

# 基于图形处理器( GPU )的通用计算

吴恩华<sup>1,2)</sup> 柳有权<sup>1)</sup>

<sup>1)</sup> 中国科学院软件研究所计算机科学重点实验室 北京 100080)

<sup>2)</sup> 澳门大学科学技术学院电脑与资讯科学系 澳门)

**摘 要** 伴随着 PC 级微机的崛起和普及,多年来计算机图形的大部分应用发生了从工作站向微机的大转移,这种转移甚至发生在像虚拟现实、计算机仿真这样的实时(中、小规模)应用中。这一切的发生从很大程度上源自于图形处理硬件的发展和革新。近年来,随着图形处理器( GPU )性能的大幅度提高以及可编程特性的发展,人们首先开始将图形流水线的某些处理阶段以及某些图形算法从 CPU 向 GPU 转移。除了计算机图形学本身的应用,涉及到其他领域的计算,以至于通用计算近 2~3 年来成为 GPU 的应用之一,并成为研究热点。文中从若干图形硬件发展的历史开始,介绍和分析最新 GPU 在通用计算方面的应用及其技术原理和发展状况,并结合作者自身的实践讨论和探索其发展前景。

**关键词** 图形硬件;图形处理器( GPU );可编程性;实时计算  
中图法分类号 TP391

## General Purpose Computation on GPU

Wu Enhua<sup>1,2)</sup> Liu Youquan<sup>1)</sup>

<sup>1)</sup> Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080)

<sup>2)</sup> Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau)

**Abstract** Along with the rapid development of personal computers, various applications in computer graphics, even real-time applications in middle or small scale such as those in virtual reality and computer simulation, are migrating from workstations to PCs. To a large extent no doubt, the migration comes from the innovation of graphics hardware. For many years, with the advancement of graphics processing unit( GPU )and the creation of its new feature of programmability, scientists begin to transfer some of the processing stages in the graphics output pipeline or some graphics algorithms from CPU to GPU. To our surprise, except from those applications in graphics itself, GPU finds its applications in general purpose computations in other fields, and it comes up as a hot topic for research in recent 2~3 years. In this paper, starting from a brief introduction to some historical events on graphics hardware development, a detail introduction and analysis will be given to the technique and the latest development of GPU for general purpose computations, in combination with our own practice. Finally, a discussion is given to probe into the development perspective of GPU in this regard.

**Key words** graphics hardware; graphics processing unit( GPU ); programmability; real time computation

## 1 引言

综观计算机图形学的发展进程,可以毫不夸张地说,计算机图形学的每次重大进展都与图形处理硬件的突破密切相关. 20 世纪 80 年代初期, Geometry Engine( GE )<sup>[1]</sup>芯片的推出对于近 20 年来图形发展和变革产生了巨大的影响. GE 从设计上可以由一个寄存器的定制码定制出不同的功能,分别用于图形流水线中的几何变换、裁剪计算、投影缩放等. 在初期设计的 IRIS( Integrated Raster Imaging System )系列中, 12 个 GE 单元即可组成完整的三维图形流水线. 以 GE 作为核心技术,其设计者 Clark James 作为 CEO 所成立的 SGI 公司的产品在其后的图形发展和工业应用中产生了巨大影响. 例如,从 SGI 的 GL 直至今今天作为事实上图形界面工业标准的 OpenGL,一个为人熟知的、最突出的特点就是将其造型变换与观察变换结合为一体,用 ModelView 矩阵栈直接完成其统一的变换过程以及复杂物体的层次造型. 而这一特点恰恰是完全秉承了 20 年前源于 GE 的最初设计系统<sup>[1]</sup>,只不过最初的设计只使用了 8 层深度的硬件矩阵栈而已.

令人不无感叹的是,GE 诞生之前多少年来关于图形接口标准输出流水线是(如 PHIGS)否(如 GKS/GKS3D)应该包括造型功能的争论从此变得无足轻重,而在图形国际标准中将造型变换与观察变换截然分开处理的理所当然的做法和理论从此成了过眼云烟. 再优雅不过的设计也要服从于硬件运行的最佳效果,因此尽管人们 20 多年来在使用 GL/OpenGL 的 ModelView 编程时大感不爽,历经 20 多年的磨砺,亦变得习惯成自然了.

图形处理的并行性以及可编程功能一直是图形硬件发展所追求的目标,这其中的佼佼者莫过于北卡罗莱那大学在 20 世纪 80 年代所设计的 Pixel Planes 系列图形系统<sup>[2-4]</sup>. 其中,20 世纪 80 年代后期所设计的 Pixel Planes 5 作为当时图形处理能力最强的处理系统,曾经达到每秒绘制 100 万个 Phong 模型多边形的能力,这在今天看起来亦是一个值得惊叹的数字. Pixel Planes 的高度并行性是由像素级的驱动单元实现的,其硬件系统可以包括多达 32 个数学处理器和 16 个绘制部件,每个绘制部件可以对一个  $128 \times 128$  像素阵列的每个像素实行二次多项式的并行计算. 并行性采用的是 SIMD 结构. 从某种程度上说,它是一个专用于图形图像处理

的 SIMD 结构的并行处理机. 为此,Pixel Planes 虽然具有超级的处理速度,却不能不受 SIMD 结构的并行处理方式所制约,其高效性只适用于那些能够有效地分解其算法到 SIMD 结构上运行的图形应用.

伴随着 PC 级微机的崛起和普及,多年来计算机图形的大部分应用发生了从工作站向微机的大转移,而这种转移甚至发生在像虚拟现实、计算机仿真这样的实时(中、小规模)应用中. 实时应用的计算机游戏的普及亦是这一发展的标志. 这一切的发生从很大程度上源自于图形硬件的发展和革新. 随着计算技术和集成电路技术的发展,图形硬件的更新速度迅猛. GPU( Graphics Processing Unit )自 1999 年首先由 nVidia 公司<sup>[5]</sup>提出出来后,其发展的速度是 CPU 更迭速度的 3 倍多. 目前,图形芯片的主要市场被 nVidia 和 ATI 这两家公司占领,从高端到低端都有相应的产品来满足市场.

新的图形硬件带来一些新的特征<sup>[6-7]</sup>,这些特征概括起来有如下几方面:

(1) 在顶点级和像素级提供了灵活的可编程特性;

(2) 在顶点级和像素级运算上都支持 IEEE 32 位浮点运算;

(3) 支持多遍绘制的操作,避免了多次 CPU 与 GPU 之间的数据交换;

(4) 支持绘制到纹理的功能( Render-to-Texture / pbuffer ),从而避免将计算结果拷贝到纹理这个比较费时的过程;

(5) 支持依赖纹理功能,以方便数据的索引访问,可以将纹理作为内存来使用.

Pixel Planes 图形系统是利用通用机的结构实现图形并行处理的,而 GPU 却是完全专用于图形输出流水线的处理和加速. 因此当 GPU 的功能越来越强时,与图形有关的处理便自然而然地从 CPU 向 GPU 转移. 最先发生的转移是最靠近应用程序的几何变换部分,包括造型变换和观察变换;其次是局部或特殊光照效果的计算和生成. 当顶点级和像素级的可编程功能越来越灵活时,图形本身的处理速度和灵活性都得到了前所未有的提高. 而当 GPU 内部像素级的纹元达到可以参与编程的运算时,它从某种程度上模拟了类似于 Pixel Planes 处理单元的部分功能,以至于向着可作通用计算的方向发展. 这时,基于 GPU 的通用计算便应运而生了.

