基于 GPU 编程的地形可视化

罗 岱 谢茂金 曹卫群 黄心渊
(北京林业大学信息学院,北京 100083)

摘 要 由于地形模型固有的复杂性,致使计算机硬件水平一直难以满足大规模地形模型的实时显示需求。为了在现有的硬件水平上实现地形模型的快速绘制,在对传统的 ROAM 算法进行改进的基础上,提出一种基于 GPU 编程的地形可视化算法,实现了视点依赖的大规模地形的快速可视化。该算法首先基于改进的 ROAM (real-time optimally adaptive meshes)算法生成视点依赖的优化连续 LOD 模型;然后用 GPU 编程计算顶点的变换、法向量、纹理坐标、纹理采样和面元光照;最后完成地形的着色。实验结果表明,利用 GPU 编程不仅能有效提高算法速度,而且能实现较大规模地形的实时漫游。

关键词 GPU 编程 地形可视化 ROAM
中图法分类号:TP391.4 **文献标识码**:A **文章编号**:1006-8961(2008)11-2244-06

A Terrain Visualization Algorithm Based on GPU Programming

LUO Dai, XIE Mao-jin, CAO Wei-qun, HUANG Xin-yuan
(*School of Information Science and Technology, Beijing Forestry University, Beijing* 100083)

Abstract Due to the inherent complexity of terrain models, the development speed of computer hardware is far from satisfaction for real-time rendering of large scale terrain model. To resolve this problem, we enhance the traditional ROAM algorithm and therefore realize view-dependent real-time rendering of large scale terrain model with GPU programming. Here, the enhanced ROAM (real-time optimally adaptive meshes) algorithm is employed to create view-dependent continuous LOD models, and the GPU programming is employed to calculate the vertices' transform, normal vector, texture coordinate, texture sampling and fragment lighting and accomplish the rendering for the terrain in the end. Experimental results show that, GPU programming is an effective way to improve the algorithm performance and to achieve real-time roaming on relatively large scale terrain.

Keywords GPU programming, terrain visualization, real-time optimally adaptive meshes (ROAM)

1 引 言

随着雷达、卫星、遥感等相关技术的发展,海量地形数据的获取已经变得相对容易,而且地形可视化的应用也越来越广泛,比如 3 维游戏、虚拟现实、GIS、计算机仿真等。然而由于硬件发展水平的限制和地形模型几何复杂度高的特征,致使海量地形数据的显示依然是当今计算机图形学的难题之一。

为了在现有的硬件水平上,实现地形模型的快速绘制,自 1976 年 Clark 提出由模型简化来构造多分辨率模型[1] 的思想以来,各类 LOD (levels of detail)模型的建立已成为提高地形绘制速度的重要方法之一,而硬件上则主要依赖于图形卡厂商显卡性能的提高。此外,该领域也有一些 CPU (center processing unit)多媒体指令和并行计算方法的应用研究。近年来,可编程图形处理单元 (graphic

基金项目:国家自然科学基金项目(60703006);国家高技术研究发展计划(863)项目(2006AA10Z232)
收稿日期:2007-03-06;**改回日期**:2007-07-13

第一作者简介:罗 岱(1974~),男。讲师。现为北京林业大学林业装备工程专业博士研究生。主要研究方向为数字林业、多媒体技术、计算机仿真。E-mail:edl7878@hotmail.com

processing unit, GPU)的出现,给复杂图形的快速绘制注入了新的活力,而高级 GPU 编程语言的出现则使基于 GPU 编程的应用开始普及起来。

相继提出的各类静态 LOD 算法[2~6]通常是先预处理好一系列不同层次细节的模型,然后在实时显示时,则根据具体的视觉要求选择合适的细节层次。这类方法能满足一些特定的应用,而对于地形数据来说,由于其本身通常就是大文件,同时静态 LOD 算法生成的多个模型会增加存储的负担,所以容易造成存储瓶颈。此外,基于静态的 LOD 模型建立的地形场景,当不同层次的模型切换时,由于容易产生视觉跳跃现象,因此需要合适的几何平滑过渡。

