用于建立三维 GIS 的八叉树编码压缩算法

曹 彤[1] 刘 臻[2]

[1](北京联合大学应用文理学院信息科学系，北京 100083)

[2](北京师范大学计算中心，北京 100875)

摘 要 复杂的空间数据结构在三维 GIS 领域中占有突出的地位，它直接关系到 GIS 的功能和效率. 为了有效地进行三维 GIS 大量数据的存储和管理，重点讨论了三维 GIS 栅格数据结构中的八叉树编码压缩技术. 由于 Morton 码值的排序是实现八叉树编码压缩的基础，为此，根据 Morton 码排序的特殊性，提出了采用时间复杂度为 $O(n)$ 的计数排序算法，使排序速度大为提高，在此基础上进行压缩处理，并对算法的时间及空间复杂度进行了分析. 在 PC 机上进行的模拟实验结果表明，在目标复杂度一定的前提下，八叉树存储数据占用空间小(当分割阶次为 9 阶时，八叉树存储量只占栅格存储量的 4.32%)，是一种较为理想的描述复杂海量地理空间数据的压缩结构.

关键词 三维地理信息系统 八叉树 计数排序 压缩算法

中图法分类号：TN919.81 **文献标识码**：A **文章编号**：1006-8961(2002)01-0050-05

Quick Encoding Compression Algorithm of Octree for Modeling Three Dimensional GIS

CAO Tong[1], LIU Zhen[2]

[1](*Information Science Department, Applied Science and Humanities College, Beijing Union University, Beijing* 100083)

[2](*Center for Computer, Beijing Normal University, Beijing* 100875)

Abstract To meet the requirements of the three-dimensional(3D) GIS community, it is necessary to study sophisticated 3D data structures, which exerts very important influence on GIS function and efficiency. An efficient and compact raster data structure in 3D GIS called octree is presented. Arbitrary 3D objects can be represented to any specified resolution in a hierarchical 8-ary tree structure or octree. To save memory, E-octants are not registered. A linear octree data structure that lists only octree codes of F-Octree is used. This paper focuses on discussing the compression technique of linear octree. According to the peculiarity of morton code, Counting sort algorithm occupying linear time is applied to sort code. Sorting time is faster 21 times than quick sort with $n=7$. An effective way of compacting sorted octants having the same attribute is given, and temporal and spatial analysis is described. Experiments on PC demonstrates that the memory storage for octree code using proposed algorithm is considerably low in comparison to others(octree/raster = 4.32% with $n=9$). The study also indicates that octree is an ideal data structure for representing the sophisticated geographic spatial information. Program "3D-Octree" has been written to develop, verify and demonstrate the octree encoding algorithms.

Keywords 3D GIS, Octree, Counting sort, Compression algorithm

0 引 言

从本质上说，与地球特征有关的数据，即通常所说的空间数据是三维连续分布的，而目前的地理信息系统(GIS)还停留在处理地球表面数据的阶段. 由于它们在处理三维问题上的不足，即目前大多是二维数据表示及 2.5 维景观模型立体表示[1]，使得 GIS 在资源勘探、地质模型、城市建设等领域的应用受到很大限制，因而这些部门迫切需要一种具有真三维处理和

收稿日期：2000-10-11；**改回日期**：2001-06-04

分析功能的 GIS 系统,然而由于三维技术比二维、2.5 维复杂得多,故三维 GIS 理论和应用已成为 GIS 研究的学术前沿. 由于 3D GIS 中数据量的增大,因此使得数据结构占有更加突出的地位,其关键问题是空间数据的算法及其空间数据的组织. 随着计算机技术的迅猛发展,计算机 CPU 的运算速度及内存容量均使得各种数据结构运算的实现成为可能,这也为三维地理信息系统的发展提供了硬件基础. 由于八叉树特别适合于描述地学中的复杂地物现象,近些年来被广泛地应用于地学空间分析研究.