基于 GPU 的通用计算( General Purpose GPU, GPGPU )指的是利用图形卡来实现一般意义上的计



机器人运动规划,即利用颜色来标示障碍物多边形,然后将帧缓冲读回进行操作。Hoff 等<sup>[26]</sup>利用图形硬件的多边形光栅化过程和深度缓冲检测来计算二维和三维的通用 Voronoi 图,其中利用颜色来标示每个位置,通过光栅化重构出距离函数,而深度缓冲用来进行距离比较,从而得到最近或者最远的位置。

在光线跟踪方面,Carr 等<sup>[27]</sup>在像素级进行多条光线和一个三角形求交运算,其中,一幅纹理存放各条光线的起始点,另一幅纹理存放对应光线的方向,同时用三幅纹理来存放三角形,每次对一个三角形进行求交运算。Purcell 等<sup>[28]</sup>将整个场景采用均匀网格来表示,将所有三角形的顶点组织到三幅纹理中,并建立一个三角形链表纹理,这样可将整个光线跟踪的算法放到 GPU 上来执行。

Carr 等<sup>[29]</sup>针对辐射度和子表面散射光照模型采用 GPU 来加速计算,利用 GPU 雅可比迭代来求解矩阵辐射度方程。对于子表面算法,利用顶点编程或者像素编程形成若干辐射图,最后采用标准的 OpenGL 光照模型来进行绘制。Coombe 等<sup>[30]</sup>则进一步将整个辐射度的计算转移到 GPU 上去,将能量发射、可见性计算以及形状因子的计算全部放在 GPU 上完成,采用了建立纹理四叉树的策略对场景进行自适应细分。

Purcell 等<sup>[31]</sup>将 Photon Mapping 方法移植到 GPU 上,将 photon 存储在规则网格上,并给出了两种实现思路:一个是采用多遍技术利用像素程序对这些 photon 进行排序;另一个就是利用顶点程序结合模板缓冲将 photon 排布到网格上,这里模板缓冲用来控制写入位置。从而采用宽度优先的随机光线跟踪求解出全局光照效果,其中排序、路由以及搜索运算都是基于 GPU 实现。

Moreland 等<sup>[32]</sup>将图像处理从时域空间转换到频域空间,利用 GPU 实现快速傅里叶变换,通过灵活组织索引避免重新排序,虽然最终的效率不如 fftw 库,但却展示了 GPU 的一个很好应用前景。文献[33]利用 GPU 来做三维卷积运算,文献[34]则利用 GPU 来做小波变换,二者都是利用 OpenGL 1.2 以上版本提供的颜色矩阵和卷积操作来实现的。Trendall 等<sup>[35]</sup>利用硬件提供的颜色融合和 OpenGL 图像子集的卷积和颜色矩阵等运算功能计算折射散射效果。Rumpf 等<sup>[36]</sup>利用图形硬件求解传热和各向异性扩散有限单元方程,实现图像处理的功能。

James 等<sup>[37]</sup>利用现有的有限元包分析出变形物

体的自振频率,然后采用模态分析的方法利用顶点编程将物体各个振型进行叠加以得到新位置,这样可将以前在 CPU 上进行的计算全部转移到 GPU 上去。

在碰撞检测方面,文献[38]采用两遍绘制的方法来计算潜在碰撞集合,文献[39]中采用了多个 GPU 来加速可见性的计算,二者都利用了硬件遮挡查询操作来加速。

在代数运算方面,Larsen 等<sup>[40]</sup>利用多纹理技术实现了矩阵乘法。Hall 等<sup>[41]</sup>更进一步充分利用硬件的可编程性对矩阵乘法运算做了很多优化工作,使得计算效率大大提升。Thompson 等<sup>[42]</sup>基于顶点编程开发了一个框架系统用来进行矢量的运算,并在此基础上实现矩阵乘法和 3-SAT 问题的求解。Krüger 等<sup>[10]</sup>利用像素程序来做基本代数运算,然后在此基础上实现共轭梯度法和高斯-赛德尔迭代法,从而完成 PDEs 的求解。Bolz 等<sup>[21]</sup>实现了基于像素编程的稀疏非结构化矩阵的共轭梯度法和正交网格的多重网格法,并用于加速几何处理和流体模拟。Moravanszky<sup>[24]</sup>利用 DirectX 9 下的 HLSL 实现了稠密矩阵的系列代数运算,并在此基础上实现了共轭梯度法和线性优化问题求解。Hillesland 等<sup>[23]</sup>将最速下降法和共轭梯度法求解带有简单约束和规则化的非线性最小二乘优化问题映射到最新的图形硬件上,并将其应用到复杂的图像建模问题上。

Li 等<sup>[43]</sup>采用 LBM(Lattice Boltzmann Method)来模拟流体和烟的效果。通过将粒子包合成纹理,将 Boltzmann 方程组映射到光栅化和帧缓冲操作上,整个计算是采用 Register Combiner 来实现的。更进一步,在文献[44]中利用硬件计算 LBM 来模拟物体在风中飞舞的情形,虽然这时 GPU 已经开始支持像素级浮点精度,但考虑到计算速度的问题,仍然采用的是固定渲染管道。Harris 等<sup>[20]</sup>通过 Register Combiner 编程求解 CML(Coupled Map Lattice)问题,从而实现交互的对流模拟、反应扩散以及沸腾效果模拟。文献[45-46]中,开始采用像素程序来求解 PDEs 以模拟云彩的运动。

Goodnight 等<sup>[22,47]</sup>实现了基于像素程序的多重网格算法,用来求解边界值问题(热传导问题、流体力学问题)。Kim 等<sup>[48]</sup>将冰晶体生长过程采用 GPU 来实施物理计算,整个物理方程组的计算在 GPU 上执行。Lefohn 等<sup>[49-51]</sup>将 level-set 的等值面数据压缩为一个动态稀疏纹理格式,针对不同边界情况采用不同的像素程序来计算 level-set 的 PDEs;另外,

将下一个活跃的片段信息压缩成一个位矢量消息，从而将该消息传递给 CPU 以决定它需要传送给 GPU 的顶点和纹理坐标，这里还充分利用了硬件的自动 Mipmapping 技术来进行低采样，以截止到一个像素。

GPU 甚至被用来进行声音的合成运算，以使 CPU 和 APU 达到一个良好的均衡<sup>[52]</sup>。

可以看出，GPU 在通用计算方面的应用领域很广，下面将详细介绍目前利用 GPU 在通用计算方面，尤其是数值计算方面的应用。

3 基于 GPU 的代数运算

在图形硬件可编程性普及以前，对于图形卡也有一些数值计算方面的研究，如随着 nVidia 公司 GeForce2 的出现，多纹理技术被用来做一些通用的计算，结合纹理环境参数和纹理函数可以实现一些很灵活的应用<sup>[40]</sup>。文献 [20] 中，则开始利用 Texture Shader 的依赖纹理进行数据的访问，用 Register Combiner 进行计算。在可编程性普及以前，通常情况下这些方法都是结合起来使用，以满足不同的具体问题。但是这几种方法最大的问题就是精度和灵活性不够，因为它们都属于固定渲染管道下的方法，每个颜色通道的精度只有 8 位，所以在计算过程中不得不预先进行数据的规则化，并保证中间计算结果的规则化<sup>[35-36, 53]</sup>。这样一来，使得在通用计算方面有着很大的局限性。

下面主要介绍一下基于 Register Combiner 和基于像素程序这两种思路，因为这两种方法相对于其他方法更为灵活，所以实现的运算也更为复杂。Register Combiner 最早是在 GeForce 2 引入的，只有 2 个 General Combiner，目前最新的硬件上则可以支持 8 个 General Combiner，另外还有一个 Final Combiner 作为最后输出运算的。基于 Register Combiner 的运算需要根据其支持的运算形式和有限的 Combiner 个数来设计整个的流程，如图 2 所示，即数据以顶点颜色的形式或者纹理的形式传入 Register Combiner，利用其支持的运算可以同时完成 4 个颜色通道的计算。

