果表明,新方法能进行正确的重组和压缩.

表 1　本文新算法与文献[7]方法处理内存模型的压缩效率比较

模型名称	三角面片(Byte)	分块数量	块中三维均匀模型(Byte)	八叉树高度[1]	三维均匀数据场采样点个数	"标志流"+"叶子流"(Byte)	顶点个数	新算法的压缩比(bit/vertex)	文献[7]的压缩结果(bit/vertex)
Dragon	30,586K	8	8,386K	8	129×129×129	7076+354	91920	0.646649	0.641459
Buddha	38,168K	8	8,386K	8	129×129×129	7972+581	115874	0.590503	0.597152
Lucy	986,335K	64	8,386K	8	129×129×129	5824+567	71632	0.713759	0.707200
Dragon88	1,957,339K	512	8,386K	8	129×129×129	20796+2815	342272	0.551865	0.569374

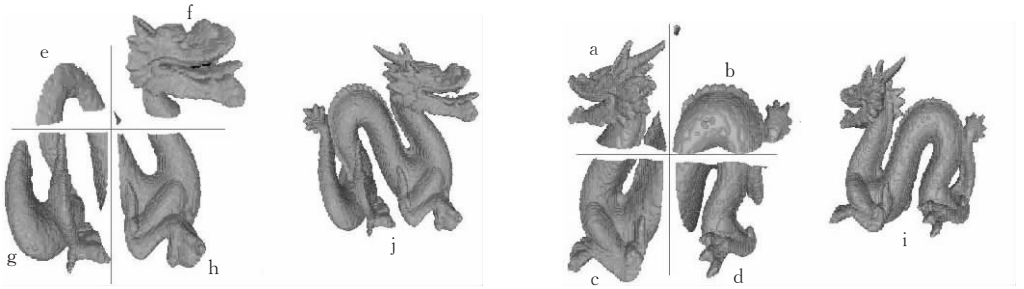


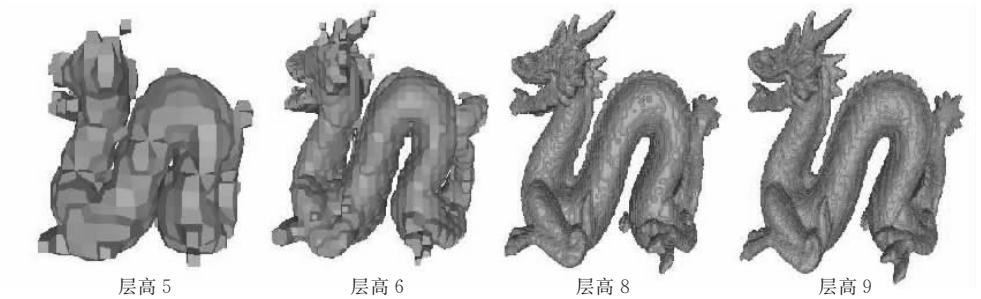
图 3　Dragon 模型的处理结果图,a～h 为 Dragon 的 8 个分块图形;i,j 为它们重组得到的图形(都是全局八叉树的第八层时的情况)

压缩效率的情况,在表 1 中列出.从中可知,本文提出的新方法与文献[7]的方法有非常近似的压缩比.这里,我们只检测了拓扑压缩效率,因为本文方法是采用文献[7]的几何压缩方法进行几何压缩,且几何压缩都是在各个体素内部局部进行的,可以预知它们有相同的几何压缩效率.

在验证了新算法的有效性后,我们将这些模型基于 Marching Cubes 重组生成外存模型,然后再用本文提出的新方法进行压缩.由于建立八叉树的高度对模型的逼近误差是有影响的,我们检测了多个八叉树高度下新方法的压缩效率及重组模型与原模型的误差.实验数据在表 2 中列出,相关的一些模型解压缩渐进显示结果在图 4 中显示.

