基于四叉树的多分辨率地形数据 3 维可视化

赵文吉[1] 胡卓玮[1] 陈永良[2] 莫 映[1]

[1](首都师范大学资源环境与地理信息系统北京市重点实验室,北京 100037)

[2](吉林大学综合信息矿产预测研究所,长春 130026)

摘 要 大场景的 3 维地形可视化模型是地理信息系统平台的重要组成部分。但至今大场景的 3 维地形数据可视化仍存在数据处理速度慢,显示滞后的问题,针对此问题提出了一种基于四叉树的多分辨率地形数据 3 维可视化算法,即首先以规则网格为基本处理单元,采用链式四叉树数据结构存储多分辨率地形数据;然后在此基础上,设计了动态 LOD 算法完成四叉树的高效遍历。在该算法中,采用了视景裁剪、三角扇形和内存池等优化显示技术,用以支持大场景 3 维地形数据的实时显示,并已在 Windows 平台上,用 VC++6.0 开发了大场景地形数据 3 维可视化演示软件。

关键词 大场景 地形数据 四叉树 LOD 算法 可视化

中图法分类号: P208 **文献标识码**: A **文章编号**: 1006-8961(2007)08-1457-06

Quadtree-based Three-dimensional Visualization for the Geomorphologic Data in Multiple Resolutions

ZHAO Wen-ji[1], HU Zhuo-wei[1], CHEN Yong-liang[2], MO Ying[1]

[1](*The Key Laboratory of Resources Environment and GIS of Beijing, Capital Normal University, Beijing* 100037)

[2](*Research Institute of Mineral Resource Prediction with Comprehensive Information, Jilin University, Changchun* 130026)

Abstract Three-dimensional visualization models for large-scale landscape geomorphology are the important parts of GIS platforms. In this paper, the problems related to the three-dimensional visualization of the geomorphologic data of a large-scale landscape are investigated. Regular grids serve as basic processing units for the representation of a large landscape geomorphology and the correspondingly multiple resolution data are saved in a link-listed quadtree structure. On the basis of this data structure, a dynamic LOD algorithm for traversing a link-listed quadtree is designed. In order to support the three-dimensional visualization of large-scale landscape geomorphology, the optimized visualization technologies, such as view frustum culling, triangle fans, and inner memory pool, are used in the algorithm. With the aid of VC++6.0, software for illustrating the three-dimensional visualization of large-scale landscape geomorphology has been developed based on Windows operating system.

Keywords large-scale landscape, geomorphologic data, quadtree, LOD algorithm, visualization

1 引 言

众所周知,3 维地形的表达一般多采用多边形网格的方法,尽管数量众多的多边形面片所表示的地形模型能够满足人们对地形真实性所提出的高要求,但受到现有计算机性能的限制而不能实时显示。尽管目前的计算机硬件已经具备相当高的水平,但在实时显示大场景的 3 维景观时,尤其系统要求具有较高的交互帧频率时,会出现处理速度慢、显示滞后的问题[1-3]。本文以规则格网为研究对象,讨论了地形数据的存储方式和基于四叉树结构的动态 LOD(level of

基金项目:国家高技术研究发展计划"863"项目(2002AA134074);国家自然科学基金项目(40471090);北京市自然科学基金项目(KZ200510028017)

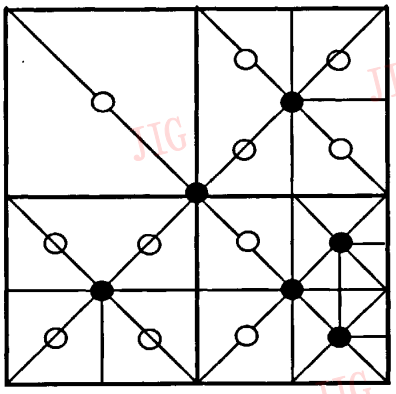
收稿日期:2007-03-28;改回日期:2007-04-28

第一作者简介:赵文吉(1967~),男,副教授,系主任。1998 年于长春科技大学获博士学位。主要从事空间信息技术在资源环境动态检测中的应用与国土资源信息化研究。E-mail:zhwenji1215@163.com

detail)算法,并在 3 维可视化技术中,使用了粗裁剪、三角扇形(triangle fans)和内存池等 3 维显示优化技术,同时用VC ++6.0开发了基于 Windows 平台的大场景地形数据 3 维可视化演示软件。

