TB 级地形数据实时漫游核心算法研究

左志权，陈 媛
（武汉大学遥感信息工程学院，武汉 430079）

摘 要：TB 级数据时代已经到来，如何有效地在普通 PC 机上实现 TB 级地形数据漫游，已逐渐成为 GIS（地理信息与系统）学术界的研究热点。海量数据组织管理是 3 维地形数据漫游的核心机制，传统四叉树、八叉树等数据结构，一定程度上解决绘制瓶颈问题，但是结构复杂，很难适应 TB 级地形场景实时、稳定的漫游需求。通过实施多级缓冲机制、采用动态 LOD（levels of detail）与显存调度技术，以及引入多核 CPU 并行计算策略，最终构建出一套与数据大小无关的实时地形绘制算法。通过对 1 GB、10 GB、200 GB、500 GB 级等海量数据进行实验，平均绘制速率可达到 30 帧/s 以上，比一般算法高出 2 ～ 3 倍，从而验证本文算法的实用性。

关键词：地形可视化；动态调度；多级缓冲；CPU 多核技术；并行计算
中图法分类号：P208 **文献标志码**：A **文章编号**：1006-8961(2010)09-1411-05

Research on kernel algorithm of real-time rendering based on TB lever terrain data

ZUO Zhiquan, CHEN Yuan
(*School of Remote Sensing and Information Engineering of Wuhan University, Wuhan* 430079)

Abstract: With the availability of terrain data of the TB level, how to render terrain data of TB magnitude based on ordinarily computer efficiently, is becoming a hot topic in GIS academia . Organization and management of massive data is the core mechanism for 3D terrain rendering . The traditional data structures as quadtree, octree and so on are not effective, which cannot support real-time displaying and roaming of large-scale terrain scene. According to the mechanism of multi-level buffer, implementation dynamic dispatching of LOD, graphics memory dispatching and multi-core CPU parallel computing strategy, this paper has good results. Finally, this paper puts forward the solution, a real-time Terrain Rendering Algorithm which has no concern with data size. It is verified by experiment on massive data of 1GB, 10 GB, 200 GB, 500 GB and so on, the rendering rate can reach to 30 frame/s. It is faster than most existing algorithms. The experiment result shows the practicability of the real-time terrain rendering algorithm.

Keywords: terrain visualization; dynamic dispatching; multi-level buffer; multi-core CPU; parallel computing

0 引 言

快速、高效的大规模地形场景绘制是 3DGIS 的基础。尽管国外已具备相对成熟的商用系统如 GoogleEarth，SkyLine 3DGIS 系统等，国内也具有相关成果，但当前海量数据实时绘制效果，距满足人类视觉的实时需求，仍有一段距离。因此，海量地形数据更快速、更稳定的绘制算法仍然需要不断地探索。

到目前为止，GIS 学术界还没有公开一套可行、稳定、高效的 TB 级地形数据绘制策略，大部分都是以优化渲染管线中的部分环节而见诸于各类公开刊物。鉴于此，笔者以自主开发的一套 PC 版 3D 浏览器（surface scan）为实验平台，通过对 1 GB，10G B，

基金项目：国家自然科学基金项目（40771177）；国家高技术研究发展计划（863）基金项目（2009AA12Z126）。
收稿日期：2009-05-21；**改回日期**：2009-09-14

第一作者简介：左志权（1983— ），男。武汉大学博士研究生。主要从事摄影测量与遥感相关方面的研究。E-mail：zzquan923@126.com。

200 GB,500 GB 级等海量数据进行实时漫游实验,都取得了良好的效果,验证了本文算法的可行性。

1 基本原理

1.1 地形渲染及其瓶颈分析

3 维地形渲染可以简要分为数据读取、顶点计算与绘制、纹理映射、投影计算与裁剪,以及光栅输出等 5 个基本步骤[1],如图 1 所示。

在整个 3 维地形渲染过程中,瓶颈主要集中在前两个阶段,即数据读取需要耗费时间 50% ~ 60% 、顶点坐标绘制与纹理映射耗费时间 20% ~ 30% 。前者受系统访问磁盘速度限制,而后者则受显卡带宽、GPU 实时计算能力限制。

