

文章编号:2095-6134(2015)03-0409-07

简 报

# 城市三维模型海量数据动态组织调度方法<sup>\*</sup>

王 锋<sup>†</sup>, 潘德吉, 王 俊

(中国科学院遥感与数字地球研究所 遥感科学国家重点实验室, 北京 100101)

(2014 年 4 月 15 日收稿; 2014 年 8 月 18 日收修改稿)

Wang F, Pan D J, Wang J. Dynamic dispatching and organization of massive data of urban 3D model[J]. Journal of University of Chinese Academy of Sciences, 2015, 32(3): 409-415.

**摘 要** 城市三维模型的海量特性和当前计算机硬件瓶颈给模型的实时渲染提出了较大的挑战. 改进三维地理信息系统(GIS)中数据的组织与管理方式是提高三维场景可视化效率的有效途径, 本文提出基于四叉树空间索引对数据进行组织, 基于视点的层次模型技术(LOD)对数据进行动态调度, 同时提出一种将纹理作为渲染单元重构网格数据的方法, 对粗糙模型的渲染进行批处理, 有效降低了耗时的渲染次数. 实验证明, 该方法是高效可行的, 渲染帧率稳定在 30 帧左右, 能够满足海量城市三维模型的可视化和实时交互需求.

**关键词** 三维模型; 三维地理信息系统; LOD; 四叉树; 批处理

中图分类号: P208      文献标志码: A      doi: 10. 7523/j. issn. 2095-6134. 2015. 03. 018

## Dynamic dispatching and organization of massive data of urban 3D model

WANG Feng<sup>†</sup>, PAN Deji, WANG Jun

(State Key Laboratory of Remote Sensing, Institute of Remote Sensing and Digital Earth, Chinese Academy of Science, Beijing 100101, China)

**Abstract** Massive data of three-dimension urban model and the bottleneck of computer hardware are great challenge to real-time rendering and visualization. The efficiency of the visualization can be significantly improved by organizing and managing the massive data effectively. This work proposes a method for organizing the three-dimension model using quad-tree spatial index and dispatching the data using the technique of view-based level-of-details (LOD). Additionally, a method for reconstructing the mesh using the texture as the render unit rather than the single model is proposed to accelerate the rendering process of rough models. Experimental results show that the proposed method works efficiently with a stable frame-rate of 30 fps and satisfies the requirements of the visualization of massive model data and the real-time interaction.

**Key words** three-dimension model; three-dimension GIS; LOD; quad-tree; batch processing

<sup>\*</sup> 国家“863”计划项目(2012AA12A401, 2013AA12A403)资助

<sup>†</sup> 通信作者, E-mail: luoying\_gis@126.com

数码城市是城市 GIS 向多维、动态和网络化方向发展的结果,是实现城市景观仿真、城市发展历史演进过程再现、城市灾害事故和突发事件动态模拟和城市综合社会效应动态模拟的基础<sup>[1]</sup>. 城市三维模型作为三维 GIS 的重要组成部分,不仅能够逼真地对现实世界进行模拟仿真,而且在空间分析、城市规划、辅助决策等方面都发挥了重要作用<sup>[2]</sup>. 然而,城市三维模型一般具有数据量庞大、结构复杂等特点;现有计算机硬件虽然有较快发展,但依然无法满足海量模型数据的一次性调度和显示. 对于一个城市而言,三维建筑物模型数据往往是从几个 GB 到几十 GB 不等,同时要涉及到地形等海量纷繁复杂的空间数据,这远远超出了普通计算机的存储和管理能力,因此,如何对数据进行合理、高效的组织调度来构建一个复杂场景的可视化环境成为 3D GIS 所亟待解决的关键技术. 不仅如此,随着计算机技术的发展,对复杂场景逼真度的要求也越来越高,如何利用图形显示卡的特性对其渲染进行优化,提高渲染帧率是目前城市三维地理信息系统所面临的挑战.

