大规模地形无缝漫游技术研究 *

邓宝松[1]，于荣欢[2]，秦方钰[1]，吴玲达[2]
（1. 总后后勤科学研究所，北京 100071；2. 国防科学技术大学 信息系统工程重点实验室，长沙 410073）

摘　要：针对大规模地形动态漫游提出实现流程和算法框架，基于分层分块地形 LOD 组织存储策略完成数据预处理，绘制阶段提出视点相关的地形调度和简化算法，利用多线程处理机制进行地形块裁剪和内外存数据交换，借助 GPU 硬件实现场景加速绘制算法，并提出分块地形和纹理数据的无缝拼接策略。真实数据实验的算法比较和性能测试结果表明，该方法具有支持数据量大，绘制效率高、实用性强等特点。

关键词：地形绘制；大规模；无缝拼接；多线程处理；视点相关
中图分类号：TP391　　**文献标志码**：A　　**文章编号**：1001-3695(2012)01-0369-04
doi:10.3969/j.issn.1001-3695.2012.01.102

Research on seamless rendering of large scale terrain

DENG Bao-song[1], YU Rong-huan[2], QIN Fang-yu[1], WU Ling-da[2]
(1. *Institute of Logistics Science, General Logistics Department, Beijing* 100071, *China*; 2. *Science & Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha* 410073, *China*)

Abstract: This paper proposed a dynamic and seamless rendering scheme for large scale terrain based on LOD terrain tiles, used multi-resolution terrain for view-dependent data control and grid simplification, and employed multi-thread mechanism for visibility clipping and data exchange between memory and disk, at the same time, used an accelerating technique based on GPU for scene rendering, and proposed a seamless combination algorithm between tiles of terrain and texture. Experimental results of real scenes and comparisons with traditional method demonstrate the feasibility, efficiency and practicality of our method.

Key words: terrain rendering; large scale; seamless combination; multi-thread processing; view dependent

0　引言

随着卫星、遥感和测绘等技术的发展，海量地理环境数据获取已经相对容易，各国也不断加强地形、影像等空间数据库的基础设施建设；同时，数字地球、虚拟城市及三维战场环境等虚拟现实应用也越来越广泛。虽然近年来计算机硬件发展迅速，但永远无法满足软件的应用需求[1,2]。又由于地形具有结构复杂、数据量大等特点[3]，使得海量地形数据的显示一直是计算机图形学研究的难点和热点之一[3,4]。

大规模地形数据动态漫游就是要处理、存储、组织和管理海量地形和影像数据，在计算机屏幕上动态调度并精简目标区域的显示内容，在有限内存、计算和图形处理资源的条件下，达到高速度、高精度和真实感的可视化目的[4]，为上层应用提供基础三维地理环境平台。LOD(level of detail)技术根据一定规则来简化目标场景的几何表示，选择不同细节程度进行模型绘制，即让每一帧在不影响显示效果的前提下尽可能地减少需要绘制的三角形个数，是解决大规模地形影像数据快速绘制的有效手段[5]。

Duchaineau 等人[6]提出实时优化自适应网格(ROAM)算法，基于二叉树结构实现近似连续 LOD 地形绘制，但顶点排序和误差度量占用大量 CPU 资源。随着图形硬件的发展，Lindstrom 等人提出基于规则网格且视点相关 LOD 算法[7,8]，并采用四叉树管理地形数据，依据屏幕误差建立数据块节点的评价函数，但是递归调用消耗内存较大。Larsen 等人[2]在此基础上进行了算法改进，重点消除不同层次细节数据过渡时产生的顶点跳跃现象，这种方法占用内存较小，但误差判定粗糙。Livny 等人[3]借助 GPU 提高系统整体绘制性能，并改进了地形块的拼接方法；Zhang 等人[9]还将 GPU 硬件用于地形数据压缩等通用计算，降低了传输和管理的数据量。

本文从工程实践角度提出一种针对海量地形数据高效漫游的解决方案：首先进行实现流程和整体框架的介绍；给出地形影像的分层分块预处理、存储和组织策略；然后基于多线程技术实现视点相关的 LOD 简化、地形块调度管理和内外存数据交换，并借助图形硬件 GPU 编程提高地形绘制的效率，同时提出地形边界和影像纹理块的无缝拼接策略；最后通过真实场景实验系统验证了本文方案的可行性，并给出与已有算法的比较和主要性能指标。