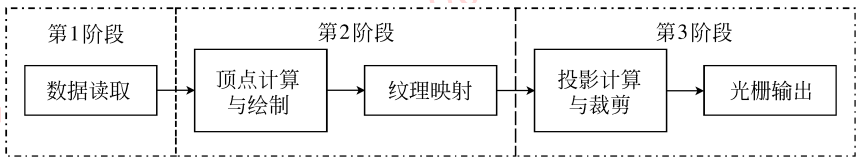

图 1 3 维地形绘制的 3 个阶段

Fig. 1 Three Stages of Terrain Rendering

1.2 传统算法比较

近年来,国内外相关学术研究也比较活跃,同时也取得一些成果。Lindstrom[2]提出了一种基于三角形融合概念的限制四叉树生成和三角化方法,Duchaineau[3]提出了基于三角形二叉剖分的 ROAM (real-time optimal adaptive mesh)算法,Hoppe[4]将其视点相关递进网格(VDPM)应用到地形中,实现了大规模地形的实时漫游,该算法首先将地形分为大块,对每块分别用递进网格预处理,记录简化过程的信息,用这些信息根据不同的视点实时生成不规则三角形网格,Ulrich 等人[5]提出了 Chucked LOD 算法将地形数据分块组织,对每一块数据都预先生成几种不同的细节层次,根据视点选择当前可见地形分块相应的细节层次模型导入内存进行渲染;国内李惠等人[6]将 ROAM 算法应用在大规模地形渲染,施松新等人[7]进行了基于分块的大规模地形实时渲染方法研究,谭兵等人[8]利用约束四叉树实现地形的实时多分辨率绘制,许妙忠[9]也进行了大规模地形实时绘制的研究。

以上绘制算法可以称为经典算法,其共同特点是从内存管理、可视区快速裁剪等两个方面进行优化。笔者通过采用 Cg、Cuda 等开发工具,对显卡性能进行测试,结果表明 GPU 访问内存延迟是渲染速度的又一大瓶颈。进一步究其原因,PC 机显卡通过 PCI-E 接口传输数据,其效率较低。本文算法与经典算法最大的区别就是,在充分分析渲染管线的基础上,通过建立内存缓冲池、显存缓冲池等多级缓冲机制,提高绘制效率。

1.3 绘制流程

通过对 3 维地形渲染流程及渲染瓶颈的分析,提出 TB 级地形数据绘制流程如下:

1)地形纹理数据预处理。包括对原始 DEM(数字高程模型)、DOM(数字正射影像图)数据进行重采样等处理。

2)地形数据组织。实现顶点与纹理数据的关联分块,将同一分辨率的高程格网点数据与纹理数据存放在相邻位置。

3)更新内存缓冲池。顾及视点相关坐标位置,实现内存地形数据的动态调度。

4)更新显存缓冲池。顾及各帧之间的关联性,实现显存地形数据的动态调度。

5)可视区域裁剪。根据数据块屏幕投影半径计算,以及 LOD 层次计算,剔除不可见地形区域。

6)渲染输出。

2 实现方法

2.1 高效的数据组织策略

目前实现大规模地形数据实时漫游都是采用规则格网数据。相比之下,非规则格网数据需要庞大的索引结构与存储空间,很难实现实时动态数据调度。针对规则格网数据,目前主要采用数据分块存储组织形式,如朱军等人[10]采用 32 × 32 大小的地形分块存储,李惠等人[6]同样也采用分块文件的组织方式。分块文件存储具有访问磁盘次数最少的优势,大大降低磁盘访问的 I/O 瓶颈。

本文也同样采用分块存储结构,但与传统方式不同的是:将对应的地形数据与纹理数据存储在相邻的位置,这样对应地形与纹理数据既可以一次读入,又可以减少一半的磁盘访问次数,使得磁盘 I/O 瓶颈得到进一步消除。本文存储结构与传统分块存储结构对比如图 2(a)、(b)所示。

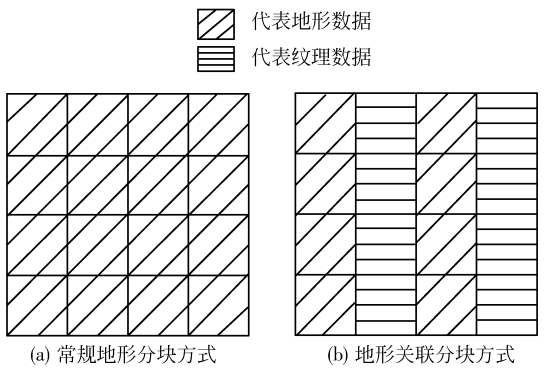

图 2　两种分块方式对比示意图

Fig. 2　Comparison of two patch methods

2.2　多级缓冲调度

传统的四叉树、八叉数索引结构在 TB 级地形数据调度中存在明显的不足。以四叉树为例,当数据量增大,四叉树层次增加,叶子节点数目也相应增多。当数据量达到 TB 级左右,将会产生庞大的索引结构,很难响应实时遍历需求。