前人对海量模型的高效渲染做了大量研究和工作的. 在基于规则网格方面, Lindstrom 等<sup>[3]</sup>、Hoppe<sup>[4]</sup>提出视点相关的连续细节层次实时高度场绘制算法,对其后的研究有重要影响. Lee 和 Nevatia<sup>[5]</sup>、Rau 等<sup>[6]</sup>、Li 等<sup>[7]</sup>、Fakir 等<sup>[8]</sup>详细研究城市模型层次细节模型 (LOD) 的动态生成,着重于不同层级模型的动态构建和提取,有效减少了渲染三角形的数目,但是无法降低显卡的渲染次数和状态切换频率. 基于规则地形网格划分和四叉树的模型数据组织方式<sup>[9-10]</sup>,在某种程度上减少了模型查找和 I/O 的时间,但是不能很好地解决模型纹理冗余重复渲染问题. 杨卫军<sup>[11]</sup>使用基于数据动态分页的动态装载方法来提高渲染效率,该方法建立前后台 2 个数据页缓冲区,通过多线程技术实现 2 个缓冲区之间数据内容的交换. 但这种方式会影响数据的更新与查找效率. Arnaud 和 Cervelle<sup>[12]</sup>、Remondino<sup>[13]</sup>、刘纪平<sup>[14]</sup>也都对城市模型数据的组织和调度进行了详细的研究并提出具体的解决方案,提供了有价值的指导和建议.

基于现代图形显示卡的特性,本文提出一种海量模型动态方法. 该方法采用四叉树索引对模型数据进行组织,方便进行数据的查询和剔除,根

据视点将模型分级渲染,有效减少要渲染的三角形的数目,同时使用基于纹理的批处理方式减少显卡的状态切换频率,将使用相同纹理的网格数据分批渲染来提高渲染效率.

## 1 基于四叉树索引的模型数据组织

空间数据组织就是对空间数据进行合理的规划并建立空间索引,以提高空间数据的检索效率<sup>[15]</sup>. 三维 GIS 中场景对象数量巨大,结构繁杂,因此在实时场景绘制中,需要进行裁剪,也就是需要在所有数据中选择那些符合条件的记录. 数据的选择自然离不开空间索引的建立. 空间索引是指依据空间对象的位置和形状,按一定顺序排列的一种数据结构. 其中包含空间对象的概要信息,如对象标识、最小外包矩形 (MBR) 等. 对于空间索引,主要有格网索引、B + 树、R 树、四叉树等<sup>[16]</sup>.

地形、影像等空间数据的组织主要采用基于格网划分的方式,但建筑物模型由于其自身具有的不规则性,采用格网划分方式势必会在格网边界处产生大量的分割. B + 树是一维索引,无法处理空间数据中的二维和多维的空间数据. R 树是 B 树在多维空间上的自然扩展,适合于多维空间查询,但由于空间数据分布的偶然性,使得各层节点 MBR 容易重叠,导致实际执行空间查询时,会产生多个分支查询,很大程度上降低了空间查询的效率,因此也不太适合随机分布的建筑物模型.

与 R 树不同的是,四叉树是属于基于空间划分组织索引结构的一类索引机制. 在内存中的层次树状结构中,其查询效率较高. 本文方法需要在程序中动态实时生成城市模型的空间划分,从树构建的复杂度和查询效率的方面考虑,本文采用四叉树索引结构.

构建四叉树索引的基本思路是:将模型的 MBR 存储在完全包含它的最小矩形节点中. 如此,每个三维模型只在树中存储 1 次,避免了存储空间的浪费. 四叉树索引构建步骤如下 (见图 1).

1) 计算所有模型链表中所有模型的 MBR,将其作为根节点的 MBR.

2) 遍历链表中所有模型,将根节点作为当前节点,检测模型 MBR 和当前节点的子节点 MBR 的拓扑关系. 如果模型 MBR 与子节点 MBR 边界相交,将模型加入到当前节点;如果模型 MBR 在