1　八叉树编码

用八叉树表示三维目标时(如本文实例——矿体信息的表达),首先采用一个立方体来包围该目标,然后将它分成 8 个大小相同的小立方体,若立方体被矿体充满或为空,则不必再分割;若此立方体部分被填充,则必须将该立方体再细分(X、Y、Z 3 个方向二分切割)为更小的 8 个大小相同的立方体,同时相应级数增加 1. 依次类推,直到分割第 n 级(第 n 次二分切割产生的立方体为 n 级立方体)满足精度要求为止. 一般地,第 1 级分割有 8 个立方体,第 2 级分割最多可有 64 个立方体,…,第 n 级分割最多可有 8^n 个立方体.

八叉树结构可看作是一种变块模型[2],其中属性相同的区域用大块表示,而复杂区域用小块表示(图 1). 为了便于对三维现象进行空间分析,通常将三维空间坐标转换为三维栅格坐标表示,就矿体而言,栅格的大小取决于矿体性质的复杂程度. 若某一栅格坐标为(I,J,K),且$(I,J,K)=0,1,2,\cdots,2^{n-1}$,则其可用二进制表示为

$$\begin{aligned} I &= c_{n-1}2^{n-1}+\cdots+c_l2^l+\cdots+c_02^0\\ J &= d_{n-1}2^{n-1}+\cdots+d_l2^l+\cdots+d_02^0\\ K &= e_{n-1}2^{n-1}+\cdots+e_l2^l+\cdots+e_02^0 \end{aligned} \quad (1)$$

其八叉树编码 Q 为

$$Q = q_{n-1}8^{n-1}+\cdots+q_l8^l+\cdots+q_08^0 \quad (2)$$

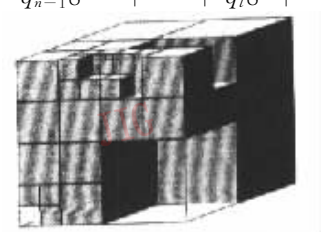

图 1　八叉树变块模型

其中,Q 为八进制码.

$$q_l=e_l2^2+d_l2^1+c_l2^0;$$
$$(e_l,d_l,c_l)\in\{0,1\};l=n-1,n-2,\cdots,0 \quad (3)$$

例如栅格坐标$(I,J,K)=(3,0,2)$,其对应的八叉树编码为$(51)_8$,十进制编码为 41.

2　八叉树的构成

2.1　计算编码

存放八叉树编码常用的地址码是 Morton 码,定义为:$Q=\mathrm{Morton}(n,x,y,z)$,式中 x,y,z 分别表示 X、Y、Z 方向的编码,n 为树的深度,Q 为线性编码.

实现 Morton 编码的最普通的方法是求余数法,这种方法根据栅格坐标,用求余数的方法来求得式(1)中的 I,J,K,再根据式(2)求编码 Q. 由于该方法运行速度慢,特别当 n 较大时,因此后来有学者提出采用比特位运算法[3],即先分别对二进制表示的 K、J、I,从高位到低位依次移入八叉树编码的地址,最后从编码地址中,取出八叉树的编码,这样,其编码速度将有所提高. 本文采用一种更为高效的编码转换算法——查表法. 它的实质是将转换 Morton 码的过程转化成为查一张 Morton 表的过程. 记此表为:

$$\begin{aligned} M(I) &= \sum_{l=0}^{n-1} i_l8^l\\ Q&=M(I)+2\times M(J)+4\times M(K)\\ &=M(I)+M(J)\ll 1+\\ &\quad M(K)\ll 2 \end{aligned} \quad (4)$$

从式(4)可以看出,转换表并不是栅格坐标与 Morton 码值的完全对应,而是相当于一个坐标分量的贡献. 另外,码表的生成也很简单,即只需将 I 的二进制表示$(i_{n-1}\ i_{n-2}\cdots i_0)_2$ 换为八进制数$(i_{n-1}i_{n-2}\cdots i_0)_8$,即可得到需要的码值. 这样由栅格坐标到 Morton 码的转换过程将只是查表,即数组元素引用的过程,这将大大提高转码处理速度.

2.2　编码排序及其时间复杂度分析

将栅格坐标变换成 Morton 编码后,得到的是一组无序的 Morton 编码记录,包括 Morton 代码和属性数据. 为了对这些记录进行压缩,首先应对 Morton 编码进行排序.