采用相机视点相关索引模式,通过建立多级缓冲池,实现数据的动态调度。多级缓冲调度算法描述如下:

1) 根据当前相机位置,快速计算以相机为中心的四周 $n(n>0)$ 个数据块文件地址索引。

2) 采用循环链表数据结构建立动态内存缓冲池,并载入初始数据块。

3) 采用循环链表数据结构建立动态显存缓冲池。

4) 通过计算数据块屏幕投影半径,以及严格的视锥体可视性裁剪判断,计算当前需要显示的数据块索引。

5) 计算缓冲区更新阈值,更新显存缓冲池中的部分数据块。

6) 根据移动后的相机位置,计算新的 n 个数据块文件地址索引。

7) 计算缓冲区更新阈值,判断是否需要更新内存缓冲池中的无效数据块。

8) 重复 4)至 7)。

设定集合 $\boldsymbol{A}$ 为当前缓冲区的数据块集,集合 $\boldsymbol{B}$ 为更新缓冲区的数据块集,那么缓冲区更新阈值计算方法如下:

1) 统计集合 $\boldsymbol{A}$ 中的有效节点集 $\boldsymbol{a}_1,\boldsymbol{a}_2,\cdots,\boldsymbol{a}_n$,并标记到视点中心距离最远的节点 a';

2) 统计集合 $\boldsymbol{B}$ 中的有效节点集 $\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_m$,并记录各节点到视点中心的距离;

3) 统计集合 $\boldsymbol{C}=\boldsymbol{A}\cap\boldsymbol{B}$ 的节点集 $\boldsymbol{c}_1,\boldsymbol{c}_2,\cdots,\boldsymbol{c}_k$,标记集合 $\boldsymbol{B}$ 中距离视点最近的节点 b',且 $b'\notin\boldsymbol{C}$;

4) 如果 $(m-k)/n>\delta$ 时,b'替换 a'。

其中,m、n、k 均为自然数,δ 大小一般设定为 0.2 比较合适。

缓冲区更新调度示意图如图 3 所示。

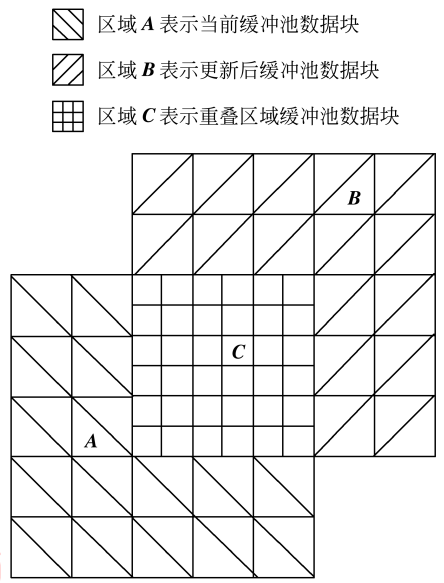

图 3　缓冲区调度示意图

Fig. 3　Explain for buffer scheduling

2.3　基于多核 CPU 的并行计算技术

在计算机硬件技术日新月异的今天,多核 CPU 技术已经非常普及。过去的单核单线程计算已无法满足大规模地形可视化计算。多核 CPU 技术可以将庞大的计算链条有效地分解,由过去串行计算模式改为流行的并行计算模式,这样有助于充分利用 CPU 的计算能力,极大地提高计算机资源的利用率,为海量地形数据的流畅绘制提供有效解决方案。

本文提出 3 线程并行协同计算模式,多线程调度管理算法简述如下:

1) 后台线程 1 负责维护内存缓冲池与数据读

取,内存缓冲池工作正常时,挂起后台线程 3;

2) 界面线程 2 负责维护显存缓冲池与绘制,显存缓冲池工作异常时,激活后台线程 3;