1　实现流程框架

本文算法的实现流程框架如图 1 所示。首先，将目标区域的地形和影像数据作为输入，地形通常采用 DEM 规则高程采样点，而影像则需转换为 TIFF 等标准图像格式，两者通过对

收稿日期：2011-04-30；**修回日期**：2011-06-15　　**基金项目**：总后勤部优秀青年科技人才扶持对象专项基金资助项目(2010807)

作者简介：邓宝松(1978-)，男(满族)，河北遵化人，工程师，博士，主要研究方向为虚拟仿真、信息系统集成等(dbs310@163.com)；于荣欢(1983-)，男，湖南怀化人，博士研究生，主要研究方向为虚拟现实、可视化；秦方钰(1976-)，男，河南驻马店人，工程师，硕士，主要研究方向为信息系统集成；吴玲达(1962-)，女，上海人，教授，博导，博士，主要研究方向为多媒体信息系统与虚拟现实.

应点或边界点建立坐标映射关系,如果参考系不同还要经过校正和转换;然后,通过迭代抽取的方式建立地形影像的金字塔存储结构,以特定格式保存到本地文件系统中,并建立各层数据块间的索引关系,从而完成场景数据的预处理。

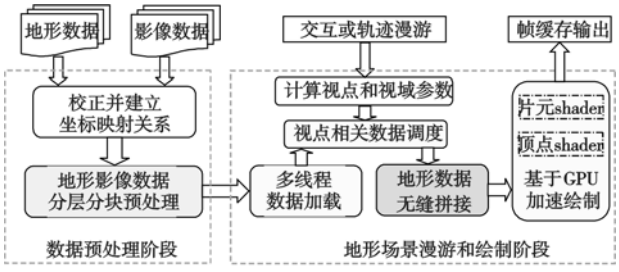


图1　本文算法实现流程框架

在场景绘制阶段,首先要在渲染主线程基础上,通过多线程技术实现内外存数据块的自动加载和场景更新遍历,此时视点和视域参数是地形绘制范围计算的依据,本文采用视点相关的加载和调度策略实现内外存数据的交换,通过链表结构实现内存数据的高效管理;其次,地形块的单独管理和绘制难免产生视觉上的裂缝或边界效果,本文分别给出网格和影像块的无缝拼接策略,增强了虚拟地形环境的真实感;最后,在传统图形绘制管线的基础上,通过 GPU 编程分担 CPU 资源瓶颈,大大提高场景显示效率,完全达到实时漫游目的。图 1 带颜色填充的环节均是本文重点研究和实现的内容,在后面章节中,分别给出图中涉及关键技术的讨论和分析。

2　分层数据处理

为提高漫游时的整体效率,地形绘制之前先要对数据进行预处理并建立索引机制[3,8],以减轻漫游时的处理负担。整个场景的地形和影像金字塔在逻辑上是完全四叉树[5],每个节点所含地形顶点数量和纹理尺寸是一样的,但物理范围和细节程度不尽相同。树根节点包含的网格是整个区域地形的最低层次细节表示,而子节点所含数据则把父节点表示的地形分成四部分,每部分都采用比父节点更精细的网格纹理表示。地形和影像的抽取直接从最精细数据开始,通过双线性插值算法迭代生成上层数据块。通过对原始地形和影像数据进行预处理,对目标场景进行分层、分块组织和存储,得到基于文件系统的四叉树存储结构,如图 2 所示。

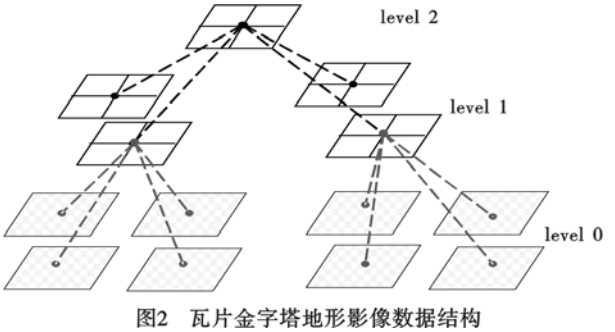


图2　瓦片金字塔地形影像数据结构

考虑到绘制效率和图形引擎自身特点,调入内存数据块宽度 N 的最佳尺寸为 2 的整幂次,并且大小要合适。太大会超出计算机瞬间处理能力,影响性能[4];太小会增加数据加载交换频度,影响效率。考虑到 Intel 的 CPU 内存页面一般设置为 4 KB,本文实现中地形数据块大小取 32×32,纹理数据块大小取 256×256,这样可在一定程度上提高磁盘存储的利用率,降低内存残页的次数,同时数据调度也不会太频繁。