连续 LOD 模型(CLOD)则根据模型在世界坐标系的几何误差阈值和在屏幕坐标系的投影误差阈值,用依赖于视点的细分方法来生成连续分辨率模型。基于规则三角形网的连续四叉树 LOD 模型[7]首先由 Linstrom 等人提出。一年后 Duchaineauy 等人又提出了实时优化适应性网格(real-time optimally adaptive meshes,ROAM)[8]。该算法是目前应用最广泛的算法之一。与静态 LOD 模型不同,由于 CLOD 不需预先处理生成多个不同层次细节的模型,而是根据视点的变化动态来生成适当分辨率的模型,因此比静态 LOD 模型节省存储空间。这类算法通常与广泛采用的规则格网地形数据相适应。

对于 CLOD[9],目前生成多分辨率地形的模型简化判断依据主要是屏幕投影误差或其变相表达——地形粗糙度。屏幕投影误差[10]是根据透视投影的原理来衡量误差。杜金莲等人从能量传播的角度出发,提出了一个适合对距离视点较远的区域进行简化的残余能量准则[9],同时将其与屏幕投影误差结合,建立了一个基于视觉原理的多分辨率地形生成准则[9]。

另一方面,GPU 自 1999 年首先由 NVidia 公司提出来后,其发展速度是 CPU 更迭速度的 3 倍多。文献[11]对 GPU 在近年来的发展和应用进行了比较全面的介绍。

Losasso 等人提出在几何裁剪图上镶嵌规则格网的地形 LOD 算法[12]。该算法提出了一套适合 GPU 流水线的 LOD 框架,其能够充分利用 GPU 强大的图形处理能力,并在提高算法速度方面取得了很大的成功,但是其压缩和误差控制尚不尽人意。

Jens Schneider 对几何裁剪图进行了改进[13],即通过对包含几何变形的 CLOD 表示和图片纹理的支持进行高质量的地形绘制,以保证在用户定义的屏幕坐标和世界坐标误差阈值内进行地形模型的细分。该算法还通过对最高分辨率模型进行优化的几何过滤避免了走样。为了获得高带宽效率,该算法在运行时,先对具有离散分辨率的简化网格的集合进行逐步传输,然后在 GPU Shader 中对这些离散的网格模型进行插值和绘制,并生成了连续分辨率的模型。

Clasen 等人将基于 GPU 的几何裁剪图应用于球形地形数据的可视化[14],其是通过映射纹理坐标,并基于顶点偏移和视点参数来计算球形地形数据高度图的采样位置。

本文尝试将传统的 ROAM 算法和 GPU 编程结合起来对地形进行可视化。

2　算法思想

本文在改进的 ROAM 算法的基础上,利用 GPU 编程来提高地形可视化算法的效率。为了对算法进行描述,首先给出以下几个定义:

定义 1　期望三角面片数:根据计算机的绘制能力预先设置的每帧期望得到的三角面片个数,比如 20 000。

定义 2　面片数帧差:前一帧绘制的三角面片个数与其期望的三角面片数的差值。

定义 3　帧阈值:当前帧的误差阈值。该值可根据面片数帧差进行动态调整。

算法在对地形数据集根据帧阈值进行非均匀采样时,首先实时构造 ROAM 网格,然后用 GPU 进行绘制输出。

3　算法的实现

3.1　地形数据的载入

由于地形数据集的数据量通常比较大,为了解决存储问题,地形数据采用了 Windows 操作系统的内存映射机制。内存映射机制是 Windows 操作系统为大文件的读写和共享所设计的解决方案。它采用高速页面缓冲的形式,最大可以支持 4GB 的文件。内存映射不仅操作简单,且效率也比较高。