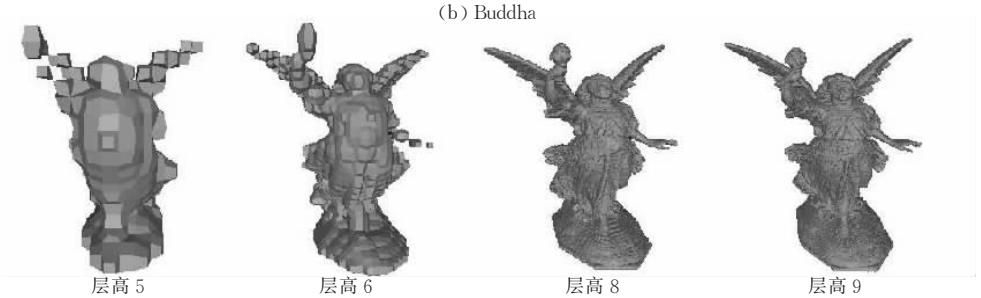
表 2　本文提出的新算法压缩外存模型的效率

模型名称	三角面片数据大小(KByte)	原模型顶点数目	分块个数	三维均匀模型大小(KByte)[2]	全局八叉树高度	采样点个数	标志流+叶子流(Byte)	重采样后顶点个数	重组模型与原模型的误差	压缩时间(h:m:s)	拓扑压缩结果(bit/vertex)
Dragon	30,586	437,645	8	66,308	9	257^3	24673+682	369,404	3.265E-7	00:00:48	0.549101
			64	527,367	10	513^3	91902+1216	1,480,840	2.756E-7	00:05:46	0.503055
Buddha	38,168	543,652	8	66,308	9	257^3	27482+1089	467,934	5.510E-7	00:01:13	0.488462
			64	527,367	10	513^3	100544+1908	1,877,666	4.766E-7	00:08:25	0.436508
Lucy	986,335	14,027,872	8	66,308	9	257^3	19851+1172	292,478	4.560E-3	00:06:17	0.575031
			64	527,367	10	513^3	71791+2280	1,181,335	1.430E-4	00:14:58	0.501609
			512	4,218,936	11	1024^3	266027+4325	4,735,757	3.611E-5	00:39:51	0.456699
Dragon8	986,335	3,501,160	8	66,308	9	257^3	50210+2436	735,360	6.464E-7	00:04:03	0.572737
			64	527,367	10	513^3	188696+4991	2,955,232	1.633E-7	00:11:16	0.524323
			512	4,218,936	11	1024^3	726505+9253	11,846,720	1.378E-7	00:37:17	0.496852
Dragon88	1,957,339	28,009,280	8	66,308	9	257^3	96506+6978	1,444,608	1.027E-6	00:07:52	0.573077
			64	527,367	10	513^3	384944+18475	5,882,864	3.232E-7	00:24:15	0.548602

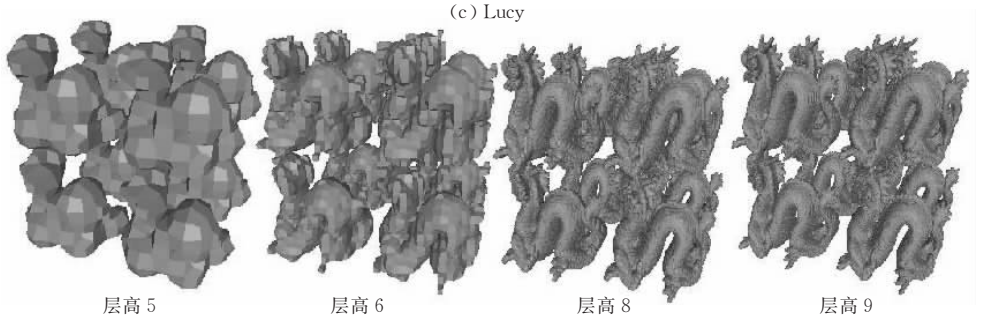


(a) Dragon

(b) Buddha



(c) Lucy



(d) Dragong

图 4 本文算法得到的大规模模型渐进显示图形

从这些实验结果可知,在压缩外存模型时,新方法的压缩比依然达到了文献[7]方法压缩内存模型的效率,甚至还略有增强. 而就我们所知,目前最好的单一层次地压缩外存模型的方法[4]在处理 Buddha 和 Lucy 模型时,其拓扑压缩效率分别是 2.43bit/vertex 和 1.88bit/vertex,而几何压缩效率根据对几何坐标量化精度的不同在 10bit/vertex 到 45bit/vertex 之间. 从表 2 可知,在重组模型非常逼近原模型时,它们的顶点数目也基本差不多,此时,新方法的拓扑压缩效率达到了 0.4～0.6bit/vertex,远好于流式压缩方法的拓扑压缩效率;而新方法的几何压缩效率也将与文献[7]方法的几何压缩效率类似,在 4～6bit/vertex,也要好于流式压缩方法. 并且新方法是渐进压缩的.

