

文章编号:1001-9081(2010)01-0047-03

## 基于深度八叉树的三维数据场 LOD 可视化

马晓晨<sup>1,2</sup>, 孔小利<sup>2</sup>

(1. 天津大学 机械工程学院, 天津 300072; 2. 承德石油高等专科学校 计算机与信息工程系, 河北 承德 067000)

(maxiaochen@sina.com)

**摘要:**提出了广度八叉树、深度八叉树概念,分析了它们逻辑结构和存储结构,探讨了这两种数据结构在三维数据场可视化中的应用,把深度八叉树应用于三维数据场 LOD 体绘制算法中。算法在某三维震波数据场进行了体绘制实验,并与传统方法进行了比较分析。结果表明,该方法通过逐层简化细节来减少场景的复杂性,提高了渲染效率,将全局和局部体绘制相结合,既提高了绘制速度,又实现了精细观察。

**关键词:**广度八叉树;深度八叉树;细节层次;体绘制;三维数据场

**中图分类号:** TP391 **文献标志码:** A

## 3D data set LOD visualization based on depth-oct-tree

MA Xiao-chen<sup>1,2</sup>, KONG Xiao-li<sup>2</sup>

(1. School of Mechanical Engineering, Tianjin University, Tianjin 300072, China;

2. Department of Computer and Information Engineering, Chengde Petroleum College, Chengde Hebei 067000, China)

**Abstract:** This paper put forward the concepts of breadth-oct-tree and depth-oct-tree. It analyzed their logic structures and storage structures, which were applied in 3D data set visualization. Depth-oct-tree was used in LOD volume rendering algorithm which was tested on a seismic wave data set. Compared with the traditional method, the new algorithm was much faster. On the contrary, the traditional method cannot be applied in PC due to the large amount of time and space required. The results indicate that the new method can improve the performance of volume rendering. Especially, combining the whole and part volume rendering method, the speed was higher and more elaborated observation was achieved.

**Key words:** breadth-oct-tree; depth-oct-tree; Level Of Detail (LOD); volume rendering; 3D data set

### 0 引言

随着数据采集技术的快速发展,震波 CT、工业 CT 等三维数据越来越精细,其体数据也越来越大,有的数据体达到几十吉字节、十几吉字节,甚至几十吉字节,给大规模三维数据体快速可视化技术带来了难题,尤其是基于微机的大规模数据场快速可视化,在传统方式下几乎难以实现,因此对数据进行预处理是实现快速可视化的前提。

近几年,人们针对大规模三维数据场体绘制技术研究了很多方法,文献[1]提出的动态纹理载入的方法和文献[2]提出的半自适应分块的方法更多地依赖纹理体绘制进行快速体绘制,由于受硬件纹理空间大小的限制,基于纹理的体绘制方法无法有效地针对超大规模的体数据,将数据块多次从系统缓存中载入到纹理内存直接影响了大规模体数据绘制的速度。文献[3]提出的基于矢量化压缩的方法虽然提高了大规模数据的体绘制速度,但针对超大规模数据场,如 10 GB 以上的数据场,此方法也很难实现快速体绘制。大规模三维数据场的体绘制要求必须对空间数据进行分割等预处理,这主要是由于一方面尽管目前计算机的性能有较大的提高,但对于大规模数据场的可视化而言,内存容量、计算和绘制性能仍然是非常有限的,不可能将海量的空间数据一次性从外存读入进行处理,而必须是分块调度;另一方面是根据人眼在观察事物时的规律,对较远处的场景人眼能够获得的信息相对较少,