3　动态地形调度与绘制

由于全部地形数据远远大于系统内存容量,因此本文采用基于外存(out-of-core)的数据存储和管理策略[8],基于视点相关的调度策略选择视野范围内的数据块,再通过多线程技术实现内外存数据的高效加载和交换。

3.1　多线程数据加载

地形绘制本身占用较大计算资源,为使外存数据读取尽量减少对绘制效率的影响,本文充分利用多核 CPU 的硬件并行处理特性[9],借助基于共享内存的多处理器编程语言——OpenMP,采用多线程技术进行系统软件实现。主线程完成整个场景地形块的绘制管理,再创建单独线程判断视野中数据块的有效性,即场景更新线程;单独线程负责地形影像块的载入,即数据装载线程。在场景漫游过程中,通过建立外存、内存及显存之间的三级缓存机制和纹理数据预取技术,在很大程度上减少了地形数据加载和纹理数据从外存到显存的传输时间,进一步增加了场景绘制的整体效率。图 3 给出本文算法实现中,内存数据管理和索引的示意图,高效的索引机制大大提高了内存管理和场景绘制的整体性能。

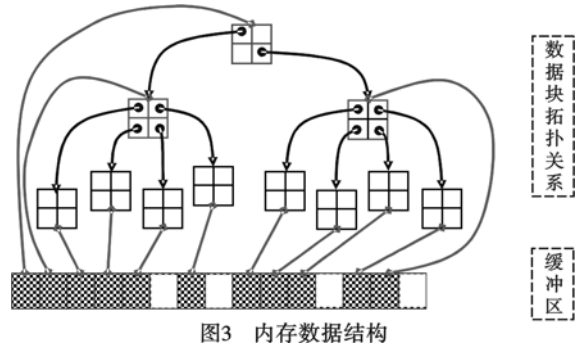


图3　内存数据结构

尽管多线程能够加大 CPU 的利用率,提高系统整体性能,但由于数据块的加载和绘制是异步进行的,算法实现中肯定出现需要层次的数据块没有载入内存的情况。此时,通过图 3 建立的数据结构递归寻找缺失块的上级节点并加入当前帧的绘制队列,同时,该上级节点后的所有子节点无论是否已经成功加载,都不会在这一帧中绘制。

此外,为避免对同一数据块的访问冲突,线程之间通过互斥量来控制[1]。为保证场景显示的平滑,避免更新加载线程执行等待过长的情况,更新和加载线程内部设有计时器,如果太长则休眠一段时间,以优先保证绘制线程的流畅性。

3.2　视点相关数据调度

有效减少每帧绘制三角形的个数,降低图形流水线模型复杂度是实现场景流畅漫游的关键[4,6]。本文采用视点相关的地形裁剪和调度算法[10],其核心思想是:根据当前视点视域参数和投影矩阵,判断地形块是否位于视锥体中,从而标记内存中需要调入和需要释放的数据区,如图 4 所示。在动态绘制过程中,随着视点移动,动态释放缓冲区中不可见的地形块并调入新的地形块,而从硬盘中读入新数据会耗用一定时间,带来视觉上的延迟现象。为了解决这个关键问题,本文实现中建立前后台缓冲区,前台缓冲区直接服务于三维场景绘制显示,后台缓冲区则对应于数据加载调度。