2 地形数据四叉树存储

由于数据存储结构直接影响算法的运行效率,因此合理的数据结构可以有效节省内存空间,使算法简单而高效。在数据结构中,树结构的存储有顺序结构和链式结构两种形式。其中,顺序结构比较常见,文献[1]~[9]中均采用顺序结构的数据存储方式。众所周知,一个二叉树可以用一个 1 维数组来存储,根据二叉树的结构特点和节点在 1 维数组中的排布顺序,可以很容易地建立起两者之间的对应关系。类似地,可以用两个 2 维数组来表示四叉树结构。其中一个数组用来按行列存储地形的高度值,这样四叉树节点的高度值就可以通过行列索引从数组中取得,如图 1(a)所示;另一个数组和高度值数组对应,用来标示四叉树节点的状态,如果节点需要分割,则标为 1,否则标为 0,而没有访问到的节点标为"不确定",如图 1(b)所示。图 1(a)中黑色节点代表该节点需要分割,白色节点则代表不用分割。图 1(b)中的问号代表"不确定"。



(a) 四叉树节点结构

?	?	?	?	?	?	?	?	?
?	?	?	?	?	0	?	0	?
?	?	0	?	?	?	1	?	?
?	?	?	?	?	0	?	0	?
?	?	?	?	1	?	?	?	?
?	0	?	0	?	0	?	1	?
?	?	1	?	?	?	1	?	?
?	0	?	0	?	0	?	1	?
?	?	?	?	?	?	?	?	?

(b) 四叉树节点状态

图 1 顺序结构的四叉树表示方法

Fig. 1 Representation of ordinal structural quad-tree

虽然上述顺序结构具有形象直观、处理简单的优点,但灵活性不够,因此,采用链式方法存储四叉树。与顺序结构相比,链式结构具有数据组织紧凑合理,不仅可以有效利用内存空间、操作灵活,而且具有很好的适应性,但链式结构的处理相对复杂些。

为了处理方便,以提高计算效率,在链式存储方式的四叉树数据结构中,每个节点除保存用于遍历树的指向其他节点的指针外,还保存本节点对应的地形区域中心点的高度值和 4 个边点的高度值。4 个角点的信息,可以通过访问父节点的边点信息得到。另外节点的尺寸信息和中心点的 x、y 坐标是可选的,且它们可以放在节点之外,在遍历节点的过程中通过计算得到。

对于单一采样精度的地形数据,可以依据上述数据结构先构建一棵四叉树,然后利用误差评判机制和 LOD 算法对其进行遍历。这种处理等价于对四叉树进行自顶向下的遍历,遍历过程在适当的节点处终止就可以把遍历过的节点构成一棵"用于显示的树"。这棵树通常是由不同精度的节点混合组成的。和原树相比,新的树是不完全四叉树,而原树则是完全四叉树。如果原树本身是不完全四叉树,那么就可以将不同采样精度的地形数据存储在同一棵四叉树中,极大地提高了地形数据描述的灵活性。这样就可以在描述整个 3 维场景时采用比较粗略的采样数据,而在局部需要细致描述的地方采用比较精细的采样数据。另外,由于随着场景的移动,可以将部分数据从树中"摘除"(即移除子树),同时"加载"新的数据(即添加子树),因此,在没有利用 LOD 算法对树进行遍历之前,就已经对地形数据进行了"优化",这种优化思想本质上和 LOD 算法是一致的。在实现本文的 LOD 算法时,对地形数据的组织方式进行了有效封装,使得四叉树存储地形数据的

形式不影响 LOD 算法本身,极大地提高了算法的灵活性。

3　动态 LOD 算法

基于四叉树的 LOD 算法,需要对四叉树进行两次遍历,其中第 1 次遍历用于更新节点,生成"供显示用的树"(原树的一棵子树);算法在对树进行第 2 次遍历时,才能实现节点对应地形的绘制,而本文提出的四叉树动态 LOD 算法,只需对树遍历一次。

算法按广度优先的次序自上而下遍历四叉树,并用两个队列存储需要处理的节点。其中一个队列用于存储当前要处理的节点,这些节点均位于树的同一层,另一个队列用于存放下一层待处理的节点,这些节点均位于树的下一层。处理时,首先从存放当前节点的队列中读取节点进行判断,如果节点需要继续分割,则将分割后的子节点放入另一个队列。处理完当前层的所有节点后,若存放当前节点的队列为空,则将这两个队列作一下交换,就可以处理下一层节点了。