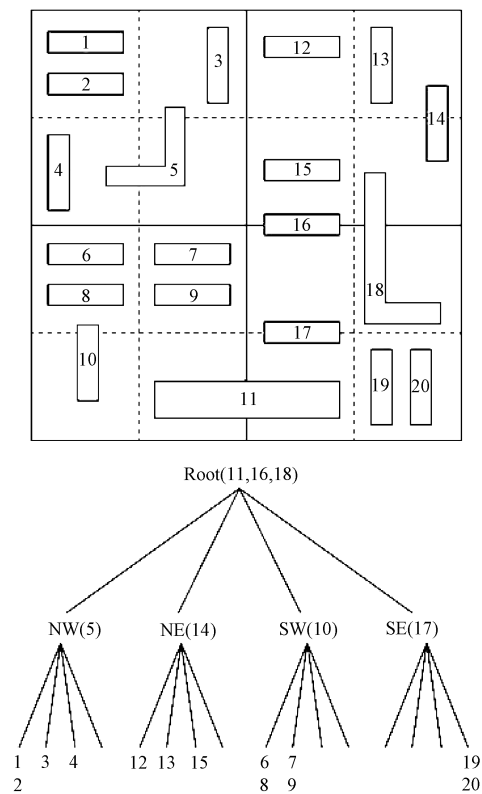


图 1 四叉树生成示例  
Fig. 1 Generation of quad tree

某个子节点内部,则将该子节点作为当前节点,递归调用步骤 2),直到子节点 MBR 小于 100.

采用四叉树机制对模型数据建立空间索引,能进行高效的视域体裁剪避,减轻显卡渲染的负担;还可以有效解决数据冗余问题,缩短数据检索和数据缓存的时间.

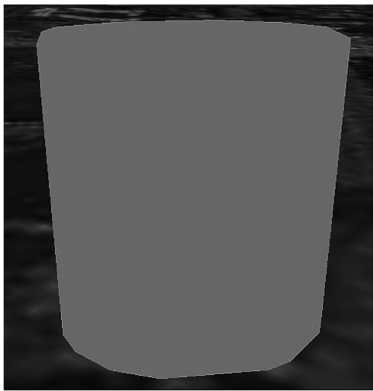
## 2 模型场景 LOD

### 2.1 模型数据及其分层

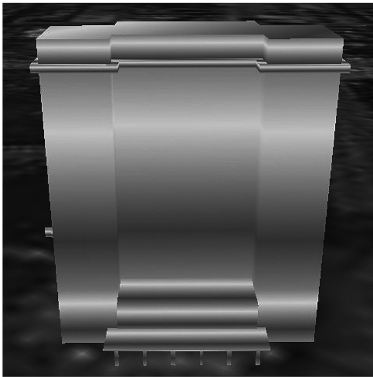
城市三维模型主要由模型几何数据、模型纹理和模型属性数据构成. 模型几何数据指模型的结构即网格数据,纹理数据即模型表面纹理,属性数据是从现实内容对模型数据的补充,如建筑物名称、高度等.

天津 60 多万个模型数据中包括 2 万多个精细模型,这些模型的网格数据顶点多、纹理精细,如果要将这些模型和粗糙模型一同加载到内存中,不仅对内存的消耗巨大,同时载入过程也非常耗时. 因此,本文根据模型和视点的距离将模型数据分为 3 个层次. 第 1 层模型使用其  $k$ -DOPs 包围盒作为网格数据;第 2 层使用模型原始网格数据,

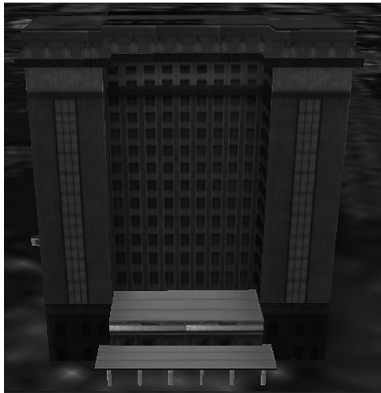
纹理使用空白纹理;第 3 层使用模型的原始网格数据和纹理数据进行渲染. 图 2 展示了 3 层模型的渲染效果.