而随着距离的拉近,对细节的观察越来越详细,因此对远近不同的场景可以采用不同的“粒度”进行描述,这就是细节层次(LOD)方法的基本原理。LOD 技术最早是在 1976 年由 Clark 提出,1992 年后,国内外学者相继提出了许多 LOD 模型的生成算法<sup>[4]</sup>。在三维数据处理中八叉树是传统的且效果较好的数据结构,因此在超大规模三维体数据处理过程中,应用八叉树对数据进行分解,按需要分级导入,可以实现对大规模三维数据局部进行快速可视化<sup>[5]</sup>。但是当需要对全局或者对超过计算机容量的较大局部进行概览和分析时,此方法无法实现,因此需要应用 LOD 技术。本文将 LOD 技术和八叉树相结合,提出了深度八叉树的概念,应用于大规模三维数据体快速体绘制中,为了与深度八叉树相区别,本文将传统的八叉树方法称为广度八叉树。

### 1 广度八叉树与深度八叉树

#### 1.1 广度八叉树

广度八叉树的逻辑结构 令  $U$  是一个可以存储某长方体的空间且大小一定,其值为  $s$ ,  $V$  为一个要表示三维数据体,其长、宽、高分别为  $m, n, k$ , 它的八叉树逻辑结构可以用如下的递归方法来定义:

设函数  $f$  表示一个体数据所需的空间大小,让八叉树的每个节点与  $V$  的一个子体对应,如果  $f(V) \leq s$ , 那么  $V$  的八叉树仅有树根,如果  $f(V) > s$ , 则将  $V$  等分为八个子体  $V_i (i = 0, 1, 2,$

收稿日期:2009-06-17;修回日期:2009-08-17。

基金项目:河北省自然科学基金资助项目(602405);中国石油天然气集团公司石油科技中青年创新基金资助项目(05E7048)。

作者简介:马晓晨(1968-),男,河北保定人,教授,博士研究生,主要研究方向:三维可视化、智能诊断;孔小利(1962-),男,天津蓟县人,教授,博士,主要研究方向:虚拟现实。

3,4,5,6,7),每个子体与树根的一个子节点相对应,如图1。

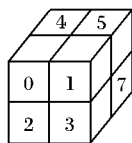


图1 子体的8等分

只要 $f(V_i) > s$ ,就要被8等分,从而对应的节点也就有了8个子节点,这样地递归判断、分割一直要进行到节点所对应的子体 $f(V_i) \leq s$ 。

**广度八叉树的存储结构** 按照此八叉树逻辑结构,可以将体数据分割成若干小体,每一次分割,都是将每一个子体分成八个部分。最终得到一棵满八叉树,分割完成后所得到的八叉树的每一个叶子节点对应一个磁盘中的文件,图2是一个深度为3的广度八叉树。

内存和磁盘是海量数据处理需要重点考虑的两个层次<sup>[6]</sup>,八叉树存放于内存,而节点对应的文件存放于外存。用广度八叉树来实现大规模数据场的存储,可以看成是四叉树方法在三维空间的推广,也可以认为是用三维体素阵列表示形体方法的一种改进。

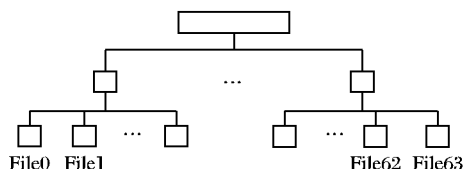


图2 广度八叉树

## 1.2 深度八叉树

深度八叉树与广度八叉树最主要的区别是:广度八叉树只有叶子节点对应一个磁盘文件,而深度八叉树每个节点都对应一个磁盘文件。

深度八叉树的逻辑结构与广度八叉树相同。

**深度八叉树的存储结构** 按照此八叉树逻辑结构,可以将体数据分割成若干小体,每一次分割,都是将每一个子体分成8个部分,即一个子体被8等分。最终得到一棵满八叉树,分割完成后所得到的八叉树的每一个节点对应一个磁盘中的文件,且文件大小相同。

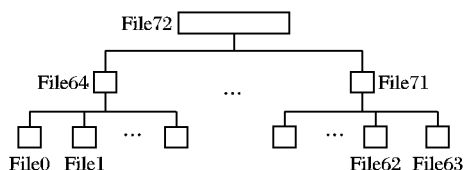


图3 深度八叉树的逻辑结构

从叶子节点开始,每个节点对应的磁盘文件所存储的数据如下:

1) 叶子节点对应的文件与广度八叉树的完全相同。

2) 每个叶子节点的父节点对应的文件都是由其子节点按X, Y, Z三个方向进行2-抽稀(即每间隔一个点取值)后合并生成,每个子节点经过2-抽稀后大小是原来叶子节点文件大小的1/8,8个子节点合并后得到的文件与子节点对应的文件大小相同。

3) 以此类推,得到全部非叶子节点所对应的文件。

图3为一个三层深度八叉树的逻辑和存储结构。

File0, File1到File8经抽稀处理后得到File64;

.....

File56, File57到File63经抽稀处理后得到File71;

File64, File65到File71经抽稀处理后得到File72;

共有 $(8^n - 1)/7$ 个文件,其中 $n$ 为八叉树的深度。

## 2 广度八叉树的应用

当一个大规模三维数据场的数据远大于计算机内存的容量时,计算机已经不能对三维数据体实现较为正常和全面的可视化功能,更不用说快速可视化了。因此,将大规模三维数据场分割存储以及合理调度是可视化工作必要的前提,应用广度八叉树可以较好地解决这个问题。按照本文广度八叉树的定义,首先确定可以存储某长方体的空间 $U$ ,其大小为 $s$ ,这里 $s$ 即为当前计算机实现快速调度的内存块的大小,因此 $s$ 是根据不同计算机的内存大小而不同,根据 $s$ 的大小,可以计算所需广度八叉树的深度,八叉树的深度要满足公式 $s \times 8^{(d-1)} \geq m \times n \times k \times p$ ,其中 $d$ 为八叉树的层数, $p$ 为数据场体素所需要的内存空间大小。令 $L = m \times n \times k \times p/s$ ,则有:

$$d \geq \log(L/8) + 1$$

为了实现每次对子体进行8等分,需要 $m, n, k$ 分别满足被 $2(d-1)$ 整除,如果不能满足,可以对原始三维数据体填充空体元,适当增加 $m, n, k$ 的值,得到 $m', n', k'$ ,使得 $s(d-1) \geq m' \times n' \times k' \times p$ 。当八叉树的深度为 $d$ 时,数据场被分割为 $(8^n - 1)/7$ 个子体, $(8^n - 1)/7$ 个节点,其中包括 $8^{d-1}$ 个叶子节点,即 $8^{d-1}$ 个分割文件。当进行局部体绘制以及和切片可视化和表面建模时,可以应用广度八叉树进行调度,根据需要读入所需数据。

## 3 深度八叉树的应用

大规模三维数据体可视化是实际应用中的一个重要研究问题,实时性要求三维场景能够以很高的帧速率按照用户的交互指令动态显示。但是海量数据和复杂的运算对体绘制速度造成很大影响,为了达到较高真实感的三维可视化,即便是最高端的图形工作站也不能满足实时绘制的需求,必须采用一些算法来改进和加速绘制过程<sup>[7]</sup>。本文应用了深度八叉树实现了大规模三维数据体的LOD体绘制。

### 3.1 数据的表示与组织

三维数据场有结构化和非结构化等不同的数据结构<sup>[8]</sup>,常用的结构化数据结构主要包括正方形网格结构、矩形网格结构、不规则结构等,由于规则排列的正方形网格除了每个网格点处的属性值以外,只需要记录一个起算点的位置坐标和网格间距,因此存储量很小,适合于大规模的使用和管理。由于采用网格数据结构,数据的分割比较容易处理。网格数据结构是规则间隔的网格阵列,可将原始的数据体分割成多个等大的“子体”,考虑到软硬件条件,如果分块过大,可能仍然超出机器的处理能力,如果分块过小,则可能带来过多的数据调度。本文采用了基于深度八叉树的分割方法,LOD处理过程主要是对三维数据体的抽稀,在三维坐标轴的每个方向上应用2-抽稀,总层数 $N$ 的计算公式为:

$$N = \lceil \log_8 \frac{V}{V_0} \rceil$$

其中: $V$ 为三维数据体字节数, $V_0$ 为一次适度载入数据的字节数。

大规模三维数据体可以被不同粒度,不同尺寸的多个子体替代,每个子体都可以独立绘制处理。为了对三维数据体进行由全局到局部、由粗略到详细的观察,支持全局范围内体绘制,需要对子体进行编码,从而建立起子体之间的空间关