3) 后台线程 3 负责内存缓冲池与显存缓冲池的协同工作,其中,后台线程 3 大部分时间处于挂起状态,仅有当显存池出现丢失异常时才被激活。

某地形区域绘制期间多核 CPU 性能监视如图 4 所示。

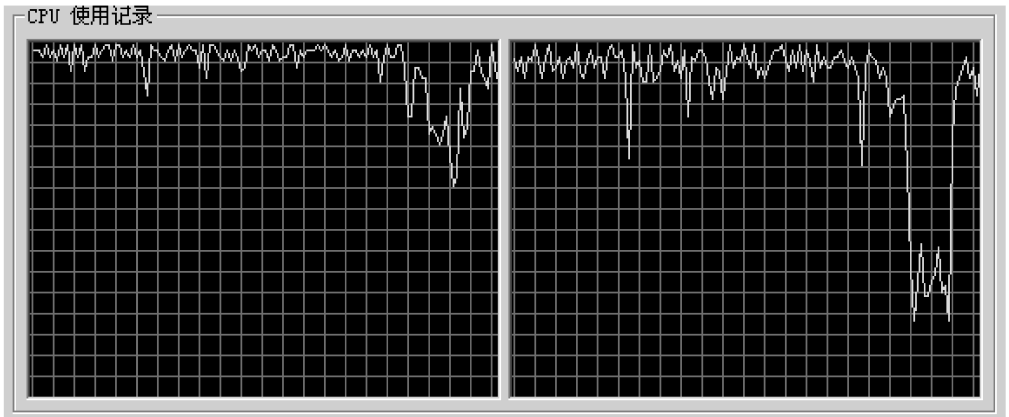

图 4 地形渲染期间多核 CPU 性能监视

Fig. 4 Muti-core CPU performance during terrain rendering

2.4 视区裁剪与动态 LOD 技术

视区裁剪是 3D 图形渲染管线中的关键步骤。其中文献[6,11]等为了提高裁剪速度,均采用基于视锥体平面投影后的三角形区域快速裁剪算法。通过实验验证,该算法有着很大的局限性,要求相机视线方向接近水平。如果采用第一人称视角实景漫游模式,相机视线为空间 360°任意朝向,此时,投影后的三角形区域与真实可见区域不完全相等,最终导致屏幕输出端出现盲区,所以,必须进行严格的视锥体投影裁剪判断。

视锥体裁剪判断是比较耗费 CPU 时间的浮点运算,在判断之前,应首先计算每一数据块在屏幕上的投影半径,以屏幕投影半径小于一个像素为阈值,剔除大量数据块。

如果视点的水平张角和投影平面的边长分别为 α 和 L,投影线段的长度为 l,视点到投影中心的长度为 d,λ 为单位长度在屏幕上的投影像素,那么屏幕投影半径为

$$\tau = \frac{l \times L \times \lambda}{2 \times \tan\frac{\alpha}{2} \times d} \tag{1}$$

LOD 技术同样也是逼真 3 维地形快速绘制流程中的必要环节。地形 LOD 技术已经非常成熟,如文献[5,7,9,10]等均有较实用的 LOD 技术描述。值得指出的是本文采用与文献[9]相近的动态 LOD 技术,并通过进一步三角形剖分实现 T 型裂缝修补。

3 实验结果

为了检测本文算法的实用性和实际效果,以便携机为主要实验平台,基本硬件配置为 Inter 2.4 G 双核 CPU,1 G 内存,NVIDIA Quadro140M 型显卡(显存 128 M,带宽 64 位),网络磁盘容量 1TB。各种数量级的模拟地形数据实验结果,如表 1 所示。

表 1 各种量级地形数据的漫游实验结果

Tab. 1 Test results of series terrain data

地形大小	数据量/GB	屏幕分辨率	平均帧速/(帧/s)	内存占用量/MB
12 000×12 000	0.94	1 400×1 050	30.7	125~135
12 000×48 000	3.75	1 400×1 050	28.4	125~135
60 000×24 000	9.38	1 400×1 050	33.5	125~135
600 000×50 000	195.6	1 400×1 050	30.4	125~135
800 000×90 000	469.38	1 400×1 500	33.4	125~135

表格中的海量地形数据均由真实数字高程模型数据以及对应数字正射影像数据拼接而成,每一个格网顶点包括 3 个字节的纹理数据(RGB 像素数据)以及 4 个字节的浮点型高程数据。通过表 1 数据不难看出:在同样的硬件条件下,当实验数据量增大时,绘制速率稳定维持在 30 帧/s 上下,内存占用量也维持在 130 MB 左右。

表 2　与部分文献中绘制效率比较

Tab. 2　Real-time rendering comparison

文献编号	最大数据量	屏幕分辨率	平均帧速/(帧/s)	主要特点
文献[6]	512M	812×512	21	屏幕分辨率较低
文献[7]	1M	—	25	实验数据较小
文献[8]	4 097×4 097	1 024×768	17	随数据增大效率下降
文献[9]	20 047×33 287	—	17~38	效率随地形简化率变化
文献[10]	8 192×8 192	—	21.28	使用 GPU 加速