(a) 模型的  $k$ -DOPs



(b) 不带纹理的模型网格



(c) 带纹理的模型网格

图 2 模型的 3 个 LOD 层级

Fig. 2 Three levels of the model

包围盒是物体的一个近似表示,其生成主要有球体包围盒、AABB、OBB、 $k$ -DOPs 等几种方式. 建筑物模型形状的多样性和真实性需要选取一种方式,使其生成的凸包能和原始网格有最大的相似性.  $k$ -DOPs 正好能够满足这一要求,随着  $k$  的增大, $k$ -DOPs 与物体的凸包越来越相似. 因此本文采用  $k$ -DOPs 方式生成模型的包围盒.  $k$ -DOPs 是一种特殊的包围盒,它的所有面的法向量均来

自一个固定方向向量集合. 设  $N = \{n_1, n_2, \dots, n_k\}$  是一个包含  $K$  个固定方向向量的集合, 对任意一个给定点集  $X$  都能生成  $k$  个以  $N$  为法向量的超平面来包裹点集  $X$ , 对平面求交, 所有平面的交集则为  $k$ -DOPs 的实际体积. 图 3 介绍一个茶杯  $k=8$  DOPs 的生成过程. 图中, 构建了以 8 个方向为法向量的平面, 依次两两获取 8 个平面的交点, 并将其依次连接起来, 就构成平面上的外包多边形.

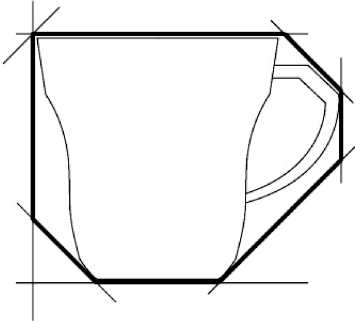


图 3 一个杯子的 DOPs 生成 ( $k=8$ )  
Fig. 3 Generation of DOPs( $k=8$ ) of a cup

2.2 视点相关的 LOD 调度

空间数据调度就是确定在什么时间, 采用什么方式, 调入还是调出空间数据的一个过程<sup>[10]</sup>. 海量三维模型数据无法一次性载入内存, 需在漫游时根据绘制需要动态分批调入, 同时由于网络和磁盘 I/O 操作花费时间较长, 因此, 本文采取多线程和实时缓存的方式将模型数据从服务器到内存中的操作放到后台进程, 防止在同一线程中该操作引起的调度延迟.

本文采用基于视点的 LOD 调度算法, 根据模型和视点的距离动态调用不同层级的模型数据进行加载. 图 4 阐述了这一过程.

1) 用户移动视点时, 整个模型场景需要更新, 根据 2.1 节建立的空间四叉树结构和当前视点位置做视域体裁剪, 剔除所有看不见的模型, 获取模型子集  $E$ .

2) 根据子集  $E$  里每个模型实体和视点距离确定模型的目标层级 TL, 将模型的 TL 和当前层级 (CL) 比较, 如果  $TL > CL$ , 则把模型加入到粗糙列表,  $TL \leq CL$  则把模型加入到加精列表.