对于那些不需要继续分割的节点,可将其交由 OpenGL 进行渲染处理,而对于不可见的节点(即位于视景体外的节点),可将其略过。这一处理方式被称为粗裁剪。检查一个节点时,由于与该节点位于同一层的其余节点的状态都已确定,因此,很容易确定当前节点是否需要继续分割。除不可见的节点外,如果节点满足分割条件,则称节点处于 Enable 状态。另外,还需考虑邻接节点的精度误差是否与待分割节点一致,如果不一致,则会出现"裂缝"现象。解决裂缝的办法有很多,这里采用的是一种简单的方法,即强制要求彼此邻接的节点属于同一分辨率。这样的处理方式可绝对避免裂缝的出现。

为了进一步提高效率和降低处理的复杂程度,在本文算法中,对节点的分割规则做了如下优化:当一个节点的 4 个子节点中的任何一个需要继续分割时,这 4 个子节点都将进行强制分割。在算法实现中,当一个节点需要分割(即处于 Enable 状态时),需强制将其 4 个子节点也都设置为 Enable 状态,并将这些子节点放入下一层次的队列中,以等待后续处理。这使得生成的网格数目有所增加,即使地形有所细化。虽然其在地形简化程度方面有所降低,但实际实验的结果表明,由于算法效率得到了很大的提高,因此是可取的。

在本文算法中,除了节点评价系统中的判断式之外,为了避免"裂缝"的产生,还需要增加一个限制条件,即与当前节点相邻的 4 个节点必须都处于 Enable 状态(相当于满足邻接节点分辨率相同这个条件)。只有同时满足这两个条件时,节点才允许被继续分割。至于已经位于最底层的节点,亦即达到了实际物理采样精度的叶节点,则将直接送入渲染通道进行绘制。

4　3 维可视化

4.1　粗裁剪

基于视景体的裁剪技术包括:视景体裁剪(view frustum culling)、背面裁剪(back face culling)和基于阻挡关系的裁剪。在这些方法当中,以前面两种方法实现起来较为简单,同时容易产生高效的算法。这里主要讨论视景体裁剪技术。

视景体裁剪技术和层次细节技术相结合将会使 3 维地形模型得到进一步简化。其主导思想就是,首先将场景中的不可见部分(即不在视景体内部的部分)利用裁剪算法去除;然后再对剩余部分利用 LOD 算法,根据实际需要忽略不必要的细节部分。在实际操作时,可以选择简洁高效的裁剪算法来对 3 维地形和其他实体数据进行粗略裁剪,先期略去很多不必要的多边形面片。这样做虽然不能达到十分精确的裁剪效果,但在此基础上再进一步采用 LOD 技术,将会省去很多不必要的操作;其最后送入渲染通道的数据量将会显著减少。由于 OpenGL 在最终渲染之前所做的裁剪处理需耗费系统时间,故这种预先做一次快速粗裁剪后再交由 OpenGL 处理的方法,能提高图形的显示速度。

3 维实体只有位于视景体内部才能被看到,而在透视投影下,视景体是一个平头锥体。为了实现粗裁剪的目的,首先需要知道构成该视景体的 6 个平面方程。有了平面方程之后,如要判断一个位于世界坐标系中的点是否处于视景体内部,则只需将该点的 3 维坐标值代入方程中进行判断即可。假设指向视景体内部的方向为平面的正方向,如果 6 个方程的计算值均大于零,则表示当前所测试的点位于视景体的内部,否则便是位于视景体的外部。其基本问题是如何确定视景体 6 个平面方程的系数。

在 OpennGL 中,代表视景体的平头锥体在经过模视变换(model-view transform)和透视变换(project

transform)之后,将会成为一个范体(如图 2 所示)。

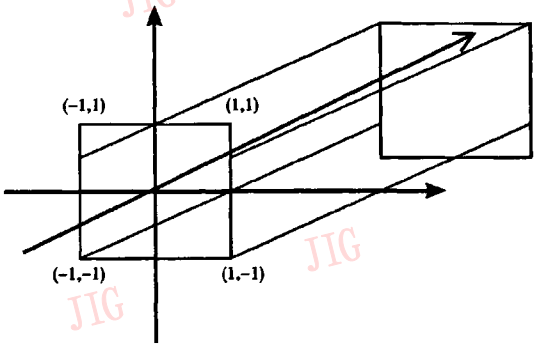


图 2 视景体经变换后所形成的范体

Fig. 2 Transformation of view frustum

假设世界坐标系中平面方程为 $Ax+By+Cz+D=0$,经过变换后的平面方程为 $\hat{A}x'+\hat{B}y'+\hat{C}z'+\hat{D}=0$,位于世界坐标系中的某一点 P 的齐次坐标为 $(x,y,z,1)$,经过变换后的齐次坐标为 $(x',y',z',1)$。则有如下等式:

$$\begin{aligned}(x,y,z,1)(A,B,C,D)^{\mathrm{T}}&=0\\(x',y',z',1)(\hat{A},\hat{B},\hat{C},\hat{D})^{\mathrm{T}}&=0\end{aligned}\tag{1}$$