为降低场景绘制负担,地形块内部网格也采用 LOD 简化方法,以满足地形漫游时实时绘制的要求,而简化依据最终要

转换为屏幕误差判定条件[4],建立顶点网格的评价函数。为降低系统绘制时的负担,本文算法实现过程中顶点评价函数的判断过程是在预处理阶段完成的[8]。

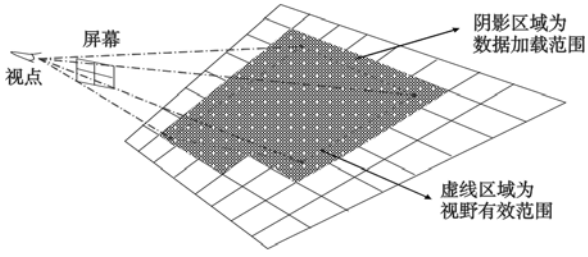


图4　视点相关的数据加载范围示意

为提高系统整体性能,还采用了地形数据块的预取策略[1],即设定时间阈值 t,设视点在 XY 平面上的移动速度为 v,绕 Z 轴旋转的速度为 u,那么由此可以计算出可能需要绘制的场景范围。数据预取在数据加载线程中实现,仅是对可能加载范围的估算,且仅在 CPU 出现空闲时运行,避免影响场景的绘制速度。

3.3　基于 GPU 的加速绘制

目前图形处理器 GPU 的发展速度远远超过 CPU,其流水线技术和并行特性使显卡逐渐减少对 CPU 的依赖,甚至部分取代原本 CPU 的工作,尤其是在三维图形场景的绘制时[1,9]。本文借助 Cg 语言通过对 GPU 的编程,传统图形管线被扩展成可编程图形管线,通过可编程顶点处理器(顶点 shader)和可编程片元处理器(片元 shader)分别对顶点数据流和片元数据流进行处理,如图 5 所示,大大分担并减轻了 CPU 的计算负担,使之能够节省更多的时间进行场景更新和内外存数据交换。

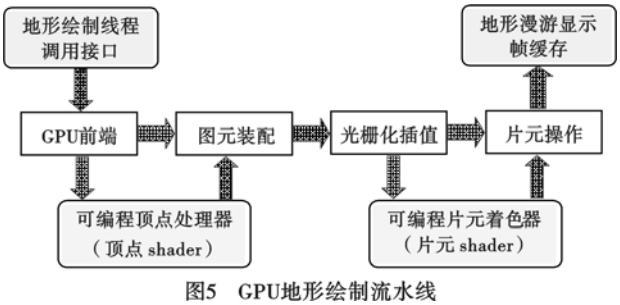


图5　GPU地形绘制流水线

分块地形和影像数据通过数据加载线程调入内存,纹理数据被调入基于 GPU 的纹理调度与绘制模块,进行基于 GPU 的数据处理变换等操作,生成纹理对象;而高程顶点数据及拓扑关系被调入基于 GPU 的三维网格模型构建模块,进行基于 GPU 的变换、网格简化等操作生成三维网格模型。用纹理对象对网格模型进行贴图,处理后的结果提交给帧缓存,这时大规模地形场景就可以在窗口中进行显示。

此外,在系统软件实现过程中,根据内存中规则网格的存储结构,本文还充分利用 OpenGL 接口对三角形条带和显示列表的支持来提高地形数据的处理和绘制速度。

4　地形数据无缝拼接

由于本文以地形块为单元控制顶点数量,顶点绘制和纹理映射均在地形块内部单独完成,因此,地形块之间的无缝拼接便成为难点问题。

4.1　地形边界缝合

由于基于视点的 LOD 可能在相邻地形数据块间选择不同的分辨率,进而引起 T 型顶点[3,4],如图 6(a)所示,导致生成拓扑网格时在地形起伏大的地方出现垂直陡峭甚至裂缝,如图 6(b)所示。虽然下节中数据块间网格裙带(skirt)会避免空洞的生成,但为了提高地形漫游的整体细节效果,本文还是提出并采用了一种新的地形边界缝合算法。

消除裂缝一般选择两种方法:a)在较低分辨率数据块中增加插值顶点,取邻接高分辨率数据块中对应点的高程,如图 6(c)所示;b)直接忽略高分辨率数据块中无用顶点,通过合并三角形的方法消除 T 型节点,如图 6(d)所示。在本文的研究和工程实践中认为,低分辨率地形边界增加网格数量不仅加大了网格绘制负担和顶点搜索复杂度,还会由于网格数量的剧烈变化影响地形的整体显示效果,因此选择后者作为地形边界缝合的处理方法。