5 结 论

为提高外存模型的处理效率,本文提出了一种渐进压缩的方法. 它将外存模型的包围盒空间均匀剖分成许多块,使得每个块中的局部模型都能完全装入内存进行处理,并为这些块形成八叉树的层次化组织;然后,对各个块中的局部模型进行基于 Marching Cubes 的均匀网格重组,使得重组的模型与原模型很

相近,并建立局部八叉树进行组织;最后,结合管理块八叉树和各个块的局部八叉树,就能形成外存模型的全局八叉树,并便于利用高效的内存模型渐进压缩方法对外存模型进行压缩.实验表明,新方法对外存模型的压缩效率达到了目前最好的内存模型压缩效率,高于已有的外存模型压缩方法(它们只能进行单一层次的压缩),且是第一个能对外存模型进行渐进压缩的方法.

外存模型处理的主要困难之一是它不能完全装入内存进行处理.本文的方法将模型的包围盒空间剖分成许多块之后,使各个块中的局部模型都能完全装入内存进行处理,由此,形成了 2 层八叉树结构,实现了对外存模型的压缩.这种方式能很好地处理本文实验的各个外存模型.但是,外存模型的规模是没有限制的,这可能会使这种 2 层八叉树方式不能运用.对此,只要将这种分层的思想推广,类似地形成多层八叉树方式的组织,就能很好地处理.

致　谢　Haeyoung Lee 教授提供了文献[7]的压缩方法的可执行代码,在此表示感谢!

参　考　文　献

1 Bernardini F., Mittleman J., Rushmeier H.. Case study: Scanning Michelangelo's florentine pietà. In: SIGGRAPH'99 Course Notes, Course 8, August 1999. http://www.research.ibm.com/pieta

2 Levoy M., Pulli K., Curless B., Rusinkiewicz S., Koller D. *et al*.. The digital Michelangelo project: 3D scanning of large statues. In: Proceedings of SIGGRAPH'00, New Orleans, Louisiana, USA, 2000, 131～144

3 Ho J., Lee K., Kriegman D.. Compressing large polygonal models. In: Proceedings of Visualization'01, San Diego, California, USA, 2001, 357～362

4 Isenburg M., Gumhold S.. Out-of-Core compression for gigantic polygon meshes. In: Proceedings of SIGGRAPH'03, San Diego, California, USA, 2003, 935～942

5 Tang R. X., Wang J. Y., Peng Q. S., Wang G. Z.. Computer Graphics Tutorial (revision). Beijing: Science Press, 2000 (in Chinese)
(唐荣锡,汪嘉业,彭群生,汪国昭. 计算机图形学教程(修订版). 北京: 科学出版社, 2000)

6 Lorensen W., Cline H.. Marching cubes: A high resolution 3D surface construction algorithm. In: Proceedings of SIGGRAPH'87, Anaheim, California, USA, 1987, 163～169

7 Lee H. L., Desbrun M., Schröder P.. Progressive encoding of complex isosurfaces. In: Proceedings of SIGGRAPH'03, San Diego, California, USA, 2003, 471～476

8 Attene M., Falcidieno B., Spagnuolo M., Rossignac J.. Swing wrapper: Retiling triangle meshes for better edgeBreaker compression. ACM Transactions on Graphics, 2003, 22(4): 982～996

9 El-Sana J., Chiang Y.-J.. External memory view-dependent simplification. In: Proceedings of Eurographics'00, Swansea, UK, 2000, 139～150

10 Cignoni P., Montani C., Rocchini C., Scopigno R.. External memory management and simplification of huge meshes. IEEE Transactions on Visualization and Computer Graphics, 2003, 9(4): 525～537

11 Pfister H., Zwicker M., Baar J. V., Gross M.. Surfels: Surface elements as rendering primitives. In: Proceedings of SIGGRAPH'00, New Orleans, Louisiana, USA, 2000, 335～342

LIU Ying, born in 1970, Ph. D. candidate. Her research interests focus on visualization of massive model.

CAI Kang-Ying, born in 1977, Ph. D. candidate. Her research interests focus on compression of massive model.

WANG Wen-Cheng, born in 1967, Ph. D., professor. His research interests focus on visualization in scientific computing;

WU En-Hua, born in 1947, Ph. D., professor. His research interests include realistic image synthesis and virtual reality.

Background

This paper is supported by the National Natural Science Foundation of China under grant No. 60373051, 60173022, the National Basic Research Program of China (973 Program) under grant No. 2002CB312102, and Research grant of University of Macau.