4　结　论

通过对各种数据量的数据进行实验,充分证明了本文算法的高效性、实用性。具体如下:

1) 随着数据量的增大,内存占用量保持在 130M 左右,绘制速率保持在 30 帧/s 以上;

2) 据统计,目前国内权威期刊上公布的各类地形绘制算法保持在 15~30 帧/s 之间,与本文算法相比一般算法漫游速度提高 2~3 倍;

3) 本文算法充分运用多核 CPU 带来的强大计算能力,完成 TB 级数据并行调度管理;

4) 本文算法支持第一人称真实感漫游,可以满足各种视角的浏览需求;

5) 本文算法可以作为虚拟城市或虚拟地球的地形绘制模块,通过进一步研发,有望实现海量模型数据的动态管理,从而为实现实用的 3 维 GIS 系统打下理论基础。

参考文献(References)

[1] Xu Qing. Three-dimensional Terrain Visualization Technology [M]. Beijing: Sruvey and Mapping Press, 2000. [徐青. 地形三维可视化技术[M]. 北京:测绘出版社,2000.]

[2] Lindstrom P, Koller D, Ribarsky W A. Real-time, continuous level of detail rendering of height fields [C]//Proceedings of ACM SIGGRAPH'96. New York, USA: ACM, 1996:109-118.

[3] Duchaineauy Mark, Wolinsky Murray, Sigeti D E, et al. ROA Ming Terrain: Real-time Optimally Adapting Meshes [C]// Proceedings of the IEEE Visualization Conference. Phoenix, AZ, USA: IEEE Comp. Soc., 1997:81-88.

[4] Hoppe H. Smooth View-Dependent Level-of-Detail Control and its Application to Terrain Rendering [C]//Proceeding of the IEEE. Visualization Conference. Research Triangle Park, NC, USA: IEEE Comp. Soc., 1998: 35~42.

[5] Ulrich Thatcher. Rendering Massive Terrains Using Chunked Level of Detail Control [EB/OL] (2002-04-14) [2009-03-10]. http://tulrich.com/geekstuff/sig-notes.pdf.

[6] Li Hui, Zhai Lei, Lin Chengkai, et al. A real-time rendering method of large scale terrain [J]. Journal of System Simulation, 2004, 16(4):736-739[李惠,翟磊,林诚凯,等. 一种超大规模地形的实时渲染方法[J]. 系统仿真学报,2004,16(4):736-739.]

[7] Shi Songxin, Ye Xiuxin, Zhang Sanyuan, et al. Partition based real-timerendering method for large-area terrain data [J]. Journal of Zhejiang University: Engineering Science, 2007, 41(12): 2002-2006[施松新,叶修梓,张三元,等. 基于分块的大规模地形实时渲染方法[J]. 浙江大学学报:工学版,2007,41(12):2002-2006.]

[8] Tan Bing, Xu Qing, Ma Dongyang. Real-time multi-resolution terrain rendering using restricted quadtree [J]. Journal of Computer-Aided Design & Computer Graphics, 2003, 15(3): 270-276. [谭兵,徐清,马东洋. 用约束四叉树实现地形的实时多分辨率绘制[J]. 计算机辅助设计与图形学学报,2003,15(3):270-276.]

[9] Xu Miaozhong, Li Deren. A fast culling algorithm for visualization of terrain [J]. Geomatics and Information Science of Wuhan University, 2004, 29(12): 1080-1083. [许妙忠,李德仁. 地形可视化中快速视区裁剪算法研究[J]. 武汉大学学报. 信息科学版,2004,29(12):1080-1083.]

[10] Zhu Jun, Gong Jianhua, Qi Hua, et al. A simple and efficient algorithm for real-time rendering of large scale terrain [J]. Geography and Geo-Information Science, 2005, 21(2): 24-27. [朱军,龚建华,齐华,等. 大规模地形实时绘制算法[J]. 地理与地理信息科学,2005,21(2):24-27.]

[11] Luo Dai, Xie Maojin, Cao Weiqun, et al. A terrain visualization algorithm based on GPU programming [J]. Journal of Image and Graphics, 2008, 13(11): 2244-2248. [罗岱,谢茂金,曹卫群,等. 基于 GPU 编程的地形可视化[J]. 中国图象图形学报,2008,13(11):2244-2248.]