3) 根据列表实体确定所需模型网格数据和纹理数据, 先在本地缓存查找, 如果找到则直接 I/O 调入内存; 如果没有找到, 则发送资源请求到

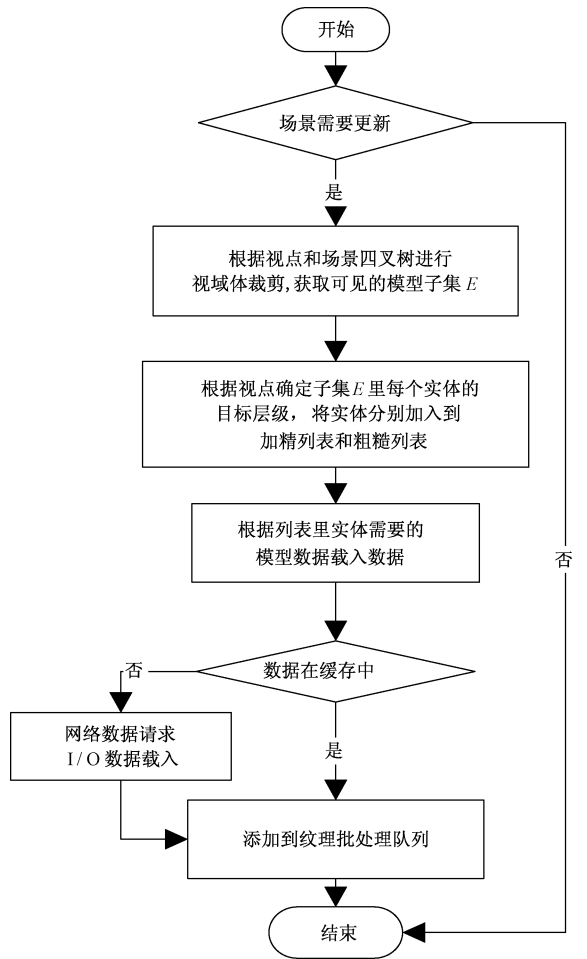


图 4 三维模型调度的流程图  
Fig. 4 Flow chat of model dispatching

服务器请求相关数据, 下载到缓存后载入内存.  
4) 将所有载入的数据按层级加入到纹理批处理队列, 供批处理渲染.

3 模型渲染方法

在精度不高的情况下, 城市三维模型的纹理决定了表达城市精细特征的精度. 纹理数据能极大地提高城市三维模型的细节层次和真实感, 有助于提高三维可视化的效果<sup>[17]</sup>. 三维模型的纹理一般不止 1 个, 纹理数据一般也远大于几何数据. 然而, 为了有效降低模型构建者的工作量, 同时减少模型数据量, 在城市三维模型中, 对于非地标性建筑如普通民宅和道路、树木等模型, 建模者往往使用相同的纹理. 因此, 可以根据模型数据的特性选取合适的渲染方式来加速模型渲染过程.

3.1 基于模型网格的渲染方式

传统的模型渲染方式将模型作为一个渲染单



元进行数据的加载和渲染,这种方式适合少量精细模型的渲染.随着模型数量的增大,重复纹理数量的增多,渲染过程中重复耗时的渲染操作会相应地大量增加.分析天津城区的 600 000 个模型数据,发现粗模中大量模型使用同一纹理,如民居的墙壁.如果我们将每个模型网格作为一个渲染单元,在渲染时每个模型调用一次图形接口中 DrawPrimitive( DP)函数,场景中 60 万个模型在使用不同纹理时需要调用 60 万次 DP,在使用同一张纹理时同样需要调用 60 万次 DP. DP 操作是整个渲染管线中最耗时的操作,如果采用这种渲染方式,显而易见渲染效率会比较低下.因此,为解决这一问题,本文提出基于纹理的批处理渲染方式来对场景中模型进行渲染.

3.2 基于纹理的批处理方式

基于纹理的批处理渲染方式是将纹理贴图作为一个渲染单元,根据该纹理重构场景中的网格数据进行渲染.这种方式在 DP 操作之前,需要根据纹理对模型网格数据进行重构.这会带来一定的计算复杂性,但在渲染大量重复模型的场景中能够有效降低渲染的时间消耗,提高渲染效率.2.2 节介绍了模型的动态调度,每一帧调度结束后将模型数据分别添加到纹理批处理队列.之后,根据队列里的纹理贴图重构模型顶点数据,提取在模型子集  $E$  中所有使用该纹理作为贴图的模型顶点,将模型的每个面根据纹理重新归类,构建顶点缓冲进行渲染.

4 试验结果和分析

4.1 试验平台和数据

基于上述研究,依托公安部天津警用地理信息系统平台项目,我们在三维数字地球平台 Geobeans3D 中实现并验证了本文方法.该平台是一个建立在多级全球数字高程模型( DEM)基础上,以海量卫星数据为主体、海量地理数据和文本数据交互式的数字地球三维网络地理信息系统.平台使用的 DEM 数据的空间分辨率全球范围是 30 m,局部是 10 m;影像数据分辨率全国是 2.5 m,部分地区是 0.6 m;矢量基础地理数据全球是 1:100 万、全国是 1:25 万、部分城区是 1:2 000.平台基于 C#和 DirectX 9.0 图形 API 实现,并使用 GPU 计算对三维空间分析、大规模地形渲染进行加速<sup>[12]</sup>.

实验使用的数据是由天地图提供的天津城区的三维模型数据,主要包括建筑物模型和地物模型 2 类.目前天津全域建筑总数达 60 万栋左右,主要覆盖市内 6 区、环城 4 区、滨海新区及其他各区县主城区.模型坐标系采用全国通用的 WGS84 坐标系,文件格式为 \*.max 和 \*.flt 格式.表 1 介绍了所提供的数据.