3.2　帧阈值的动态调整

将数据载入后,算法即根据面片数的帧差来动态调整帧阈值。如果面片数的帧差不为 0 时,即调

整帧阈值,那么当动画停下来一段时间后,由于帧阈值的调整,使得面片数的帧差会接近 0,这样帧阈值会在其平衡点附近进行小范围的浮动,从而出现小部分网格闪烁的现象。为了防止这种闪烁现象,算法采用了滞后处理的方法,即只有某帧得到的三角面片数与期望的三角面片数的差值超过某个合适范围 R 的时候才进行帧阈值的调整,并将调整的速度乘上一个系数 S,以控制帧阈值的适应速度,以便使得每帧得到的三角面片的数量相对稳定。R 和 S 这两个常数属于经验数值,需要根据多次实验方可获得比较理想的值。一般来说,R 不宜过大或过小(本实验中取 500 比较合适),由于若过大,则导致帧速率调整滞后严重,从而出现帧速率下降或者三角面片数过少的问题,而过小则还会出现网格闪烁。S 的取值与 R 正相关,如果过大,则画面有可能会出现轻微的跳跃,而过小则增加调整的时间。本实验 S 取值 50,视觉上几乎感觉不到画面的跳跃。实验表明,若 R 和 S 选取合适,则可以得到流畅的画面。

3.3 地形分块

1998 年,Hoppe 等人将视点依赖的渐进网格(view dependent progressive meshes, VDPM)应用到地形绘制中,实现了大规模地形的实时漫游[5]。其分块的方法对于大规模地形的绘制有着重要的指导意义。地形分块有以下优点:①有利于提高裁剪效率;②有利于地形数据读取;③有利于纹理映射和并行绘制;④有利于提高算法的适应性,以适应非正方形地形的绘制。

为了避免块与块之间出现拼接缝,本文采用了边缘重叠的方法。比如每个块的数据是 64 × 64,则实际处理的数据是 65 × 65。这样,每块四周(边缘块的外侧除外)都有一排宽度为 1 的数据相重叠。分块与重叠如图 1 所示。

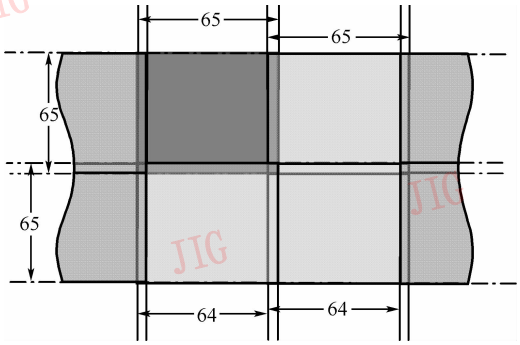


图 1 地形分块

Fig. 1 Terrain blocking

3.4 视锥预裁剪

为了初步减少算法所处理的数据量,本文采用了视锥预裁剪的方法。视锥预裁剪是在地形分块的基础上进行的。为了提高算法运行效率,考虑到地形漫游动画的特殊性,其视域裁剪可以简化成只进行相对于视锥的左右两个侧面和远裁剪面在水平面上的三角形投影的裁剪,首先产生一个较粗的裁剪结果,之后将各个块是否可见进行标记。视锥预裁剪如图 2 所示,其中浅色部分为可见区域。