系。子体间的空间关系主要有两方面:一是细节层次,是指一个较粗的子体与多个较细子体之间的“父子”关系;二是同一层的位置,表达同一细节层次的子体之间的邻接关系。编码的目的是使子体便于定位,避免复杂的查找过程。数据编码如下:

[LevelNum][XCode][YCode][ZCode]

其中:LevelNum 代表该子体所处的细节层次,XCode、YCode 和 ZCode 分别表示在该层内按照 X、Y 和 Z 方向的编码。通过对编码分别加 1 或减 1,便可以确定同一层上邻接的子体编码。大规模三维数据场的空间数据最终被组织成一个由不同细节层次的多个子体构成的树形结构,见图 3。位于根节点的子体具有最大的数据粒度,可以称其为全局场景,由上至下粒度逐层减小,细节的描述越来越详尽,体现出场景的逐级放大;同一层子体之间数据粒度相同,覆盖整个场景的不同区域,且依次邻接。

八叉树的节点存储结构如下所示。

Node * pParent	Node * pChild[8]	int SN	Type Prop
父节点	子节点	节点序号	属性

### 3.2 实验及结果分析

将上述算法应用于光线投射算法中,在配置为:CPU Intel Core2 6320,内存 2 GB,显卡为 NVIDIA GeForce 8600 的微机上对两个震波数据体(300 × 800 × 2 500)、(1 200 × 1 500 × 1 501)进行了实验,实验结果见表 1。

表 1 绘制速度比较

数据体	不采用 LOD 技术	新算法
300 × 800 × 2 500	12 秒/帧	0.8 秒/帧
1 200 × 1 500 × 1 501	无法加载	2.6 秒/帧

从实验结果可以看出,应用本文提出的新算法可以使大规模三维数据场实现实时体绘制。而在本文前面提到文献中的算法中,与本文相当大小的数据体绘制的速度比本算法慢几倍到十几倍。因为在新算法中,LOD 体绘制根据需求确定应该选用哪一层数据,这样就大大加快了体绘制速度。LOD 技术使大规模数据场的体绘制由原来的加载困难,变成流畅地交互。图 4 是数据体(300 × 800 × 2 500)得到的体绘制结果。由图 4(a)我们看到在全局体绘制时由于丢掉了一些细节比原有图像要模糊一些,但基本满足需要。虽然在全局绘制时细节损失较大,但局部精细体绘制弥补了全局体绘

制的粗糙,绘制结果如图 4(b)所示。实验证明以数据分割和动态调度为主要思想的解决方案在提高大规模数据场的体绘制速度方面作用显著。

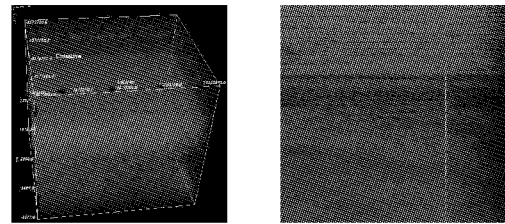


图 4 体绘制结果

## 4 结语

针对大规模三维数据场,提出了广度八叉树、深度八叉树的概念,分析了广度八叉树的应用。探讨了新的三维 LOD 图像渲染技术,并应用于三维数据场体绘制。应用本方法,使得传统方法在微机上不可能实现的大规模三维数据场体绘制得以快速绘制。该技术尽量保持了画面视觉效果,逐层简化细节,大大提高了可视化效率。基于深度八叉树的体绘制方法实现了全局和局部体绘制相结合,较低层次的 LOD 体绘制提高了绘制速度,同时,较高层次的 LOD 体绘制实现了精细观察。