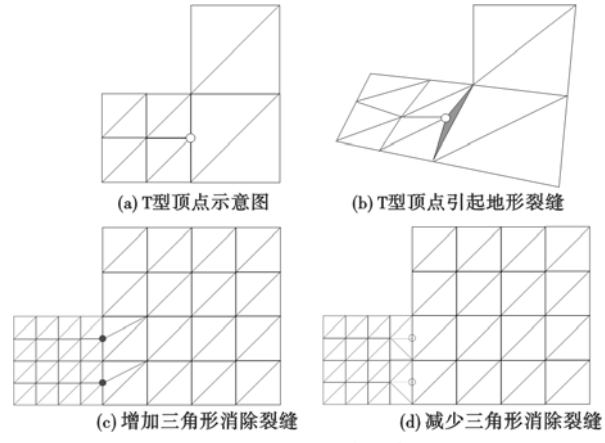


图6　地形边界缝合示意图

4.2　纹理无缝拼接

由于数据分块管理和绘制,纹理映射在每个地形块上单独完成,影像拼接便成为难点问题,而目前很多算法都没有充分考虑最终绘制效果,因此在近距离观察地形表面细节时,难免在地形块之间存在影像接缝[3]。由于 OpenGL 等图形接口均支持双线性插值方式进行纹理滤波,而影像预处理阶段也采用双线性插值算法进行纹理抽取,本文通过地形块周围生成裙带(skirt)的方式解决了纹理无缝拼接问题。

如图 7 所示,设地形数据块内的有效网格宽度为 $N=2^n(n=1,2,\cdots)$,则有效顶点数量实际为 $(N+1)\times(N+1)$,本文为其分配 $(N+3)\times(N+3)$ 组数据空间,在地形边界形成垂直向下的裙带,这样由于纹理插值导致的边界现象隐藏在裙带网格上,又因为影像数据预处理、场景绘制和纹理映射阶段均采用双线性插值的方式,使得数据块间纹理实现真正意义上的无缝拼接。图 8 给出了本文方法与传统方法在漫游视点非常接近地面时地面影像纹理显示效果的比较。图 8(a)中椭圆形区域内暴露出明显纹理边界拼接痕迹,而相同位置在(b)中则难以发现,体现了本文纹理拼接策略的优越性。

5　实验结果及分析

基于本文主要思想和算法进行系统软件的工程实现,操作系统为 Windows XP,开发环境为 Visual Studio 2005 C++,底层图形接口采用 OpenGL,GPU 开发接口采用 NVIDIA 的 Cg 语言;硬件采用普通 PC 机,主要配置为 Intel Core 2 Quad CPU 2.66 GHz,2 GB 内存,可编程图形显卡 NVIDIA GeForce 9300M GS,256 MB 显存,屏幕显示分辨率为 1024×768。

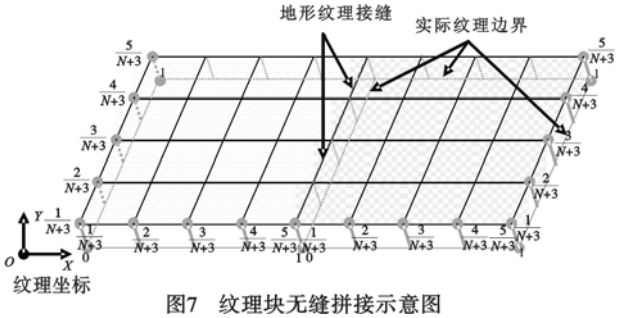
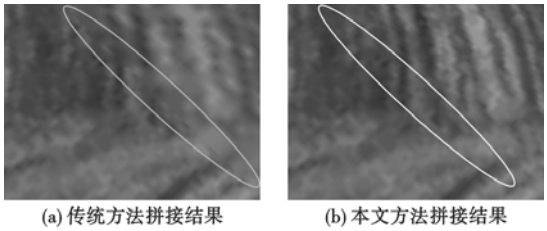