目前，顶点程序不管是从支持的指令数还是寄存器的个数都较以前有了很大的提高，如 nVidia CineFX 目前每个顶点能支持 64K 条指令，常量寄

存器的个数和临时寄存器个数也都较以前有了增加，更重要的是开始支持动态和静态控制流的循环和分支操作以及子函数操作。

像素程序能够访问的独立的纹理目前达到 16 个之多，支持的指令数也多达 1 024 条，寄存器的个数增加到 64 个。

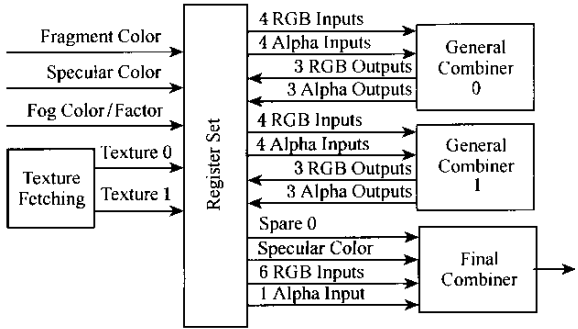


图 2 Register Combiner 总的流程

显然，与 Register Combiner 相比顶点编程和像素编程更具可编程性，有着更为灵活的操作，而且它们都提供 32 位的浮点精度，从而对于通用计算来说更为实用化，可以设计出更为复杂的运算。在采用顶点编程方面，文献 [42] 将矢量转换为一系列的四元组，实现了矢量、矩阵的一些运算。但由于需要光栅化这个过程，将结果送入纹理或者帧缓存中，然后再读回到主内存进行处理，因此如果数据量太大，可能会形成数据向显卡传输的带宽瓶颈。图 3、4 分别给出了顶点编程和像素编程的模型，可以看出二者很相似。虽然目前顶点编程已支持分支操作和循环，但不能访问纹理数据，而图形硬件的像素处理能力要比顶点处理能力高一个数量级，像素填充速度远比三角形传输速度快，且目前的像素渲染管道也要比顶点渲染管道多，所以像素程序更适合用来做通用计算。目前大多数文献均采用这种途径来做通用计算<sup>[10, 21]</sup>。

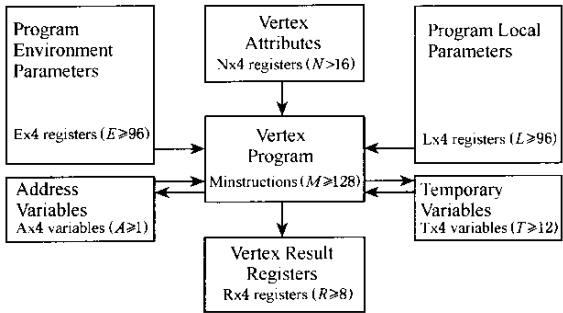


图 3 顶点程序模型



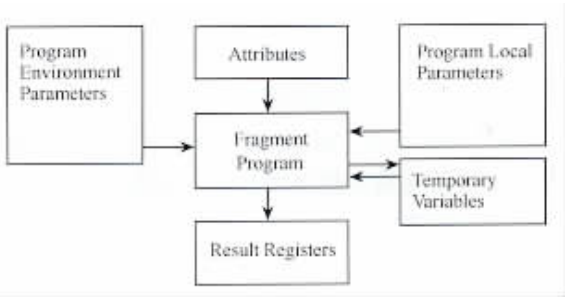


图 4 像素程序模型

根据目前已有的方法可以看出,整个基于 GPU 的通用计算的基本过程如图 5 所示,图的右边为选择的方法。

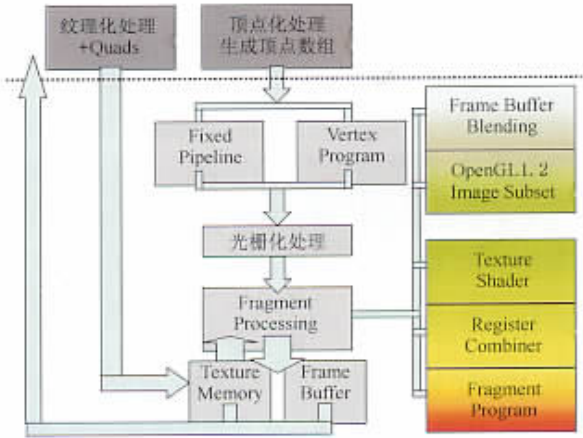


图 5 常见的基于 GPU 的通用计算流程图

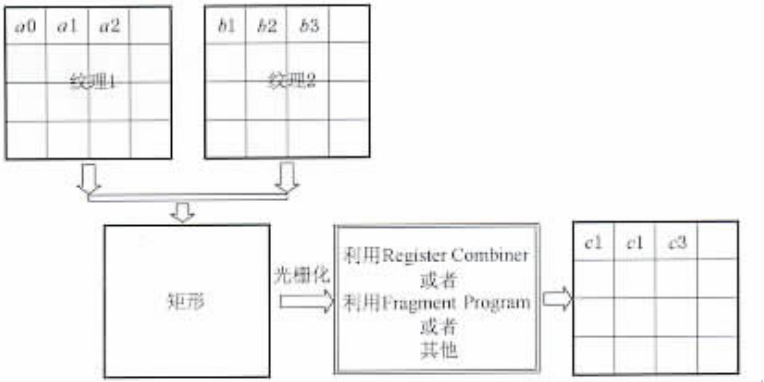


图 6 两个矢量在 GPU 上的运算过程

可以充分利用 RGBA 4 个颜色通道以减少处理的像素个数。例如,  $512^2$  维矢量采用 4 个颜色通道要比只采用一个颜色通道节省 3/4 的像素个数,但在实验中我们发现时间上只节省了 25%,效果没有预期的理想,主要的原因是像素填充率很高,而像素数目对性能不构成瓶颈。

基于前人的若干工作,我们采用像素编程在通用计算方面做了一些工作,对已有的算法提出了自己的改进,同时分析比较了各方法之间的效率。由于基于 GPU 的通用计算涉及的内容很广,而且与具体的问题紧密联系,本文主要讨论基于像素编程实现的一些基本代数运算,并在此基础上再来讨论线性方程组求解的 GPU 实现,从而更进一步应用到复杂 PDEs 的求解上去。

(1) 矢量与矢量的运算

在基于像素级上实现这种运算是很显然的,直接采用纹理来表示两个输入矢量,对应系数采用参数的方式传入,运算形式为  $r = a \cdot x \oplus b \cdot y$ ,其中运算符可以是加、减和乘三种操作。

为了充分比较各种方法的优劣,我们对于这种最简单的代数运算采用 Register Combiner 以及像素程序这两种思路来实现。虽然采用一维纹理表示矢量很直接,但硬件对于一维纹理的大小限制很多,而且硬件针对二维纹理进行了优化。与前人一样,我们采用二维纹理来表示,这样矢量的最大维数可以达到一维纹理所能表示的平方倍。下面详细地介绍一下具体的实现过程,以能更清楚地说明问题。

纹理化的过程就是用一幅纹理来表示该矢量的过程,而矢量的每个元素对应于该纹理的每个纹元。在具体实现中,需要将该幅纹理映射到一个与之匹配的矩形上,选取正投影模式,通过光栅化的过程来完成运算,输出的结果为帧缓存或者纹理,如图 6 所示。

图 7 所示为两个  $512^2$  维大小的矢量加法运算的几种方法的效率对比本文采用的是 4 个颜色通道的二维纹理表示方法 a 为利用 Register Combiner(8 位精度);b 为利用 16 位精度的像素程序;c 为利用 32 位精度的像素程序;d 为结合 Render to Texture 的 32 位精度像素程序。对于像素程序,我们尝试了

16 位和 32 位 的数据精度 ,其中 8 位精度利用的是固定渲染管道下 Register combiner 方法 ,可以看到明显的效率差异. 我们需要根据实际问题的需要来选择合适的方法. 而采用绘制到纹理( Render to Texture )的技术无疑可以大大加快计算的速度 ,避免了将结果拷贝到纹理这个很耗时的过程. 需要注意的是 ,在选择不同精度的时候 ,必须确保纹理精度和缓冲区精度一致 ,否则由于硬件要进行精度转换会导致性能严重下降.