### 参考文献

- [1] 薛健, 吕捷, 戴亚康, 等. 海量医学数据处理框架及快速体绘制算法[J]. 软件学报, 2008, 19(12): 3237 - 3248.
- [2] 郑杰, 姬红兵. 基于动态纹理载入的大规模数据场体绘制[J]. 中国图象图形学报, 2008, 13(2): 317 - 321.
- [3] 吴仲乐, 鲍旭东, 罗立民. 基于矢量量化压缩的大规模体数据直接绘制[J]. 东南大学学报: 自然科学版, 2005, 35(3): 475 - 479.
- [4] 王臻, 胡敏, 李响. 基于四叉树的动态多分辨率 LOD 地形快速简化[J]. 计算机应用, 2007, 27(7): 1641 - 1643.
- [5] DORN J, HRASTNIK P, RAINER A. Web service discovery and composition for virtual enterprises [J]. International Journal of Web Service Research, 2007, 4(1): 23 - 39.
- [6] VITTER J. External memory algorithms and data structures: Dealing with massive data [J]. ACM Computing Surveys, 2001, 33(2): 209 - 271.
- [7] 童欣, 唐泽圣, 许忠信. 基于纹理硬件的大规模体数据快速绘制算法[J]. 清华大学学报: 自然科学版, 2000, 40(1): 72 - 75.
- [8] 唐泽圣. 三维数据场可视化[M]. 北京: 清华大学出版社, 2000.

(上接第 46 页)

图均等化为基础的改进方法应用于集装箱 X 光图像的增强。该方法可以将感兴趣的目标区域增强,同时将不感兴趣的区域进行削弱,从而消减了集装箱构造特点的噪声因素。实验表明本文方法能够较准确、快速地对集装箱 X 光图像进行增强,通过设置参数使图像达到对比度高,信噪比高的效果,不仅突出了集装箱 X 光图像中感兴趣区域的清晰度,同时还抑制图像中由于集装箱车箱竖纹的影响所带来的噪声,使图像更便于海关以及工业等领域对集装箱 X 光图像的处理。

### 参考文献:

- [1] GONZALEZ R C, WOODS R E. Digital Image Processing[M]. 2nd ed. Massachusetts: Publishing House, 2007.
- [2] FAYAD L M, JIN YINPENG, LAINE A F, et al. Chest CT Window settings with multiscale adaptive histogram equalization: Pilot study

[EB/OL]. [2009 - 04 - 22]. [http://hbil.bme.columbia.edu/wp-content/themes/HBIL\\_MJP/publications/164.pdf](http://hbil.bme.columbia.edu/wp-content/themes/HBIL_MJP/publications/164.pdf).

- [3] PALANISWAMI M, NGUYEN T O, OXBROW R. An efficient filter structure for edge detection[EB/OL]. [2009 - 04 - 22]. [http://videoprocessing.ucsd.edu/publications/Year\\_1988/File3.pdf](http://videoprocessing.ucsd.edu/publications/Year_1988/File3.pdf).
- [4] 苏明忠. 医学图像分割算法研究[D]. 长春: 长春理工大学, 2007.
- [5] 夏勇. 图像分割技术研究[D]. 西安: 西北工业大学, 2004.
- [6] 章毓晋. 图像工程(上册): 图像处理和分析[M]. 北京: 清华大学出版社, 1999.
- [7] PIZER S M, AMBURN E P, AUSTIN J D, et al. Adaptive histogram equalization and its variations[J]. Computer Graphics Image Processing, 1987, 39(3): 355 - 368.
- [8] 苏金明, 王永利. Matlab7.0 实用指南[M]. 北京: 电子工业出版社, 2004.