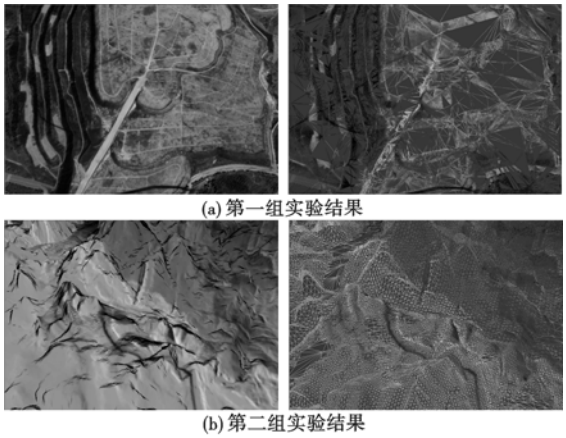
图7　纹理块无缝拼接示意图



(a) 传统方法拼接结果　　(b) 本文方法拼接结果

图8　纹理拼接效果比较

本文实验采取了两组测试数据：a) 特定应用数据，采用国内某区域 Quick Bird 0.6 m 高分辨率的卫星影像图，大小为 13108 × 14744，DEM 高程点的采样间隔约为 2 m，网格大小为 3980 × 4478，预处理金字塔结构设为 8 层，处理后数据量为 1.9 GB；b) 公开实验数据，DEM 高程数据采样点为 16385 × 16385 网格，影像分辨率为 16384 × 16384，预处理金字塔结构设为 10 层，处理后数据量约为 2.8 GB。基于文件系统存储金字塔瓦片数据并建立目录索引结构，图 9 给出基于本文实现导出的两组数据界面截图，左侧为影像纹理贴图后的效果，右侧为地形场景的网格拓扑结构。



(a) 第一组实验结果

(b) 第二组实验结果

图9　真实地形场景实验结果

基于上面给出的两组原始地形场景数据，本文方法与 Duchaineau 提供的 ROAM 算法[6] 和 Lindstrom 的 CLoD 算法[7] 进行了绘制效率的比较。这两个算法是基于规则和非规则网格中的典型代表。ROAM 算法是支持非规则网格连续 LOD 的经典算法，其核心思想在地形漫游中应用非常广泛；而 CLoD 算法则是基于规则高程网格地形简化和绘制中的常用手段。

为了有针对性地进行性能比较，三个算法在实现过程中采用同样的输入数据、统一的参考坐标，编辑生成并设置相同的漫游路径，视点变化时设置相同的步长和速度。绘制同时，间隔 1 s 采样系统绘制的即时平均帧率作为算法性能指标的主要衡量参数。

表 1 给出了两种方法的帧率变化比较。本文方法针对第一组场景平均帧率为 46.75 fps，第二组场景平均帧率为 44.57 fps；ROAM 算法针对第一组场景平均帧率为为 41.14 fps，第二组场景平均帧率为 39.44 fps；而 CLoD 算法第一组场景平均帧率为 40.61 fps，第二组场景平均帧率 41.50 fps。从中可以分析出，由于在 ROAM 算法中顶点队列处理和网格简化操作本身占用大量 CPU 时间，影响了系统的整体性能；CLoD 算法不仅内存资源占用较大，误差度量也挤占了处理资源；而本文算法不仅合理利用 CPU 和内存资源，还借助 GPU 提高图形绘制能力，表现出了较好的整体性能。此外，表中还对帧率的方差进行了比较，方差大小反映变化强度，能够体现算法在视点变化，内外存大量瞬间交换数据时的稳定性，从中也可看出本文算法的良好表现。

表 1　三种算法绘制的性能参数比较

算法	第一组实验		第二组实验	
	平均帧率/fps	方差	平均帧率/fps	方差
ROAM	41.14	42.31	39.44	48.09
CLoD	40.61	38.94	38.50	39.83
本文算法	46.75	37.81	44.57	35.05

从实验结果和算法比较中可以看出，本文提出的大规模地形绘制方法不仅表现出良好的漫游效果，而且具有很强的实时性。整个场景绘制过程中，帧率基本维持在 50 fps 左右，视点整体切换时由于瞬间更新数据量大而导致间歇性帧率降低，但也均保持在 30 fps 之上。此外，虽然两组实验的数据量有较大区别，但是在平均帧率上差别不大，这也说明了本文方法与数据量的无关性。

6　结束语

大规模地形漫游是很多虚拟现实和三维地理信息应用中的基础性关键技术，具有广阔的应用前景。本文面向大规模地形无缝漫游提出高效解决方案，包括分层分块数据的预处理和索引机制、基于多线程技术的数据加载和内外存调度管理、基于 GPU 编程的地形加速绘制技术，还提出针对分块地形的无缝拼接策略，最后进行算法实现，真实图像的实验结果以及与典型算法的比较，表明完全满足实时性要求。