假设模视变换和投影变换矩阵分别为 M 和 P,则有

$$(x,y,z,1)\times M\times P=(x',y',z',1)\tag{2}$$

由上述公式可得到如下结果:

$$M\times P\times(\hat{A},\hat{B},\hat{C},\hat{D})^{\mathrm{T}}=(A,B,C,D)^{\mathrm{T}}\tag{3}$$

由此,只需要将范体所对应的 6 个平面方程的系数代入上述等式,便可以求得世界坐标系中视景体的 6 个平面方程的系数。范体的 6 个平面方程如下:

$$\begin{cases}\text{near}: & 0x+0y+z+0=0 & (z=0)\\ \text{fear}: & 0x+0y-z+1=0 & (z=1)\\ \text{left}: & x+0y+0z+1=0 & (x=-1)\\ \text{right}: & x+0y+0z-1=0 & (x=1)\\ \text{top}: & 0x+y+0z-1=0 & (y=1)\\ \text{bottom}: & 0x+y+0z+1=0 & (y=-1)\end{cases}\tag{4}$$

在求得视景体的 6 个平面方程之后,就可以根据这些方程来判断 3 维实体的可见性。这里,3 维实体是指四叉树节点所对应的部分地形。在具体判断时,由于 3 维实体具有尺寸大小,而不是一个点,因此,对其所进行的判断要比简单的 3 维坐标点可见性的判断复杂一些。具体而言,需要考虑该 3 维实体和视景体本身的空间位置关系,这些关系包括相容、相交和相离 3 种情况(如图 3 所示)。图中的方块表示 3 维实体,梯形表示视景体。白色方框(A)位于视景体内部,属于相容情形;灰色方框(B)位于视景体边沿,属于相交情形;黑色方框(C)位于视景体外部,属于相离情形。

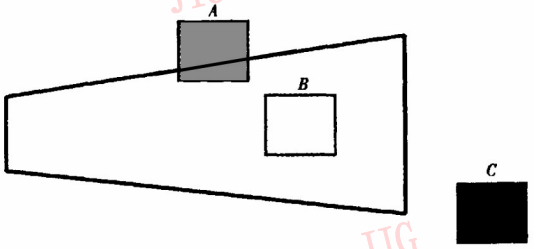


图 3 节点包围盒与视景体的 3 种位置关系

Fig. 3 Spatial relationships between view frustum and envelop box

在判断时,可引入包围盒的概念。包围盒一般采用长方体或者球体等一些简单的几何形体,它们内含 3 维实体,这样对包围盒的可见性判断,可以简单地认为等价于 3 维实体的可见性判断。对于节点而言,应首先检查视点是否在节点所在的包围盒内,如果是则认为可见,否则再将节点中心点坐标代入平面方程,由于此处使用球体作为节点的包围盒,其判断公式为

$$Ax+By+Cz<-r\tag{5}$$

式中,x,y,z 为节点中心点坐标,r 为球体半径,它与节点对应地形的尺寸大小成正比关系。

4.2 三角扇形

四叉树的固有特性使得 LOD 算法很适合采用三角扇形(triangle fan)优化技术。所谓三角扇形,就是指存在若干三角形,它们彼此相联,并共享同一个节点[10,11]。

OpenGL 在绘制这类三角形时,无需将每个三角形都各自绘制一遍,而是采用更加快速的优化方法进行绘制。基于四叉树结构的 LOD 算法,在对地形细分之后,树上的每个节点,都具有三角扇形特征。并且从前面算法描述中可以看到,在绘制某个节点时,由于其邻居节点的状态也都已经确定,因此,可以利用这一点来加快系统的绘制速度。具体绘制时,可以以节点的中心点为中心,从左上角点开始,顺时针方向遍历各个角点和边点,其中角点自然要参与绘制,而边点则需判断邻居节点是否是 Enable 状态,若是则参与绘制,否则跳过。