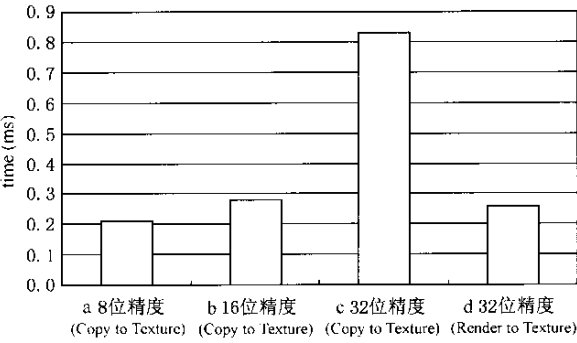


图 7 两个矢量加法运算的若干方法的效率对比

( 2 ) 稠密矩阵与矢量乘法运算

对于一般的矩阵与矢量的运算 ,最直接的方法就是采用纹元来表示对应位置上的数据<sup>[ 24 ]</sup> ,但为了方便利用基本运算 ,对行采用 4 个通道进行压缩表示. Krüger 等<sup>[ 10 ]</sup>采用对角线的方式来将矩阵划归为一系列的二维纹理<sup>⑤</sup> ,在像素程序中对因子矢量进行位置索引偏移 ,从而实现矩阵与矢量的乘法. 该方法对  $N \times N$  的矩阵与矢量乘法运算 ,需要进行  $N$  遍绘制才能完成. 该方法共需要 15 条指令 ,其中将二维坐标一维化占用 1 条 ,纹理获取 3 条 ,计算  $x$  的索引位置占 8 条 ,判断占 2 条 ,而真正用于计算的只有 1 条 ,整个需要处理的像素个数为  $N$ .

如果采用将 4 个相邻元素压缩至一个纹元的 4 个颜色通道 ,势必导致计算元素偏移的复杂度. 为此 ,我们基于文献 10 提出一种新的思路 ,不仅使得绘制的遍数大大减少 ,而且充分利用了硬件的资源 ,即如果硬件同时支持  $n$  个纹理获取操作 ,那么在一个像素程序中同时访问  $( n - 2 )$  个矩阵  $A$  的对角线矢量纹理<sup>⑥</sup> ,这样绘制的遍数减少为  $\lceil N/( n - 2 ) \rceil$  . 更进一步 ,将 4 个对角线元素压缩到一幅纹理以减少纹理访问的次数<sup>⑦</sup> ,那么在一个像素程序中就同

时可以访问  $4( n - 2 )$  个  $A$  矩阵的对角线矢量纹理 ,将绘制的遍数进一步减少为  $\lceil N/( 4 \times ( n - 2 ) ) \rceil$  ,处理像素个数也减少为  $\lceil N/4 \rceil$  . 这种方法通过增加指令数来最大限度地利用硬件资源 ,但可以大幅度减少绘制的次数 ,从而弥补了指令数增加带来的性能损失. 在本文实验的机器上 , $n = 4$  ,最后两种方法分别比原来方法<sup>[ 10 ]</sup>的效率提高了 2.0 倍和 8.6 倍 ,虽然总的指令数分别增加为 70 条和 96 条.

( 3 ) 稠密矩阵与稠密矩阵的乘法运算

文献 24 在计算矩阵乘法时把基本的  $4 \times 1$  矩阵与矢量作为原子 ,将整个运算化解为若干原子操作 ,根据矩阵的规模和目前硬件支持的最大指令数设定像素程序的长度 ,从而采用多遍绘制的思路来实现.

Larsen 等<sup>[ 40 ]</sup>将矩阵单元表示为纹理颜色值 ,通过设置纹理坐标来进行行和列的操作 ,将帧缓冲作为内存 ,利用多纹理和颜色融合技术实现了矩阵与矩阵的乘法. Hall 等<sup>[ 41 ]</sup>在文献 40 的基础上通过对矩阵分块存储 ,以充分利用高速缓存的作用 ,并充分利用 4 个颜色通道进行并行计算. 这样 ,不仅减少了指令数和需要处理像素的个数 ,还降低对带宽的要求 ,使得性能大大提升.

显然 ,稠密矩阵与稠密矩阵的乘法运算方法可以用于进行稠密矩阵与矢量乘法操作.

( 4 ) 稀疏矩阵与矢量的乘法运算

实际上矩阵通常为稀疏的 ,如 PDEs 的求解. 文献 10 中的稠密矩阵与矢量乘法运算所采用的方法对于带状矩阵特别方便 ,另外 ,文献 10 还提出了另一种处理随意稀疏的情况的方法 ,根据矩阵信息生成顶点几何信息 ,顶点的坐标记录着对应行号 ,颜色值为该元素的数据 ,而对应的纹理记录着列号信息 ;然后通过光栅化得到结果 ,需要  $N$  遍绘制完成.

Bolz 等<sup>[ 21 ]</sup>采用多个纹理来进行位置的索引<sup>①</sup> ,以减少对于稀疏矩阵中 0 元素的存储. 其中 ,对角线元素存放到纹理  $A_{ii}$  ,将非对角线上的非 0 元素压缩存放到另一个纹理  $A_{ij}$  ,而每一行非对角线上第一个非 0 元素在上一个纹理中的位置也存放到纹理  $R$  ,对应于矩阵中每一个非 0 元素的矢量  $X$  的分量位置同样也存放到纹理  $C$  . 由于各行非 0 元素个数不同 ,因此每组具有相同个数非 0 元素的行都有一个像素程序与之对应.

对于稀疏矩阵 ,Buck 等<sup>[ 13 ]</sup>将全部非 0 元素存储为一幅纹理  $elem$  中<sup>③</sup> , $iipos$  记录这些非 0 元素的

①~⑦ 见表 1 .

列号 ;*start* 记录每一行第一个非 0 元素在纹理 *elem* 中的位置 ;*len* 记录每一行非 0 元素的个数 ;整个运算共需要 6 幅纹理 . 每遍需要 16 条指令 ( 纹理获取指令占了 6 条 ) . 执行遍数为 *k* 次 , 这里 *k* 为各行中最大非 0 元素的个数 .

针对上面两种方法都可以做进一步的优化加速 . 对于文献 [21] 这种思路 , 我们在纹理 *R* 中除了记录每行非对角线上第一个非 0 元素在纹理 *A<sub>ij</sub>* 中的位置<sup>②</sup> , 同时记录下该行除过对角线元素的非 0 元素的个数 , 将相同布局的数据采用同一幅纹理来存储 . 整个只需要 2 个像素程序 , 需要执行的遍数也减少为每一行中除过对角线的非 0 元素最多的个数 + 1 . 最长的像素程序只需要 12 条指令 , 其中 4 条用于纹理获取 . 对于一个稀疏度为 50 % 的 1024 × 1024 大小、各行最大非 0 元素个数为 564 的矩阵来说 , 运算效率比原来的方法提高 2 % .

对于文献 [13] 这种思路 , 我们可以将 *len* 这幅纹理归并到纹理 *start* 的第 3 个分量中去 , 或者整个采用一个颜色通道来做计算<sup>④</sup> , 则 *elem* 和 *ipos* 可以整合成一幅纹理 , 而 *start* , *len* 以及输入的矢量因子 *x* 可以整合成一幅纹理 *x* . 这样做最大的好处是减少了纹理的访问 , 使得纹理获取指令从 6 条减少为 4 条 , 这样总的指令数也减少为 13 条 . 这种方法相对于文献 [21] 的方法来说 , 在对角线元素全部为 0 的情况下 , 可以将绘制的遍数减少一次 . 但通过实验我们发现这种方法所带来的性能提升不是很明显 .

表 1 所示为几种方法的效率的比较 , 可以看出 , 对于稀疏矩阵与矢量的乘法运算 , 前面几种方法效率无明显的区别 , 但后两种处理稀疏矩阵的方法比前几种要好 , 因为对于带状矩阵 , 我们改进文献 [10] 的方法减少了绘制的遍数 .

表 1 各种稀疏矩阵与矢量乘法方法的效率比较

方法	文献 [21] ①	文献 [21] 的改进②	文献 [13] ③	文献 [13] 的改进④	文献 [10] 带状矩阵⑤	文献 [10] 的改进⑥	文献 [10] 进一步改进⑦
稀疏程度	$[M]_{1024 \times 1024} \times V_{1024}$ ( 带状矩阵的宽度为 11 )						
绘制遍数	11	11	11	11	11	6	2
花费时间 (ms)	2.01	1.97	2.22	2.18	2.26	1.63	1.06
CPU 时间 (ms)				18.39			

( 5 ) 消减运算

消减运算将一个 *N* 维矢量 ( 二维纹理 ) 整理成一个标量 , 如求出该矢量中的最大、最小元素或者求该矢量的某个范数 . 我们采用的方法与文献 [10 , 21] 类似 , 但不是针对邻近的 4 个纹元操作 ( 如图 8 思路 1 所示 ) , 而是对整个纹理的一半进行偏移 , 不过这两种思路效率基本一样 . 由于硬件同时支持多个纹理获取操作 ( 如 *n* 个 ) , 因此如果只是采用一个颜色通道 , 那么需要的绘制遍数为  $\lceil \log_n N \rceil$  .