此外，本文方法中所有地形块的 LOD 层次与相邻子块没有关系，独立性强，为实现网络条件下大规模地形分布式实时漫游具有很好的可移植性，完全可以在 Web 环境下实现。

下一步工作主要包括以下几个方面：a) 优化预处理后数据的外部存储结构，建立高效文件索引机制，进一步提高运行时数据块的搜索和加载速度；b) 跟踪 GPU 图形硬件的新增和扩展功能，继续挖掘 GPU 的图形处理潜力，借助图形硬件减轻 CPU 瓶颈负担，提高整体性能；c) 在现有地形漫游绘制平台基础上叠加矢量和模型数据，集成信息查询和空间分析功能，开展数字化三维战场环境等上层应用。

参考文献：

[1] 高宇，邓宝松，李苏军，等. Quickwalk：一个大规模场景交互漫游系统[J]. 小型微型计算机系统，2008，29(1)：175-179.

[2] LARSEN B D，CHRISTENSEN N J. Real-time terrain rendering using smooth hardware optimized level of detail [J]. Journal of WSCG，2003，11(2)：282-289.

[3] LIVNY Y，KOGAN Z，ELSANA J. Seamless patches for GPU-based terrain rendering[J]. Visual Computer，2009，25(3)：197-208.

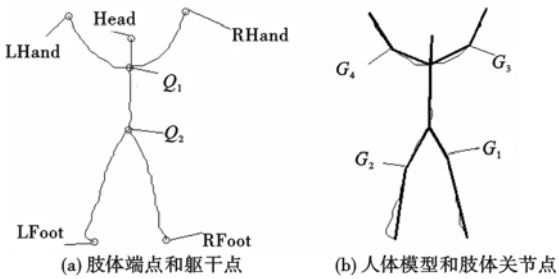
[4] RENATO P，ENRICO G. Survey of semi-regular multiresolution models for interactive terrain rendering[J]. Visual Computer，2007，23(8)：583-605.

（下转第 385 页）

3　实验结果分析

本文基于阴影的亮度特征，采取 Otsu 算法能较准确地去除阴影；在人体建模中，设计 RL 和 LR 两种遍历结构和遍历次序，采取链码方式遍历骨架，获取躯干端点；依肢体范围内骨架像素到肢体直线的最大距离来确定肢体关节点，由此得到的人体骨架模型能准确定位各个肢体的位置信息。

由本文提出的算法得到的人体骨架模型如图 7 所示。



(a) 肢体端点和躯干点　　(b) 人体模型和肢体关节点

图7　标志人体肢体端点和人体模型

图 7(a)表示由对人体骨架链码遍历得到的特征点标注，如躯干端点、四肢和头部的端点；(b)表示利用关节点定位算法得到的四肢关节点的标注，连接关节点与各自端点构成人体模型的表达。对人体四肢部分各个骨架点到对应肢体直线上的距离进行统计如图 8 所示。

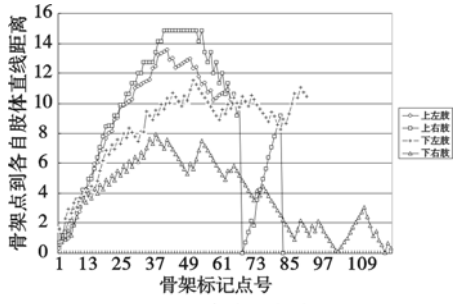
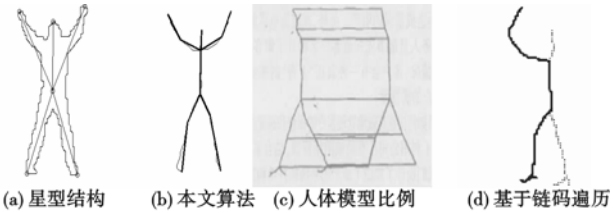


图8　四肢关节点分析

图 8 中横轴信息表示人体四肢区域内骨架点的个数，纵轴信息表示各骨架点到各自肢体直线的距离，各条曲线最大值处的骨架点表示各个关节点。

图 9 展示了各人体建模方法的结果，本文方法较其他建模方法明显准确度更高，但运行时间比图 9(a)稍高，低于(c)、(d)两种方法。



(a) 星型结构　(b) 本文算法　(c) 人体模型比例　(d) 基于链码遍历

图9　人体结构建模的实验结果对比

4　结束语

综上所述，本文基于阴影亮度特征采用 Otsu 算法实现了阴影的自适应分割，有效地去除了阴影；提出的新连通性结构标准和肢体关节点定位算法，能有效减少计算的复杂度，并准确获取人体躯干分支点和之后肢体关节点。与其他人体建模[3,4,15]相比在定位准确性和适用通用性上有所提高，且耗时更少。基于本文提出的模型，可以对人体行为作进一步的解读，如人群中危险行为识别、动作解释、人体行为理解等。