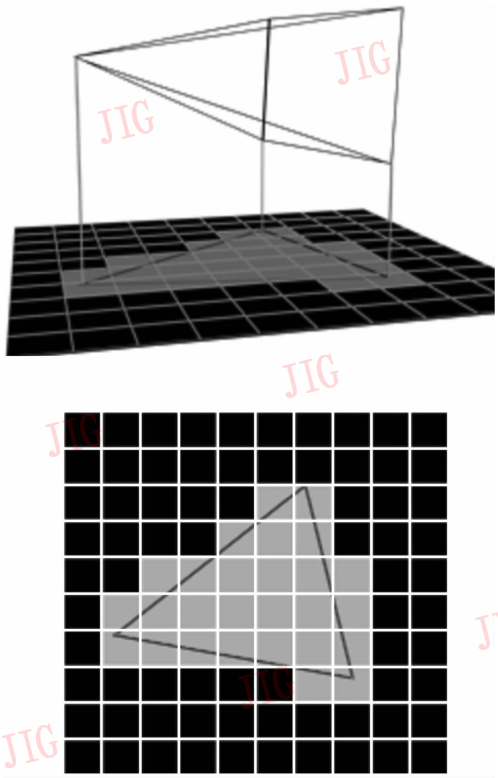


图 2 视锥预裁剪

Fig. 2 Frustum pre_culling

3.5 基于 GPU 编程实现地形模型的绘制

(1)顶点法向量计算

大家知道,对物体表面进行光照计算,需要每个顶点的法向量信息。

求取法向量的传统做法是预先计算好每个顶点的法向量存储成法向图,使用的时候直接对法向图进行采样。但是如果地形数据本身比较大,比如 8 192 × 8 192 大小的地形,其顶点数是 64M 个,如果用法向图的方式,用 4Byte 表示一个浮点数,则法向图的大小达到 64 × 3 × 4 = 768M Byte。这种存储瓶颈将会显著限制可视化地形的

大小。

另一种做法是在 CPU 上实时计算顶点的法向量。但是向量的叉乘运算和归一化要涉及多次加减乘除和开平方运算,也需要消耗多个 CPU 时钟周期。如果一帧需计算 1 万个三角面片,每个顶点用其周围 4 个顶点围成的 4 个三角面片的法向量的平均值来近似,则需要 12 万次这样的计算,因此容易造成计算瓶颈。

由于本文将顶点法向的计算过程转移到 GPU 中实时进行,充分利用了 GPU 强大的浮点向量计算能力,并利用了 Cg 语言提供的叉乘函数、向量标准化函数和线性插值函数,因此能够非常方便地完成法向量的计算和标准化工作。

如果把地形的高度图当作一张明度纹理,并用 Cg 语言中的 texRECT 函数对高度图进行采样,那么即可得到顶点周围 4 个顶点的高度值,据此即可计算顶点的法向量。实验表明,这种方式比 CPU 直接对内存进行访问的方式在性能上有较大程度的提高,在作者所进行的测试实验中约可快 15fps。而且在帧速率提高的同时,帧速率变得更加平稳。

(2)地面光照计算

本文采用了 Phong 模型计算地面的光照。由于可编程 GPU 具有可根据需要灵活控制图形生成的特性,且算法基于地形表面一般不存在镜面反射的考虑,因此在利用 GPU 生成图形的计算中就去除了其镜面反射计算的部分,以加速地形场景的绘制。

4　实验结果

算法的测试采用了 8 192 ×8 192 大小的地形数据。PC 平台关键硬件配置如下:

CPU:Intel Pentium4 2.8G,内存:Kingston 1G DDRII,显卡:NVidia 7800GT,显示器:Philips 107B5。操作系统为 Microsoft Windows XP Pro. Version 2002 Service Pack2。Cg 编译器版本为 1.50。开发环境为 Microsoft Visual C + + 6.0 sp5。

测试结果:图 3 显示了一组对比实验的数据曲线。实验都使用了相同的 ROAM 算法,其中帧速率数据为一次漫游过程中,中间连续 30 秒的帧速率数据(测试编号中的“A”表示采用 GPU 编程加速,“B”表示未采用 GPU 编程加速,数字表示地形网格的大小,比如“A1 024”表示“采用 GPU 编程加速,地形网格大小为 1 024 × 1 024”。)由图 3 可见,A 组的 5 条曲线都在 B 组的 5 条曲线上方。实际上,从表 1 统计的平均帧速率可以看出,A 组的平均帧速率大约是 B 组平均帧速率的 4 倍。