4.3 静态信息的预先计算

为了进一步提高绘制速度,在实施 LOD 算法之前,可以将一些能够确定的静态数据预先计算出来,并保存起来,而剩下的内容则需要在系统运行时才可以确定。比如误差信息(variant)的计算便可以在系统初始化时确定,其前提是地形始终保持不变。如果地形发生变化,则相应的误差信息也需要更新。另外,光照所需要的结点法向量也可以预先计算好。

4.4 内存优化手段

在 LOD 算法的实现过程中,由于实时交互的特性所致,对于四叉树的更新可能需要频繁分配和回收内存。大家知道,虽然单次分配与回收内存所要花费的时间是微不足道的,但是由于在显示每一帧的时候,都需要做相应的内存分配和回收操作,因此这种效应的累积很可能是相当可观的。与此同时,随着时间的推移,对内存资源进行频繁的分配与回收操作,最终将会导致小片内存的激增,由于这种现象会使得后续内存的分配与回收变得越来越慢,从而导致了系统整体性能的下降。

解决内存分配问题的办法是采用内存池技术。其具体的实施办法很多,其中一种简单的办法是:在系统启动时,预先分配一块足够大的全局内存,该内存由相同大小的若干内存区块构成,并有一个指针指向当前第 1 块可用内存区块。初始时,指针指向内存的首部;然后在程序请求分配内存时,再从该内存区中顺序划出一个区块交由程序处理,并使指针指向下一个相邻区块。通常程序不需要显示的释放内存区块,直到时机适合时才"释放"(比如,所有分配的区块都已无效),此时,只需将指针移动至内存首部即可。这么做并非真正释放内存,因为最终的内存释放动作需要等到系统运行结束时才进行。这样一来,内存的分配和回收就只发生于系统的启动和结束这两个时刻了。

前面在讨论 LOD 算法的时候,曾提到需要用两个队列来存放待处理节点的相关信息,由于这些节点相关信息属于中间结果,其所占内存需要动态分配,并且在绘制完一帧后,本次绘制所产生的这些中间结果就没有用了,因此可以将其所占内存一次性"回收"。在笔者开发的算法代码中就采用了上述简单的内存池方案。

4.5 几何形变处理

由于随视点移动时对地形细节的省略处理,因此场景在随用户视点的行进过程中,一些原来被省略的节点可能会突然跳出,而另一些原来存在的节点则可能会突然消失,由于这种现象会在视觉上给人以一种跳跃感,因此把它称为几何形变(Geomorphing)。

为了使结点高度值的变化在视点移动过程中不产生明显的跳跃感,一种很直观的思路是,由于可通过插入中间值,使得高度值的变化趋于缓和,因此,需要引入一个取值范围介于 0.0 至 1.0 的形变值(事实上就是一个权数),对于结点的高度值(比如某个节点的边点),若要从较粗精度的近似值得到实际值,采用渐变的策略,使其逐渐达到实际值。这一过程是通过每隔一段时间间隔调节一次形变的取值来实现的。具体如下:

$$m_z = (ma_z) + ((1-m)p_z) \tag{6}$$

式中,m_z 为节点 z 的高度值,m 为形变系数,a_z 为节点 z 实际高度值,p_z 代表 z 点形变前的高度值。

令形变值从 0.0 逐渐增大至 1.0,就可以使节点的高度值渐变至节点实际的高度。实际上,这种方法可能比较复杂而难于达到实用程度,其计算代价可能对系统性能构成较大的影响,因而是得不偿失的。为此,在本文算法中,未采用这种消除几何形变的处理手段,而是通过适当调节 C_1 和 C_2 这两个常量值来达到一定程度的降低几何形变的效果。

4.6 系统性能的代码级优化

由于前述的优化方案,多数是在"宏观"尺度上的优化,因此优化后系统的性能提高多少,除了诸如帧频率之类的较为粗略的衡量指标之外,只能依靠主观的视觉评判了。如果采用专门的性能测评工具来对系统的表现进行测评的话,则可以从"微观"尺度——代码一级的层面,对系统性能有一个更为精确的认识和把握,而且根据性能测评工具得到的分析结果,并对系统进行代码一级的优化,就会变得更有针对性[12,13]。

在 Windows 平台上,使用 Microsoft Visual C++ 6.0进行程序的开发,并使用 Microsoft Visual studio 自带的性能测评工具——Profile,对程序的运行效果进行了初步的评测。同时根据评测结果,对程序进行了代码级优化,进一步提高了整个系统的性能。