表 1 实验模型数据的详细介绍

Table 1 Details of experimental data

分类	描述	数量/个
建筑模型	市中心区及各区县主城区的地标性建筑、大型商业区、写字楼、大厦及主干道沿线两侧	48 万
	建筑物居民住宅、普通办公楼、工矿厂房等	
	地物模型 立交桥、过水桥、树木、路灯等	
		11 万

4.2 试验结果和分析

本文在 3 台测试用机上测试了天津和平区和河西区的 2 万多个建筑物模型数据,表 2 详细描述了测试机的配置,表中数据显示测试机 1 的配置最高,测试机 2 次之,测试机 3 最差.三维应用程序的效率可以用帧率( frames per second, fps)这一指标来衡量,帧率表示图形处理器处理场景时每秒钟能够更新的次数.高的帧率可以得到更流畅、更逼真的动画.一般来说,对于三维游戏,30 fps 是可以接受的,电影一般在 24 fps 左右.本文用 Fraps3.1 测试实验程序的帧率,实验结果见表 3.

表 2 3 个测试机的详细配置

Table 2 Configurations of three experimental platforms

	测试机 1	测试机 2	测试机 3
CPU 型号	Intel Core2 Q6600	Intel Core2 Quad	Intel Core2 Duo
CPU 主频/GHz	2.4	2.5	2.4
内存/GB	3.25	3.25	2
GPU 型号	NVIDIA GTX280	NVIDIA GF9800	NVIDIA GF8600
显存	1 GB	1 GB	256 MB

表 3 在 3 台测试机上测试本文方法的帧率

Table 3 Frame rate of application on three platforms

	测试机 1	测试机 2	测试机 3
本文算法帧率/fps	33	31	27

从表中可以看到,系统的帧率保持在 30 fps 左右,用户交互漫游的流畅度满足需求;同时随着测试机配置的降低,帧率也随之降低,说明显卡性

能的优劣对系统性能有较大的影响. 图 5 给出了一些程序截图.

尽管实验证明本文方法能够较好地进行海量三维模型组织和调度,使应用程序保持稳定的系统帧率,依然存在一些有待改进和发展的地方. 对于精细的模型网格,可以考虑做更多层次的

LOD,根据可视化需求动态生成多层次简化的模型网格,减少进入渲染通道的顶点数. 微软提供的 DirectX 11 图形 API,提供用户在渲染流水线中进行顶点镶嵌的能力,因此,进一步可以考虑将图形处理器(GPU)引入到模型简化,使用 GPU 来加速建筑物模型简化操作<sup>[18]</sup>.



图 5 程序运行效果图(帧率保持在 30 fps 左右)

Fig. 5 Screenshot of Geobeans application with frame rate of 30 fps

不仅如此,还可以对模型的纹理数据进行处理以进一步优化应用程序,可以提供多个层次不同精细程度的模型纹理和网格数据进行匹配. 当模型纹理数据中包含很多小尺寸的纹理时,会大大增加文件读取时磁盘 I/O 寻道的时间开销,这对应用程序的效率有很大的影响,在下一步工作中,我们可以考虑做纹理 atlas<sup>[19]</sup>,将小纹理合并成一张大纹理贴图,降低算法的 I/O 开销.

从目前的研究进展来看,虽然取得了一定的研究成果,也实现了实际的应用系统,但依然存在一些问题需要进一步深入研究,主要集中在以下方面.

1) 动态模型的渲染和处理:目前的海量模型交互绘制系统大多只支持静态模型,但在实际应用中,场景模型往往是动态的,因此如何有效地处理动态模型成为海量模型交互绘制的一个新的挑战.

2) 利用 GPU 的并行能力进行并行绘制:GPU 的处理能力在过去几年得到了很大的提高,如何在海量模型实时交互可视化中合理地利用 GPU 的新特性<sup>[20]</sup>,也是计算机图形学当前研究的一个热点方向.