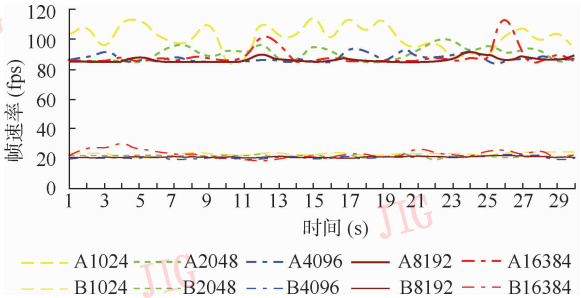


图 3　帧速率分析

Fig. 3　Frame rate analysis

表 1　测试结果统计分析

Tab. 1　Statistic analysis of test result

测试编号	平均帧速率(fps)	方差
A1024	101.30	63.20
A2048	90.34	18.53
A4096	87.74	5.73
A8192	86.86	2.61
A16384	88.07	33.36
B1024	22.71	0.89
B2048	21.43	0.46
B4096	20.66	0.81
B8192	21.28	0.12
B16384	22.89	6.90

表 1 统计了各组测试的平均帧速率和帧频方差数据,图 4 则展示了各组测试的平均帧速率和帧频方差的变化曲线。该统计结果表明:①平均帧速率不会随地形数据的增大而急剧下降;②整体上看,ROAM 算法帧频方差较小,帧速率稳定,这说明 ROAM 算法适合于大规模地形的实时绘制;③帧频方差与地形网格的大小相关,不论是 A 组或者是 B 组,过大或者过小的地形都显示出较大的方差,而 8 192 ×8 192 大小的地形的方差最小;④A 组的帧频方差明显高于 B 组,这可能是由于 GPU 编程加速放大了方差的缘故。

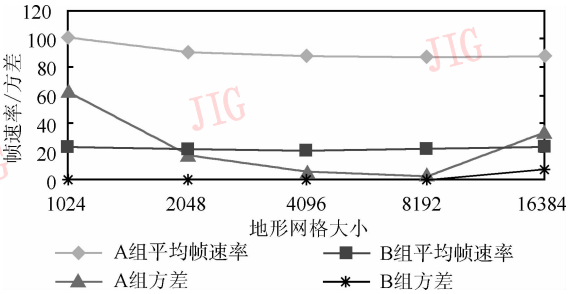
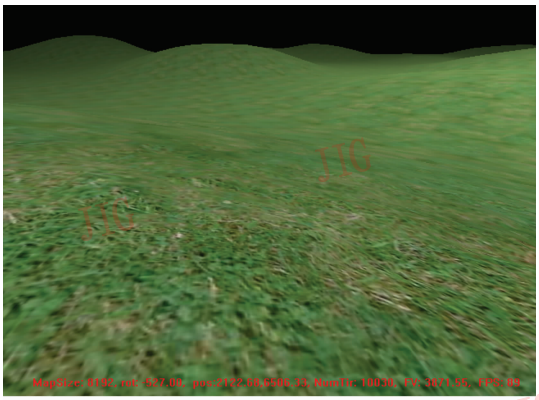


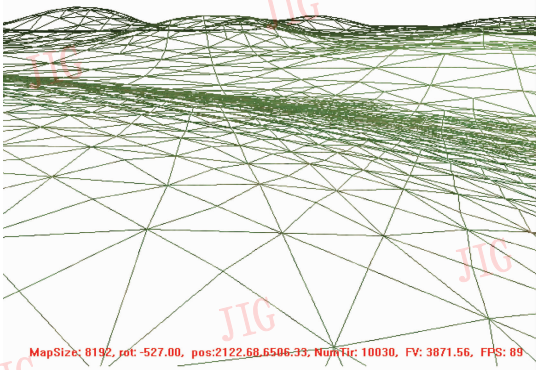
图 4 帧速率统计分析

Fig. 4 Frame rate statistic analysis

实验结果表明,该算法所得漫游画面平稳,视觉效果良好。运行屏幕截图如图 5(a)所示。



(a) 运行效果截图



(b) LOD网格

图 5 测试结果

Fig. 5 Testing result

5 结 论

本文描述了一种在改进的 ROAM 算法基础上结合 GPU 编程的地形可视化方法。该方法首先用改进 ROAM 算法结合分块和视锥预裁剪等技巧实时生成视域内物体的连续 LOD 模型;然后用 GPU 编程实时计算物体表面的顶点法向以及光照、纹理采样,进而实现物体表面的着色。实验表明,该算法不仅节省存储空间,而且具有较高的效率,能够在高帧速率下实现大规模地形数据的可视化,且帧速率平稳、视觉效果较好,有能力对更大规模的地形实现实时可视化,具有一定的应用价值。