若采用基于比较的排序方法,如冒泡排序[4],其时间复杂度为 $O(n^2)$,很明显,当 n 较大时,其速度会很慢. 即使采用通用排序算法中效率最高的快速排序法[5],其时间复杂度的下界是 $O(n\lg n)$,排序依然占

用了不少时间,而计数排序[6]则适用于待排序对象的排序关键字是非负整数的情况. 因为这种排序事先知道关键字的取值范围,且在排序过程中没有比较过程,所以排序时间较短. 具体思路是首先分配一个大小为最大关键值的数组,然后用此数组对排序关键字进行计数,即计数数组中每个元素的值为待排序数组中关键字值出现的次数,最后从大到小,根据计数数组将待排序数组提取到结果数组中,从而完成排序过程,其整个算法的时间复杂度为 $O(n)$. 在本次工作中考虑到八叉树编码排序的实质是要按 Morton 码的大小进行排序,在排序后的数据中,由于 Morton 码是从小到大连续排列的,且 Morton 码本身又是自然数,因此,采用具有线性运行时间 $O(n)$ 的计数排序是最佳策略. 其核心代码如下:

```
for(i=0;i<pointNumber;i++)
    resData[rowData[i].mortonValue].attrib=
rowData[i].attrib;
```

具体做法是:

(1) 将网格化后栅格坐标对应点的属性值,依照一定的次序(按 x,y,z 方向分别变化),读入内存 readData[i]中;

(2) 通过查表(Morton = morTab[x] + (morTab[y]≪1) + (morTab[z]≪2))将栅格坐标转换成 Morton 编码 Q_i,这时得到的是一组无序的 Morton 编码记录;

(3) 为排序后的 Morton 编码记录开辟一块新的内存 writeData[Q_i],用于存放属性值,将与栅格坐标对应的属性值 readData[i]赋予排序后的 Morton 编码记录 writeData[Q_i],即 writeData[Q_i]=readData[i].

在具体的操作中,由于计算机的资源有限,因此,当切割立方体的阶数较高,而内存不够使用时,则会出现频繁的磁盘操作而降低处理效率. 为使读盘次数尽可能减少,拟采用如下对策:假设计算机配置(主要指内存)不需读盘就能支持的分割阶次是 n 阶,在栅格化处理对象时,如果要求的精度大于 n 阶(如需要 $n+1$ 阶),则可先将对象划分为 n 阶大小的 8 个立方体,同时对这 8 个立方体分别进行栅格化,然后按顺序对各块进行 Morton 码转换、排序、合并和生成八叉树. 因为计数排序的实质是以 Q_i 编码作为排序后的属性数组的下标,所以实现了 Morton 编码的自然排序,其时间复杂度为 $O(n)$. 表 1 为快速排序与计数排序算法的时间对比,7 阶时,比快速排序法要节省 21 倍的时间.

表 1 计数排序与快速排序算法所需时间对比

阶次	快速排序多次累计用时(ms)	计数排序多次累计用时(ms)	计算次数	快速排序每次平均用时(ms)	计数排序多次平均用时(ms)	时间比(T_1/T_2)
1	4 500	4 510	6 000	0.75	0.75	1.00
2	4 330	3 680	5 000	0.87	0.74	1.18
3	4 230	1 650	2 000	2.12	0.82	2.56
4	14 550	1 650	1 000	14.55	1.65	8.82
5	30 480	1 650	200	152.40	8.25	18.47
6	40 590	2 140	30	1 353.00	71.33	18.97
7	72 280	3 450	5	14 456.00	690.00	20.95

注:此表中数据获取的实验环境如下:硬件环境:Pentium MMX 233M,32M 内存,硬盘 3.2G;软件环境:Windows98,VC5.0.
阶次表示对处理对象进行二分切割的最大次数.

2.3 压缩算法

排序后的数据经压缩,形成八叉树,具体做法如下:

① 对排序后的 8^n 个记录,按顺序检查是否有连续属性相同的记录,如果编码为 $i\times8$,$i\times8+1$,…,$i\times8+7$,($i=0,1,2,\cdots,8^{n-1}-1$)的 n 级八分体属性相同,则用一个编码为 i 的 $n-1$ 级八分体来代替这 8 个 n 级八分体,即当 8 个连续的八分体在空间分布上同属一个较大的立方体,且它们的属性相同时,可以用一个上级较大的立方体来代替,此时 8 个存储空间就减少到了一个存储空间,如果属性不同,则设定用 FF 作为属性数据,赋给编码为 i 的 $n-1$ 级八分体,这样可得到 8^{n-1} 个 $n-1$ 级八分体. 此时,n 级八分体的数目将大大减少(为原数据的八分之一),从而实现了压缩;

② 对 $n-1$ 级八分体继续进行合并处理,一直到不能再压缩为止;

③ 将属性为 FF 的记录去掉,即得到压缩后的记录. 由栅格存储转换成二阶八叉树存储的压缩结果如表 2 所示.