参考文献：

[1] 黎洪松，李达. 人体运动分析研究的若干新进展[J]. 模式识别与人工智能，2009，22(1)：70-78.

[2] 周进，刘冀伟，张雷. 人体骨架模型的建立[J]. 微计算机信息，2006，22(19)：226-228.

[3] FUJIYOSHI H, LIPTON A J. Real-time human motion analysis by image skeletonization[J]. IEICE Trans on Inf Syst, 2004, 87(1): 113-120.

[4] 贾程程，许相莉，周春光，等. 基于链码的人体骨架建模[J]. 吉林大学学报：理学版，2010，48(4)：641-645.

[5] 李宁，黄山，张先震，等. 基于背景差分的人体运动检测[J]. 微计算机信息，2009，25(21)：256-257.

[6] STAUFFER C, GRIMSON. Adaptive background mixture models for real-time tracking[C]//Proc of IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 1999：246-252.

[7] 邬大鹏，程卫平，于盛林. 基于帧间差分和运动估计的 Camshift 目标跟踪算法[J]. 光电工程，2010，37(1)：55-62.

[8] SHAFIE A A, HAFIZ F, ALI M H. Motion detection techniques using optical flow[R]. [S. l.]: World Academy of Science, Engineering and Technoloy, 2009：559-551.

[9] 熊亮，刘伟铭. 基于背景 Codebook 模型的前景检测算法[J]. 科学技术与工程，2010，9(10)：2118-2122.

[10] OTSU N. A threshold selection method form gray-level histograms [J]. IEEE Trans on Systems, Man, and Cybernetics In Systems, Man and Cybernetics, 1979, 9(1): 62-66.

[11] XIAO Mei, HAN Chong-zhao, ZHANG Lei. Moving shadow detection and removal for traffic sequences[J]. International Journal of Automation and Computing, 2007, 4(1): 38-46.

[12] ZHANG T Y, SUEN C Y. A fast parallel algorithm for thinning digital patterns[J]. Communications of the ACM, 1984, 27(3): 236-239.

[13] 李玉海，高敬惠，姜子敬. 基于形态学的视频序列人体骨架提取[J]. 微计算机信息，2007，23(23)：246-248.

[14] THEODORIDIS S, KOUTROUMBAS K. Pattern recognition [M]. 2nd ed. Beijing: China Machine Press, 2003：298-303.

[15] 毛以芳，黄襄念. 室内复杂背景下人体骨架的提取[J]. 西华大学学报：自然科学版，2011，27(5)：50-53.

(上接第 372 页)

[5] LIU X, ROKNE J G, GAVRILOVA M L. A novel terrain rendering algorithm based on quasi Delaunay triangulation[J]. Visual Computer, 2010, 26(6): 1-10.

[6] DUCHAINEAU M, MILLER M C, WOLINSKY M, *et al*. ROAMing terrain: real-time optimally adapting meshes[C]//Proc of IEEE Visualization Conference. 1997: 81-88.

[7] LINDSTROM P, KOLLER D, RIBARSKY W, *et al*. Real-time, continuous level of detail rendering of height fields[C]//Proc of ACM SIGGRAPH Conference on Computer Graphics. 1996: 109-117.

[8] LINDSTROM P. Out-of-core construction and visualization of multiresolution surfaces[C]//Proc of ACM SIGGRAPH Symposium on Interactive 3D Graphics. 2003: 93-102.

[9] ZHANG Y, HUANG Q, HAN J. Real-time rendering of large-scale terrain based on GPU[C]//Proc of the 4th IEEE Conference on Industrial Electronics and Applications. 2009: 3795-3799.

[10] NIE J, GUO D, WANG Y, *et al*. Multilevel tile load map on massive terrain visualization [J]. Journal of Computational Information Systems, 2011, 7(2): 452-461.