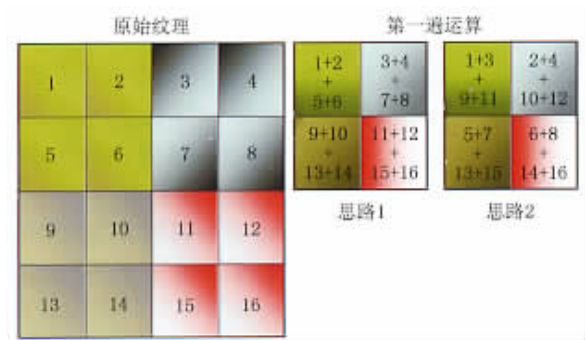


图 8 求矢量的消减运算过程

我们采用根据 4 个通道表示矢量 , 这样绘制遍数减少为  $\lceil \log_n (N/4) \rceil$  , 运算效率将比文献 [10 , 21] 的方法有大幅度提高 .

( 6 ) 雅可比迭代法

雅可比迭代方法比较简单 , 其迭代形式为  $x^{(k+1)} = D^{-1}[b - (L + U)x^{(k)}]$  . 其中 , *D*<sup>-1</sup> 为对角矩阵 ; *L* 和 *U* 分别为上三角和下三角矩阵 . 可以看出 , 雅可比迭代方法很适合进行并行处理 , *D*<sup>-1</sup> 生成一幅纹理 ( *L* + *U* ) 构成一幅纹理 *b* 为一幅纹理 , *x* 为一幅纹理 , 这样总共需要 4 幅纹理 . 利用上面提供的几种基本运算可以很容易地实现该算法 .

( 7 ) 共轭梯度法

共轭梯度法更为复杂 , 该方法迭代的具体形式见文献 [10] .

图 9 所示为共轭梯度法和雅可比迭代法迭代一次所用的时间比较 , 针对的是稠密矩阵 , 数据项都只占一个颜色通道 , 其中矩阵与矢量的乘法采用文献 [10] 方法 , 没有采用本文进一步改进的算法 , 计算精度为 32 位浮点数 , 方程规模为  $[A]_{1024 \times 1024} x_{1024} =$



$b_{1024}$ . 可以看出,共轭梯度法虽然整体收敛速度快,但在硬件上实现时需要消减运算,必须有一个从显存读回到主内存这个过程,而且这是一个多遍的操作,会导致每一次迭代的性能有所下降.

对于迭代收敛的条件判断方法,最简单的是采用限定迭代次数,该方法可以将计算的速度限定在一个数量上,实现起来比较简单,一般情况下使程序能够及时终止,而不是继续迭代下去.另一种方法是计算残差,可以在满足收敛精度的条件下终止迭代,但需要增加一个判断的过程;如果在 GPU 上实现,势必要增加 GPU 处理的遍数.可以采用硬件遮挡查询(NV-OCCLUSION-QUERY)来判断是否达到稳态<sup>[22]</sup>,对残差进行绘制,同时在像素程序中根据误差阈值决定是否处理该像素,从而根据硬件遮挡查询返回的像素个数判断是否达到收敛.

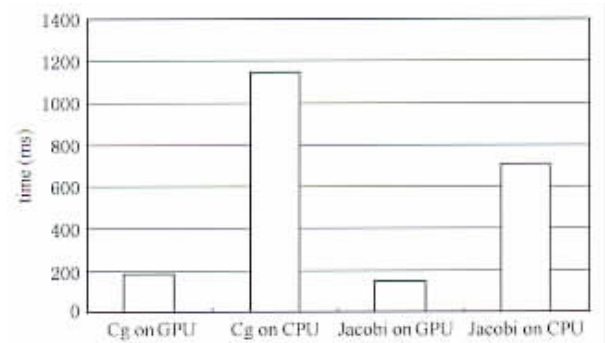


图9 基于CPU与基于GPU运算效率对比

(8) PDEs 求解

根据前面这些简单代数运算可以进一步拓展到复杂物理方程(PDEs)的求解.文献[21-22,47]针对多重网格的求解方法采用了不同的技术路线,文献[10]针对NS方程采用基本的运算模块来模拟流体效果,文献[45-46]求解云彩的运动方程,这些方法都是基于像素程序实现的.在GPU可编程性普及以前,文献[20]采用Register Combiner来求解扩散方程,更早的如文献[36,54]则只是采用多纹理或者OpenGL提供的图像子集函数来求解扩散方程.显然,不管是从精度还是从灵活性上,文献[10,21-22,45-47]更为实用.

这里我们采用NS方程组模拟烟雾的流动效果,其原始方程为

$$\begin{cases} \nabla \cdot \mathbf{u} = 0 \\ \frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla) \mathbf{u} + \nu \nabla^2 \mathbf{u} - \frac{\nabla p}{\rho} + \mathbf{f} \\ \frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla) \rho + k \nabla^2 \rho + S \end{cases} \quad (1)$$

整个方程的求解采用 Stam 的半拉格朗日法进行求解,这种方法绝对稳定,可以采用较大的时间步长.具体实现上我们采用像素程序来做整个计算和绘制,可以利用前面提到的各个基本模块来搭建整个复杂运算,但针对具体问题可以采用更为灵活的策略.相对于文献[10]的方法,由于速度和密度的扩散方程类似,我们将速度与密度两个变量整合成纹理的3个颜色通道(对于三维速度场类似),运算整合到一个像素程序中去,从而减少状态量的改变和整个绘制的遍数.对于源项和边界条件我们采用了相同的策略.

对于式(1)中扩散方程和压强泊松方程的求解我们均采用5点离散格式.为了快速求解,采用比高斯-赛德尔并行程度高的雅可比迭代的方法,我们只是固定迭代的次数,而没有进行收敛条件的判断,而这种权宜之计能够满足计算机图形学上的要求的,减少了判断所需的绘制过程.通过实验我们发现,基于GPU的算法比基于CPU的算法高出好几倍,甚至15倍以上.图10所示为一个绕流体结果,可以看到明显的卡门涡街效应(演示动画见<http://lcs.ios.ac.cn/~lyq/demos/gpgpu/gpgpu.htm>).



a 第1038帧



b 第1082帧

图10 绕流体效果,颜色块代表障碍物

文献[55]给出了采用像素程序来求解PDEs过程中涉及性能影响的一些具体细节.

4 性能分析

我们采用的实验平台为 Intel Pentium 2.8 GHz,主内存为 2 GB,而显卡采用的是 GeForce FX5950,显卡内存为 256 MB,显卡的核心频率为 375 MHz,驱动

的版本为 53.03, 操作系统为 Windows 2000, 整个实现基于 OpenGL. 本文给出的所有数据都是在该平台下得到的. 在我们使用的平台上, GPU 的每秒最大理论浮点指令数指标达到 11.7 GFLOPS, 而 CPU 的指标只有 2.8 GFLOPS. 测试的结果也证明了 GPU 在一些计算上已经大大超越了 CPU. 当然, 本文的测试和比较都没有经过很好的优化, 不管是 GPU 还是 CPU 上的性能仍然都有提升的空间.

Krishnar<sup>[56]</sup>提出了一个花费函数  $Cost Function = Np(Cp + Cr) + NrbCrb$ . 其中,  $Np$  表示体素的个数;  $Nrb$  表示需要读回的缓冲个数;  $Cp$  表示生成每个体素需要的时间;  $Cr$  表示绘制每个体素需要的时间;  $Crb$  表示每个缓冲读回需要的时间. 通过这个公式可以计算出整个花费的时间. 由于本文中主要运算都是基于像素程序, 而输入的几何数据很少, 即输入的体素是为数不多的矩形, 显然对基于 GPU 的通用计算来说运算的瓶颈在于像素片断的读回操作. 图 11 所示为两个  $512^2$  维的矢量相加运算的各个阶段时间耗费图, 其中送入的数据包括三个矢量的纹理数据. 可以看出, 将结果数据从纹理内存返回到主内存占了整个时间的绝大部分, 通常情况下绘制的结果将直接应用到下一遍处理中, 从而避免读回到主内存这一个过程.