表 2　二阶压缩过程

X	Y	Z	八进制编码 Q_1	十进制编码 Q_2	属性	压缩结果 八进制编码 Q_1	压缩结果 属性
0	0	0	00	0	1	0	1
0	1	1	06	6	1	6	1
1	1	1	07	7	4	7	4
2	0	0	10	8	1	10	1
3	0	1	15	13	1	15	1
2	1	1	16	14	4	16	4
3	1	1	17	15	1	17	1
0	2	1	24	20	1	24	1
1	2	1	25	21	4	25	1
0	3	1	26	22	1	26	1
3	3	0	33	27	1	33	1
2	2	1	34	28	4	34	4
3	2	1	35	29	1	35	1
0	1	2	42	34	1	42	1
1	1	2	43	35	4	43	4
0	0	3	44	36	1	44	1
3	0	2	51	41	1	51	1
2	1	2	52	42	4	52	4
3	1	2	53	43	1	53	1
0	2	2	60	48	1	60	1
1	2	2	61	49	4	61	4
0	3	2	62	50	1	62	1
1	3	3	67	55	1	67	1
2	2	2	70	56	0	70	0
3	2	2	71	57	0		
2	3	2	71	58	0		
3	3	2	73	59	0		
2	2	3	74	60	0		
3	2	3	75	61	0		
2	3	3	76	62	0		
3	3	3	77	63	0		

注:表中的数据是本次工作模拟矿体切割两次时的数据(表中属性 0～4 即图 2 中不同色调表示的属性).

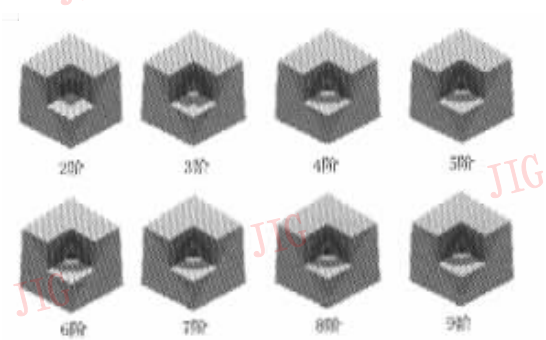

图 2　2 阶～9 阶矿体投影图
(切除 1/8 立方体,外层正方体为围岩)
(不同色调代表不同属性)

在实际操作过程中,压缩后,八叉树的每个节点至少应该有 3 个数据域,即 Morton 码、属性、Level 级,其中,每个节点就是一个立方体,且每个节点属性不尽相同. n 级八分体经过排序之后,按顺序对这些八分体进行分组判断(连续的 8 个元素为一组);如果某个组的所有元素属性一致,则合并成为一个节点存放,其 Level 为 $n-1$,Morton 码即为该数组中的第 1 个元素;如果该组元素的属性值不完全一样,则 8 个节点原样存放. 这样对 n 级八分体进行一轮压缩之后,八叉树就包含有 n 级和 $n-1$ 级节点,再对 Level 为 $n-1$ 的所有节点进行同样上述操作,将得到 $n-2$ 级节点;以此类推,同样的压缩过程一直进行到 1 级或不能再进行合并操作为止. 对于一个经过 n 次剖分的立方体最多需进行 n 轮压缩运算,最终八叉树所剩节点数目越少,压缩效率越高.

3　八叉树存储空间复杂度分析

用八叉树存储矿体信息所占用的空间主要与矿体的尺寸、复杂程度及采用的空间分辨率有关. 本次工作对具有 5 级属性的矿体进行了模拟实验,分级越粗,八叉树存储占用的空间越小,一般具有 4～5 级属性的矿体分级已够用[1]. 表 3 为八叉树存储和三维栅格存储占用的空间比较.