## 5 总结和展望

数字城市是综合运用 GIS、遥感、遥测、宽带网络、多媒体及虚拟仿真技术,对城市的基础设施、功能机制进行信息自动采集、辅助决策服务的技术系统. 是对现实世界的简化和抽象.

本文提出一种基于四叉树的海量城市三维模型组织与调度的方法. 该方法根据视点将模型数据分成 3 层,不同层次有不同的分辨率,同时使用纹理批处理方式提交和渲染网格数据,能够有效地提高应用程序的渲染效率,保持稳定的帧率. 实验证明,该方法实现简单,稳定性好,具有较高的应用价值.

## 参考文献

- [1] 李德仁,朱庆,李霞飞. 数码城市:概念、技术支撑和典型应用[J]. 武汉测绘科技大学学报, 2000, 25(4): 283-288.
- [2] 熊汉江,龚建雅,朱庆. 数码城市空间数据模型与可视化研究[J]. 武汉大学学报:信息科学版, 2001, 26(5): 393-398.
- [3] Lindstrom P, Koller D, Ribarsky W, et al, Real-time, continuous level of detail rendering of height fields[C] // SIGGRAPH'96 Proceeding, Los Angles, California, 1996:

109-118.

[ 4 ] Hoppe H. Smooth view-dependent level-of-detail control and its application to terrain rendering[ C] //Proceedings of IEEE Visualization Research Triangle Park,1998;35-42.

[ 5 ] Lee S C, Nevatia R. Interactive 3D building modeling using a hierarchical representation[ C] //Higher-Level Knowledge in 3D Modeling and Motion Analysis, 2003. HLK 2003.

[ 6 ] Rau J Y, Chen L C, Tsai F, et al. Lod generation for 3d polyhedral building model[ C] //Advances in image and video technology[ C], 2006, Springer: 44-53.

[ 7 ] Li Q Q, Sun X, Yang B S, et al. Geometric structure simplification of 3D building models[ J]. ISPRS Journal of Photogrammetry and Remote Sensing,2013, 84: 100-113.

[ 8 ] Fakir S, Nooruddin, Greg T. Simplification and repair of polygonal models using volumetric techniques [ J]. IEEE Transactions on visualization and computer graphics, 2003,9 (2):191-206.

[ 9 ] 翟巍. 三维 GIS 中大规模场景数据获取,组织及调度方法的研究与实现 [ D]. 大连:大连理工大学, 2003.

[ 10 ] 刘恒飞, 刘纪平,王勇,等. 网格划分与四叉树相结合的海量建筑物数据组织与调度[ J]. 测绘通报,2010(11): 4-6.

[ 11 ] 杨卫军.海量三维城市模型的调度与场景管理[ D]. 武汉: 武汉大学,2005.

[ 12 ] Arnaud D L L, Cervelle B. 3D topological modeling and visualisation for 3D GIS [ J]. Computers & Graphics-Uk, 1999, 23(4): 469-478.

[ 13 ] Remondino F. From point cloud to surface: the model and visualization problem [ J]. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2003, Vol. XXXIV-5/W10.

[ 14 ] 刘纪平. 海量空间数据组织与管理初探[ J]. 中国图象图形学报,1998(6):500-503.

[ 15 ] 闫超德,赵学胜. GIS 空间索引方法述评[ J]. 地理与地理信息科学, 2004, 20(4):23-27.

[ 16 ] 陈永康. 3D GIS 中大数据量场景可视化研究[ D]. 北京:中国科学院遥感与数字地球所,2004.

[ 17 ] 李道远,李英成,肖金城. 大范围城市三维模型管理技术研究[ J]. 测绘科学,2011,36(5): 70-72.

[ 18 ] Fang C, Yang C J, Chen Z, et al. Parallel algorithm for viewshed analysis on a modern GPU[ J]. International Journal of Digital Earth, 2011, 4(6):471-486.

[ 19 ] 高宇,吴玲达,魏迎梅. 海量模型实时交互可视化技术综述[ J]. 中国图形图象学报,2008, 13(9):1 633-1 640.

[ 20 ] 杨崇俊,赵彦庆,王锋,等. 3 维数字地球快速缓冲区分析算法[ J]. 遥感学报,2014,18(2):353-364.