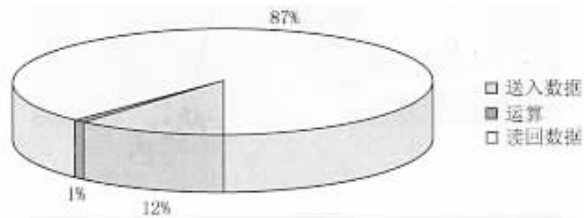


图 11 两个矢量相加运算的各个阶段的时间耗费

需要注意的是, 为了保证统计时间的准确性, 在 OpenGL 中通常需要采用 `glFinish` 来代替 `glFlush`, 因为后者是非阻塞式的, 只是将命令传给 GPU, 而不能保证所有的命令都执行完成.

## 5 结论与思考

本文针对目前图形硬件在通用计算上的应用开展了一系列的讨论, 并针对前人的工作做了若干改进, 试图更充分地利用硬件同时支持的多纹理个数来减少需要绘制的遍数, 使得性能得到进一步提高. 显然 GPU 在通用计算方面有着无比的优势和潜力.

可以看到, 结合具体的问题可以采用灵活的策略来减少绘制的次数, 或者减少纹理获取的指令数,

以充分发掘硬件的潜能来加速. 例如, 对于 RGBA 4 个通道就有着不同的处理方法, 一种方法就是将每个矢量采用一幅纹理表示, 而用该矢量的各个元素填充 RGBA 4 个通道; 如果是多个矢量同时参与运算, 亦可以将 4 个矢量合并到一幅纹理的 RGBA 4 个通道, 从而减少纹理访问的次数. 另外, 为了减少不同纹理间的不断切换, 可以采用将多幅纹理拼接到一幅纹理中去形成一幅纹理集, 以减少纹理访问次数和像素程序所需指令数, 从而得到加速. 具体在使用绘制到纹理功能的过程中, 为了减少 pbuffer 句柄切换所造成的瓶颈<sup>[21]</sup>, 可以采用多个绘制表面<sup>[47-49]</sup>的方法来代替多个 pbuffer 作更进一步的加速.

另外, 精度的选取对整个计算的速度有着很大的影响, 要根据实际的需要来进行选择.

目前图形硬件还存在很大的局限性, 如指令数的限制, 寄存器个数的限制, 同时由于访问纹理个数的限制, 使得像素程序不能进行分支和循环操作, 不能进行逻辑运算, 虽支持浮点类型却缺乏其他类型, 不支持任意的读写操作等. 对于诸如指令数数目和寄存器个数等资源的限制, 可以采用多遍绘制的思路, 将那些大规模问题分解为若干小问题来进行求解, 使每一个小问题所需要的资源满足其限制条件, 在其分解过程中可采用某些优化策略<sup>[57]</sup>. 全局寄存器的缺乏也是我们不得不采用多遍绘制思路的一个原因, 在实际计算中需要将中间运算的结果保存下来以避免多次的读写操作. 另外, 就是在工程上用到数值计算的精度通常为双精度的, 这些无不妨碍了在通用计算上的进一步应用.

在硬件上如果采用浮点运算, 则不能利用硬件本身的颜色融合操作, 也不能进行 Mipmapping 的自动运算, 这些也妨碍了 GPU 在通用计算上的应用, 这些问题应该可以在将来新的硬件中得到解决. 另一个可以预见的目标就是对显存的分配与访问将更加直接和方便, 如 Super buffer 和 Render to vertex array 等功能相继出现, 无疑将会带出一些新的应用.

利用多个 GPU 同时求解问题将是一个很有意义的发展方向. 文献[39]采用 2 个 GPU 分别作遮挡剔除操作之用, 用一个 GPU 来进行绘制. 研究人员正在探索如何组建 GPU 集群, 或者将多个 GPU 组合到一台机器中, 这些都正在变为现实.

由于图形硬件的发展速度很快, 一些新的特性还在不断地出现, 如 nVidia 公司和 ATI 都将有新的产品推出, 另外还有一些新的公司加入到这个领域.

就在本文完成之际,GDC2004( Game Developers Conference 2004 )刚刚结束不久,会上 nVidia 公司公布了今年下半年即将上市的 NV40 的很多新特性,如 GL\_NV\_vertex\_program3 将可以直接访问纹理,而 GL\_NV\_fragment\_program2 则开始支持循环和分支操作,并支持子函数运算.像上文中提到的一些局限也已经不再是问题,如在像素级将支持 16 位浮点精度的颜色融合,一个像素程序将允许 4 个输出.而实时绘制语言的普及也正推动着 GPU 的进一步应用.显然 GPU 将越来越强大,在通用计算方面的应用将更为深入.但不管怎么说,GPU 仍然是 GPU,这也带来了它的一个最大缺陷——至少到目前为止,与主存打交道的效率很低.这在某种程度上限制了它在通用计算上的应用.值得回味的是,两年多来,GPU 在通用计算上所开辟的应用以及相关的众多研究很像是其发展的一个副产品,可谓是一项无心插柳柳成荫的意外成果.那么,今天我们有理由去设想,是否可以在 GPU 的结构以及通用计算的未来发展上为其特意提供更有利的条件呢?例如我们可以设想以下两个有意思的问题(1)将来是否可能在总体结构上为 GPU 与主存之间开辟特别通道,以扩展 GPU 访问主存的能力,由此将会大大加强 GPU 的通用计算能力,目前新提出的 PCI-Express 可能是一个很好的解决方案(2)在将来通用计算软件(如数学库及其应用软件)的发展上是否有可能将 GPU 的功能充分利用起来,就像人们透明地使用 CPU 的浮点加速器一样,这将是一件令人向往而受益匪浅的事情,因为并不昂贵的高速 GPU 已可纳入计算机的常规配置.让我们静观其变,拭目以待.

参 考 文 献

[ 1 ] Clark James H. The geometry engine : A VLSI geometry system for graphics[ A ]. In : Computer Graphics Proceedings , Annual Conference Series , ACM SIGGRAPH , Boston , 1982. 127 ~ 133

[ 2 ] Fuchs Herry , Poulton John. Pixel-planes : A VLSI-Oriented design for a raster graphics engine[ J ]. VLSI Design , 1981 , 2 ( 3 ) : 20 ~ 28

[ 3 ] Eyles John , Austin John , Fuchs Henry , *et al* . Pixel-plane 4 : A summary , advances in computer graphics hardware II [ A ]. Eurographic Seminars Tutorials and Perspectives in Computer Graphics , New York : Springer-Verlag , 1988. 183 ~ 208

[ 4 ] Fuchs Herry , Israel Laura , Poulton John , *et al* . Pixel-planes 5 : A heterogeneous multiprocessor graphics system using processor-enhanced memories [ A ]. In : Computer Graphics Proceedings , Annual Conference Series , ACM SIGGRAPH , Boston , 1989: 79 ~ 88

[ 5 ] <http://www.nvidia.com/object/gpu.htm> [ OL ]

[ 6 ] <http://developer.nvidia.com/> [ OL ]

[ 7 ] <http://www.ati.com/developer/> [ OL ]

[ 8 ] <http://www.gpgpu.org/> [ OL ]

[ 9 ] João Luiz Dihl Comba , Dietrich Carlos A , Pagot Christian A , *et al* . Computation on GPUs : From a programmable pipeline to an efficient stream processor [ J ]. Revista de Informática Teórica e Aplicada , 2003 , 10 ( 2 ) : 41 ~ 70

[ 10 ] Krüger Jens , Westermann Rüdiger . Linear algebra operators for GPU implementation of numerical algorithms [ J ]. ACM Transactions on Graphics , 2003 , 22 ( 3 ) : 908 ~ 916

[ 11 ] Macedonia Michael . The GPU enters computing 's mainstream [ J ]. Computer , 2003 , 36 ( 10 ) : 106 ~ 108

[ 12 ] Venkatasubramanian Suresh . The graphics card as a stream computer [ OL ]. <http://www.research.att.com/areas/visualization/papers/videos/papers/2003v.pdf>