由于没有采取反走样的相关措施,致使图像出现了边缘锯齿状等走样现象。为了进一步提高图像质量,寻找一种合适的反走样方法是必要的。由于地形采用分块纹理,致使距离视点较远的纹理重复比较明显,因此对顶点纹理坐标进行合适的扰动,使生成的地形场景更为自然真实,也是下一步要开展的工作。

通过测试发现,CPU 的负担依然比较重,这是由于地形表面细分时进行的平方根运算比较费时的缘故,此外,CPU 对映射文件高度图数据的频繁访问,也是很重要的原因。对该部分进行优化或者进行转移计算,方可进一步提高算法的性能[16]。

GPU 技术日新月异,市场上不断有高端的 GPU 推出。目前,三角面片的绘制速度已经达到 100M 个/s[11],其顶点处理速度也不断增加,同时 GPU 还具备很强的并行计算能力[15]。因此充分利用 GPU 强大的功能来对地形数据进行绘制、压缩,对地形高度图进行纹理分块,是进一步增加可实时绘制的地形数据规模、提高纹理和地形采样效率的一个改进方向。

参考文献(References)

1 Clark J H. Hierarchical geometric models for visible surface algorithm [J]. Communications of the ACM, 1976, **19**(10): 547 ~ 554.

2 Schroder W, Zarge J, Lorensen W. Decimation of triangle meshes [J]. Computer Graphics, 1992, **26**(2): 65 ~ 70.

3 Hoppe H, De Rose T, Duchamp T, *et al*. Mesh optimization [A]. In: Proceedings of the SIGGRAPH [C], Anaheim, California, USA, 1993: 19 ~ 26.

4 Hamann B. A data reduction scheme for triangulated surface [J]. Computer Aided Geometry Design, 1994, **11**(2): 197 ~ 214.

5 Hoppe H. Progressive meshes [A]. In: Proceedings of the SIGGRAPH [C]. New Orleans, Louisiana, USA, 1996: 99 ~ 108.

6 Garland M, Heckbert P S. Surface simplification using quadric error metric [A]. In: Proceedings of the SIGGRAPH [C]. Los Angeles, California, USA, 1997: 209 ~ 216.

7 Perter Linstrom, David Koller, William Ribarsky, *et al*. Real-time,

continuous level of detail rendering of height fields [A]. In: Proceedings of SIGGRAPH [C], New Orleans, Louisiana, USA, 1996: 109 ~ 118.

8 Duchaineau M, Wolinsky M, Sigeti D E, *et al*. ROAMing terrian: Real-time optimally adaptive meshes [A]. In: Proccedings of IEEE Conference on Visualization [C], Phoenix, Arizona, USA, 1997: 81 ~ 88.

9 Du Jin-lian, Du wei, Chi Zhong-xian. Vision theory based multi-solution terrain generation principles [J]. Journal of Image and Graphics, 2003, **8**(11): 1295 ~ 1298. [杜金莲,杜薇,迟忠先. 基于视觉原理的多分辨率地形生成准则[J]. 中国图象图形学报, 2003, **8**(11): 1295 ~ 1298.]

10 Tang Ze-sheng. 3D Data Field Visualization [M]. Beijing: Tsinghua University Press, 1999. [唐泽圣. 三维数据场可视化[M]. 北京:清华大学出版社, 1999.]

11 Wu En-hua. The present technique situation and challenge of the GPU used as a GPGPU [J]. Journal of Software, 2004, **15**(10): 1493 ~ 1504. [吴恩华. 图形处理器用于通用计算的技术现状及其挑战[J]. 软件学报, 2004, **15**(10): 1493 ~ 1504.]