5 结　论

大场景地形数据 3 维可视化技术研究是一个较

为前沿的课题,由于该技术直接受计算机软硬件系统性能的制约。因此如何在现有的技术条件下,研制出大场景 3 维可视化的高效算法并开发出实用的 3 维可视化软件系统,一直是研究者们探索研究的目标。笔者在地形数据存储结构研究的基础上,开发了基于 Windows 平台的大场景 3 维可视化演示软件,是大场景 3 维可视化研究的有益尝试。

参考文献(References)

1 Li Qing-quan, Li De-ren. Data structure in 3D geographical information system[J]. Journal of Wuhan Technical University of Surveying and Mapping, 1996, 21(2): 128 ~ 133. [李清泉, 李德仁. 三维地理信息系统中的数据结构[J]. 武汉测绘科技大学学报, 1996, 21(2): 128 ~ 133.]

2 Li De-ren, Li Qing-quan. A type of GIS hybrid data structure[J]. Acta Geodaetica et Cartographica Sinica, 1997, 26(2): 130 ~ 132. [李德仁, 李清泉. 一种三维 GIS 混合数据结构研究[J]. 测绘学报, 1997, 26(2): 130 ~ 132.]

3 Li Pei-jun. Three dimensional modeling and visualization for stratified geological objects[J]. Earth Science Frontiers, 2000, S2: 271 ~ 275. [李培军. 层状地质体的三维模拟与可视化[J]. 地学前缘, 2000, S2: 271 ~ 275.]

4 Ruan Qiu-qi, Luo Min. Three dimensional reconstruction algorithm of geological structure[J]. Journal of Northern Jiaotong University, 1995, 19(40): 449 ~ 451. [阮秋奇, 罗民. 地质构造三维重建算法研究[J]. 北方交通大学学报, 1995, 19(40): 449 ~ 451.]

5 Cao Dai-yong, Li Qing-yuan. Study on the 3D visual model of geological structure[J]. Geology and Prospecting, 2001, 37(4): 60 ~ 62. [曹代勇, 李青元. 地质构造三维可视化模型探讨[J]. 地质与勘探, 2001, 37(4): 60 ~ 62.]

6 Chai He-jun, Huang Di-long. Research on the 3D extension model of geological structural surface[J]. Hydrogeological Engineering, 1999, 4: 16 ~ 17. [柴贺军, 黄地龙. 地质结构面三维扩展模型研究[J]. 水文地质工程, 1999, 4: 16 ~ 17.]

7 Shen Da-yong, Mao Shan-jun. Experimental research on the 3D geological objects[J]. Computer Engineering, 2001, 27(3): 49 ~ 50. [沈大勇, 毛善军. 地质体三维模型试验研究[J]. 计算机工程, 2001, 27(3): 49 ~ 50.]

8 Zhang Min-na, Lin Guo-zhang. Geology property modeling framework[J]. Control and Automation, 2001, 17(1): 62 ~ 63. [张敏娜, 林国璋. 地质属性模型框架[J]. 微计算机信息, 2001, 17(1): 62 ~ 63.]

9 Du Pei-jun. 3D GIS data structure and visualization based on mine properties[J]. Journal of China University of Mining and Technology, 2001, 30(3): 238 ~ 241. [杜培军. 顾及矿山特性的三维 GIS 数据结构与可视化[J]. 中国矿业大学学报, 2001, 30(3): 238 ~ 241.]

10 Yang Qin, Xu Yong-an, Chen Qi-ming. A method of triangulation for scattered point set in arbitrary plan field[J]. Journal of Software, 1998, 9(4): 243 ~ 245. [杨钦, 徐永安, 陈其明. 任意平面域离散点集的三角化方法[J]. 软件学报, 1998, 9(4): 243 ~ 245.]

11 Li Qing-yuan. Research on the Topological Relationship of 3D Vector Data Structure[D]. Beijing: China University of Mining, 1995. [李青元. 三维矢量结构 GIS 拓扑关系研究[D]. 北京: 中国矿业大学, 1995.]

12 Xia Yan. Research on the 3D Vector Structural Geological Model and Its Visualization[D]. Beijing: China University of Mining, 1997. [夏炎. 三维矢量结构地质模型及其微机可视化图形显示系统研究[D]. 北京: 中国矿业大学, 1997.]

13 Zhang Jian-qiu. Research on the 3D Geological Modeling and Visualization System Development[D]. Nanjing: Nanjing University, 1998. [张剑秋. 三维地质建模与可视化系统开发研究[D]. 南京: 南京大学, 1998.]