[ 13 ] Buck Ian , Hanrahan Pat . Data parallel computation on graphics hardware [ OL ]. <http://graphics.stanford.edu/projects/brookgpu/>

[ 14 ] Mark William R , Ghanville R Steven , Akeley Kurt , *et al* . Cg : A system for programming graphics hardware in a C-like language [ J ]. ACM Transactions on Graphics , 2003 , 22 ( 3 ) : 896 ~ 907

[ 15 ] <http://www.cgshader.org/> [ OL ]

[ 16 ] <http://www.microsoft.com/directx/> [ OL ]

[ 17 ] <http://www.opengl.org/> [ OL ]

[ 18 ] Cohen Michael F , Chen Shenchang Eric , Wallace John R , *et al* . A progressive refinement approach to fast radiosity image generation [ A ]. In : Computer Graphics Proceedings , Annual Conference Series , ACM SIGGRAPH , Atlanta , 1988. 75 ~ 84

[ 19 ] Lindholm Erik , Kilgard Mark J , Moreton Henry . A user-programmable vertex engine [ A ]. In : Computer Graphics Proceedings , Annual Conference Series , ACM SIGGRAPH , Los Angeles , 2001. 149 ~ 158

[ 20 ] Harris Mark J , Coombe Greg , Scheuermann Thorsten , *et al* . Physically-based visual simulation on graphics hardware [ A ]. In : Proceedings of Graphics Hardware , Saarbrücken , 2002. 109 ~ 118

[ 21 ] Bolz Jeff , Farmer Ian , Grinspun Eitan , *et al* . Sparse matrix solvers on the GPU : Conjugate gradients and multigrid [ J ]. ACM Transactions on Graphics , 2003 , 22 ( 3 ) : 917 ~ 924

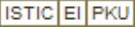
[ 22 ] Goodnight Nolan , Woolley Cliff , Luebke David , *et al* . A multigrid solver for boundary value problems using programmable graphics hardware [ A ]. In : Proceedings of Graphics Hardware , San Diego , 2003. 102 ~ 111

[ 23 ] Hillesland Karl E , Molinov Sergey , Grzeszczuk Radek . Nonlinear optimization framework for image-based modeling on programmable graphics hardware [ J ]. ACM Transactions on Graphics , 2003 , 22 ( 3 ) : 925 ~ 934

[ 24 ] Moravanszky Adam . Dense matrix algebra on the GPU [ OL ]. [www.shaderx2.com](http://www.shaderx2.com) , 2003

[ 25 ] Lengyel Jed , Reichert Mark , Donald Bruce R , *et al* . Real-time robot motion planning using rasterizing computer graphics hardware [ A ]. In : Computer Graphics Proceedings , Annual Conference Series , ACM SIGGRAPH , Dallas , 1990. 327 ~ 335

- [ 26 ] Hoff Kenneth E , III , Keyser John , Lin Ming , *et al.* Fast computation of generalized voronoi diagrams using graphics hardware[ A ]. In : Computer Graphics Proceedings , Annual Conference Series , ACM SIGGRAPH , Los Angeles , 1999. 277 ~ 286
- [ 27 ] Carr Nathan A , Hall Jesse D , Hart John C. The ray engine [ A ]. In : Proceedings of Graphics Hardware , Saarbrücken , 2002. 37 ~ 46
- [ 28 ] Purcell Timothy J , Buck Ian , Mark William R , *et al.* Ray tracing on programmable graphics hardware [ J ]. ACM Transactions on Graphics , 2002 , 21( 3 ) : 703 ~ 712
- [ 29 ] Carr Nathan A , Hall Jesse D , Hart John C. GPU algorithms for radiosity and subsurface scattering [ A ]. In : Proceedings of Graphics Hardware , San Diego , 2003. 51 ~ 59
- [ 30 ] Coombe Greg , Harris Mark J , Lastra Anselmo. Radiosity on graphics hardware[ OL ]. <http://www.cs.unc.edu/~coombe/research/radiosity/>
- [ 31 ] Purcell Timothy J , Donner Craig , Cammarano Mike , *et al.* Photon mapping on programmable graphics hardware[ A ]. In : Proceedings of Graphics Hardware , San Diego , 2003. 41 ~ 50
- [ 32 ] Moreland Kenneth , Angel Edward. The FFT on a GPU[ A ]. In : Proceedings of Graphics Hardware , San Diego , 2003. 112 ~ 119
- [ 33 ] Hopf Matthias , Ertl Thomas. Accelerating 3D convolution using graphics hardware[ A ]. In : Proceedings of IEEE Visualization , San Francisco , 1999. 471 ~ 474
- [ 34 ] Hopf Matthias , Ertl Thomas. Hardware accelerated wavelet transformations [ A ]. In : Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym , Netherlands , 2000. 93 ~ 103
- [ 35 ] Trendall Chris , Stewart James. General calculations using graphics hardware , with application to interactive caustics[ A ]. In : Proceedings of Eurographics Workshop on Rendering , Brno , 2000. 287 ~ 298
- [ 36 ] Rumpf Martin , Strzodka Robert. Using graphics cards for quantized FEM computations [ A ]. In : Proceedings of VIIP , Marbella , 2001. 98 ~ 107
- [ 37 ] James Doug L , Pai Dinesh K. DYRT : Dynamic response textures for real time deformation simulation with graphics hardware[ J ]. ACM Transactions on Graphics , 2002 , 21( 3 ) : 582 ~ 585
- [ 38 ] Govindaraju Naga , Redon Stephane , Lin Ming C , *et al.* CULLIDE : Interactive collision detection between complex models in large environments using graphics hardware[ A ]. In : Proceedings of Graphics Hardware , San Diego , 2003. 25 ~ 32
- [ 39 ] Govindaraju Naga , Sud Avneesh , Yoon Sung-Eui , *et al.* Parallel occlusion culling for interactive walkthroughs using multiple GPUs [ R ]. Carolina : University of North Carolina at Chapel Hill. UNC Computer Science Technical Report TR02-027 , 2002
- [ 40 ] Larsen E Scott , McAllister David. Fast matrix multiplies using graphics hardware [ A ]. In : Proceedings of Supercomputing , Denver , 2001. 55 ~ 60
- [ 41 ] Hall Jesse D , Carr Nathan A , Hart John C. Cache and bandwidth aware matrix multiplication on the GPU [ R ]. Champaign : University of Illinois at Urbana-Champaign. UIUCDCS-R-2003-2328 , 2003
- [ 42 ] Thompson Chris J , Hahn Sahngyun , Oskin Mark. Using modern graphics architectures for general-purpose computing : A framework and analysis [ A ]. In : Proceedings of International Symposium on Microarchitecture , Istanbul , 2002. 306 ~ 317
- [ 43 ] Li Wei , Wei Xiaoming , Kaufman Arie. Implementing lattice Boltzmann computation on graphics hardware [ J ]. The Visual Computer , 2003 , 19 : 44 ~ 456
- [ 44 ] Wei Xiaoming , Zhao Ye , Fan Zhe , *et al.* Blowing in the wind [ A ]. In : Proceedings of SIGGRAPH/Eurographics Symposium on Computer Animation , San Diego , 2003. 75 ~ 85
- [ 45 ] Harris Mark J , Coombe Greg , Scheuermann Thorsten , *et al.* Simulation of cloud dynamics on graphics hardware [ A ]. In : Proceedings of Graphics Hardware , Saarbrücken , 2003. 92 ~ 101
- [ 46 ] Harris Mark J. Real-time cloud simulation and rendering [ D ]. Chapel Hill : The University of North Carolina , 2003
- [ 47 ] Goodnight Nolan , Lewin Gregory , Luebke David , *et al.* A multigrid solver for boundary value problems using programmable graphics hardware [ R ]. University of Virginia , CS-2003-03 , 2003
- [ 48 ] Kim Theodore , Lin Ming C. Visual simulation of ice crystal growth [ A ]. In : Proceedings of SIGGRAPH/Eurographics Symposium on Computer Animation , San Diego , 2003. 86 ~ 97
- [ 49 ] Lefohn Aaron E , Kniss Joe M , Hansen Charles D , *et al.* Interactive deformation and visualization of level set surfaces using graphics hardware [ A ]. In : Proceedings of IEEE Visualization , Seattle , 2003 , 15 ~ 82
- [ 50 ] Lefohn E Aaron , Kniss M Joe , Hansen D Charles , *et al.* Interactive deformation and visualization of level set surfaces using graphics hardware [ R ]. Salt Lake City : University of Utah. UUCS-03-005 , 2003
- [ 51 ] Lefohn Aaron E , Kniss Joe M , Hansen Charles D , *et al.* A streaming narrow-band algorithm : Interactive computation and visualization of level sets [ OL ]. <http://www.sci.utah.edu/~lefohn/work/rls/visLevelSet/>
- [ 52 ] [http://www.gamasutra.com/resource\\_guide/20030528/tsingos\\_01.shtml](http://www.gamasutra.com/resource_guide/20030528/tsingos_01.shtml) [ OL ]
- [ 53 ] Harris Mark J. Analysis of error in a CML diffusion operation [ R ]. UNC Chapel Hill : The University of North Carolina , TR02-015 , Carolina : University of North Carolina at Chapel Hill , 2002
- [ 54 ] Strzodka Robert , Rumpf Martin. Nonlinear diffusion in graphics hardware [ A ]. In : Proceedings of Visualization , San Diego , 2001. 75 ~ 84
- [ 55 ] Batty Chris , Wiebe Mark , Houston Ben. High performance production-quality fluid simulation via Nvidia's quadro FX [ OL ]. <http://film.nvidia.com/docs/CP/4449/frantic-GPUAccelerationofFluids.pdf> , 2003
- [ 56 ] Krishnan Shankar. SIGGRAPH2002 Course [ OL ]. <http://www.cs.unc.edu/~geom/SIG02-COURSE/krishnan.ppt>
- [ 57 ] Chan Eric , Ng Ren , Sen Pradeep , *et al.* Efficient partitioning of fragment shaders for multipass rendering on programmable graphics hardware [ A ]. In : Proceedings of Graphics Hardware , Saarbrücken , 2002. 69 ~ 78