12 Losasso F, Hoppe H. Geometry clipmaps: Terrain Rendering Using Nested Regular Grids [A]. In: Proceedings of ACM SIGGRAPH [C]. Los Angeles, California, USA, 2004: 769 ~ 776.

13 Jens Schneider, Rudiger Westermann. GPU-friendly high-quality terrain rendering [J]. Journal of WSCG, 2006, **14**(1): 49 ~ 56.

14 Clasen Malte, Hege Hans-Christian. Terrain rendering using spherical clipmaps [A], In: Proccedings of IEEE Conference on Visualization [C], Lisbon, Portugal, 2006: 91 ~ 98.

15 Fernado Randima, Kilgard Mark J. The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics [M]. Beijing: Posts & Telecom Press, 2004. [Fernado Randima, Kilgard Mark J. Cg 教程——可编程实时图形权威指南. 北京:人民邮电出版社, 2004.]

16 Fernando Randima. GPU Gems Programming Techniques, Tips, and Tricks for Real-Time Graphics [M]. Beijing: Posts & Telecom Press, 2006. [Randima Fernando. GPU 精粹——实时图形编程的技术、技巧和技艺[M]. 北京:人民邮电出版社, 2006.]

Special Issue Call For Paper
International Journal of BIOMEDICAL
BIOMEDI SOFT COMPUTING AND HUMAN SCIENCES
Official English Journal of the Biomedical Fuzzy Systems Association

Machine Understanding of Human Behavior in Human Computer Interaction

Human computer interaction (HCI) is one of the foremost challenges of our society. New paradigms for interacting with computers will define the 21st century and enable the world to communicate and interact effortlessly and intuitively. A widely accepted prediction is that the emerging user interfaces should be human-centered, based on human models and accorded with human custom. They should include natural, human-like interactive functions including understanding and emulating certain human behaviors such as affective and social signaling, enabling computers to understand human behavior.

The goals of the special issue are to provide the reader with an overview of the state of the art in this emerging field, and to collect significant research results about new developments in this cross-discipline. We are soliciting original contributions which address a wide range of theoretical and application issues in Machine Understanding of Human Behavior in Human Computer Interaction including, but not limited to:

- New paradigms in HCI
- Detection and recognition of behavioral signals or cues
- Affective and attitudinal state detection and recognition
- Context model in behavioral signal analysis and context sensing in HCI
- Learning, especially unsupervised (or semi-supervised) in HCI
- Body communication explanation and understanding, including facial expression, head, hand body gesture detection and recognition
- Multimodal fusion for automatic human behavior analysis
- Human motion tracking and gesture recognition
- Multimodal event detection and recognition
- Perceptual user interfaces

INSTRUCTIONS FOR MANUSCRIPTS

Manuscripts should be submitted electronically in PDF or PS format by utilizing the system at http://www.f.waseda.jp/watada/BMFSA/journal-IJ/CallForEPaper/IJCall_for_Papers.pdf. Both the manuscripts and cover letter should be clearly marked to indicate that they are being submitted for consideration for this Special Issue. They will be logged and sent to the Special Issue Editors for review. Both full-length regular papers and short papers will be considered, subject to the normal Transactions page limits. Papers must not have been published previously or submitted for publication else-where. All papers will be reviewed by following the guidelines of the transactions.

IMPORTANT DATES

Nov. 2008-Jan. 2009 Call for papers published
Feb. 1st, 2009 Papers are due
Apr. 1st, 2009 Completion of first review
Jun. 1st, 2009 Completion of revisions
Jul. 1st, 2009 acceptance
Aug. 1st, 2009 Final manuscripts to the Editor
Sept. 1st, 2009 Publication of the Special Issue

Guest Editor XU Guangyou
Dept. of Computer, Tsinghua Univ.
徐光祐,清华大学计算机系.
E-maill: xgy-dcs@tsinghua.edu.cn