表 3　八叉树与栅格存储占用的空间对比

阶数	栅格存储 (Byte)	八叉树存储 (Byte)	压缩比	八叉树/栅格 (%)
1	108	140	0.77	129.9
2	164	385	0.42	238.1
3	612	1 365	0.45	222.2
4	4 196	4 550	0.92	108.7
5	32k	20.4k	1.56	64.1
6	256k	86.9k	2.95	33.9
7	2 048k	358k	5.72	17.5
8	16M	1.37M	11.68	8.56
9	128M	5.53M	23.15	4.32

由表 3 可见:

① 低阶时,出现八叉树存储占用空间比栅格存储占用空间还多的情况,这是由于本次工作存放栅格数据时,只记录其属性信息的缘故(其位置信息已暗含在存储属性信息时的存放规律之中),而按八叉树存储方式存放数据时,不仅要记录其属性信息,还要记录其存放位置信息,而且低阶时,由于划分粗糙,因此各立方体属性相同的可能性小,压缩的比例也相对要小. 但由表 3 中 1～4 阶数据的存储空间可见,由于两种方法实际占用的存储空间都很小,因此不会影响八叉树的实际应用.

② 从 5 阶开始,使用栅格结构存储数据所占用的空间与使用八叉树结构存储数据占用空间的压缩比呈递增状况,当 9 阶时,八叉树数据存储量只占栅

格数据存储量的 4.32%.这充分表明了使用八叉树存储数据占用空间小的特性.

③ 在实际应用中,特别是对于地学中的复杂地物而言,只分到四阶往往不能满足精度要求,因为不同切割阶数产生不同的表达精度.图 2 显示了切除掉八分之一立方体(为观察到矿体内部不同属性信息)的矿体,在不同阶数时的信息表达.2 阶分割,只表达出两种矿体属性;3 阶表达出 3 种属性;4 阶虽可表达出矿体内部的 4 种属性,但边缘呈锯齿状;从 5 阶开始,表达的矿体边缘逐步趋于平滑,由此可见,阶次越高,精度越高,八叉树压缩效率也会越高.

4 结 论

根据 Morton 码的特殊性,采用具有线性运行时间的计数排序法,7 阶时,比快速排序法运算速度还快 21 倍,可见是一种高效可行的算法.在此基础上,还对属性连续相同的编码进行了合并压缩处理,9 阶时,八叉树数据存储量只占栅格数据存储量的 4.32%,这充分表明八叉树存储数据占用空间小的特性.本次工作模拟了三维矿体的 2～9 阶线性八叉树表示,并根据该算法编写了一套用八叉树表示的具有矿体信息存储、查询检索、量算分析及其三维显示等功能的"3D_Octree"软件包,从而为深入开展三维 GIS 的理论研究和拓宽 GIS 的应用领域,起到了积极的推动作用.

参 考 文 献

1 Li R. Data structure and application issues in 3D geographic information systems[J]. Geomatica, 1994,48(3):209～224.

2 Irene Gargantini. Linear Octrees for fast processing of three-dimensional objects[J]. Computer Graphics and Image Processing, 1982,(20):365～374.

3 韩国建, 郭达志, 金学林. 矿体信息的八叉树存储和检索技术[J]. 测绘学报,1992,21(1):14～17.

4 徐士良. C 常用算法程序集[M]. 北京: 清华大学出版社,1994.

5 严蔚敏, 陈文博. 数据结构[M]. 北京: 机械工业出版社,1990.

6 潘金贵, 顾铁成. 现代计算机常用数据结构和算法[M]. 南京: 南京大学出版社,1994.

曹 彤 1970 年生,1999 年获北京师范大学地图学与地理信息系统专业硕士学位,现为北京联合大学应用文理学院信息科学系讲师,澳大利亚新南威尔士大学博士生.主要研究领域为计算机图象处理、三维 GIS 原理与方法、遥感技术及应用等.

刘 臻 1972 年生,1998 年获北京师范大学地图学与遥感专业硕士学位,现为北京师范大学计算中心讲师,资源所博士生.主要研究领域为 GIS 原理与方法、MIS 系统与应用、教育信息与技术等.