作者: 吴恩华, 柳有权  
作者单位: 吴恩华(中国科学院软件研究所计算机科学重点实验室, 北京, 100080; 澳门大学科学技术学院  
电脑与资讯科学系, 澳门), 柳有权(中国科学院软件研究所计算机科学重点实验室, 北京  
, 100080)  
刊名: 计算机辅助设计与图形学学报   
英文刊名: JOURNAL OF COMPUTER-AIDED DESIGN & COMPUTER GRAPHICS  
年, 卷(期): 2004, 16(5)  
被引用次数: 101次

参考文献(57条)

1. Clark James H The geometry engine:A VLSI geometry system for graphics 1982
2. Fuchs Herry. Poulton John Pixel-planes:A VLSI-Oriented design for a raster graphics engine 1981(03)
3. Eyles John. Austin John. Fuchs Henry Pixel-plane 4:A summary, advances in computer graphics hardware  
II 1988
4. Fuchs Herry. Israel Laura. Poulton John Pixel-planes 5:A heterogeneous multiprocessor graphics  
system using processor-enhanced memories 1989
5. [查看详情](#)
6. [查看详情](#)
7. [查看详情](#)
8. [查看详情](#)
9. Joo Luiz Dihl Comba. Dietrich Carlos A. Pagot Christian A Computation on GPUs:From a programmable  
pipeline to an efficient stream processor 2003(02)
10. Krüger Jens. Westermann Rüdiger Linear algebra operators for GPU implementation of numerical  
algorithms 2003(03)
11. Macedonia Michael The GPU enters computing's mainstream 2003(10)
12. Venkatasubramanian Suresh The graphics card as a stream computer
13. Buck Ian. Hanrahan Pat Data parallel computation on graphics hardware
14. Mark William R. Glanville R Steven. Akeley Kurt Cg:A system for programming graphics hardware in a  
C-like language 2003(03)
15. [查看详情](#)
16. [查看详情](#)
17. [查看详情](#)
18. Cohen Michael F. Chen Shenchang Eric. Wallace John R A progressive refinement approach to fast  
radiosity image generation 1988
19. Lindholm Erik. Kilgard Mark J. Moreton Henry A user-programmable vertex engine 2001
20. Harris Mark J. Coombe Greg. Scheuermann Thorsten Physically-based visual simulation on graphics  
hardware 2002
21. Bolz Jeff. Farmer Ian. Grinspun Eitan Sparse matrix solvers on the GPU:Conjugate gradients and  
multigrid 2003(03)
22. Goodnight Nolan. Woolley Cliff. Luebke David A multigrid solver for boundary value problems using  
programmable graphics hardware 2003



23. [Hillesland Karl E. Molinov Sergey. Grzeszczuk Radek Nonlinear optimization framework for image-based modeling on programmable graphics hardware 2003\(03\)](#)
24. [Moravanszky Adam Dense matrix algebra on the GPU 2003](#)
25. [Lengyel Jed. Reichert Mark. Donald Bruce R Real-time robot motion planning using rasterizing computer graphics hardware 1990](#)
26. [Hoff Kenneth E. III. Keyser John. Lin Ming Fast computation of generalized voronoi diagrams using graphics hardware 1999](#)
27. [Carr Nathan A. Hall Jesse D. Hart John C The ray engine 2002](#)
28. [Purcell Timothy J. Buck Ian. Mark William R Ray tracing on programmable graphics hardware 2002\(03\)](#)
29. [Carr Nathan A. Hall Jesse D. Hart John C GPU algorithms for radiosity and subsurface scattering 2003](#)
30. [Coombe Greg. Harris Mark J. Lastra Anselmo Radiosity on graphics hardware](#)
31. [Purcell Timothy J. Donner Craig. Cammarano Mike Photon mapping on programmable graphics hardware 2003](#)
32. [Moreland Kenneth. Angel Edward The FFT on a GPU 2003](#)
33. [Hopf Matthias. Ertl Thomas Accelerating 3D convolution using graphics hardware 1999](#)
34. [Hopf Matthias. Ertl Thomas Hardware accelerated wavelet transformations 2000](#)
35. [Trendall Chris. Stewart James General calculations using graphics hardware, with application to interactive caustics 2000](#)
36. [Rumpf Martin. Strzodka Robert Using graphics cards for quantized FEM computations 2001](#)
37. [James Doug L. Pai Dinesh K DYRT: Dynamic response textures for real time deformation simulation with graphics hardware 2002\(03\)](#)
38. [Govindaraju Naga. Redon Stephane. Lin Ming C CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware 2003](#)
39. [Govindaraju Naga. Sud Avneesh. Yoon Sung-Eui Parallel occlusion culling for interactive walkthroughs using multiple GPUs 2002](#)
40. [Larsen E Scott. McAllister David Fast matrix multiplies using graphics hardware 2001](#)
41. [Hall Jesse D. Carr Nathan A. Hart John C Cache and bandwidth aware matrix multiplication on the GPU 2003](#)
42. [Thompson Chris J. Hahn Sahngyun. Oskin Mark Using modern graphics architectures for general-purpose computing: A framework and analysis 2002](#)
43. [Li Wei. Wei Xiaoming. Kaufman Arie Implementing lattice Boltzmann computation on graphics hardware 2003](#)
44. [Wei Xiaoming. Zhao Ye. Fan Zhe Blowing in the wind 2003](#)
45. [Harris Mark J. Coombe Greg. Scheuermann Thorsten Simulation of cloud dynamics on graphics hardware 2003](#)
46. [Harris Mark J Real-time cloud simulation and rendering 2003](#)
47. [Goodnight Nolan. Lewin Gregory. Luebke David A multigrid solver for boundary value problems using programmable graphics hardware 2003](#)

48. [Kim Theodore, Lin Ming C Visual simulation of ice crystal growth 2003](#)

49. [Lefohn Aaron E, Kniss Joe M, Hansen Charles D Interactive deformation and visualization of level set surfaces using graphics hardware 2003](#)

50. [Lefohn E Aaron, Kniss M Joe, Hansen D Charles Interactive deformation and visualization of level set surfaces using graphics hardware 2003](#)

51. [Lefohn Aaron E, Kniss Joe M, Hansen Charles D A streaming narrow-band algorithm: Interactive computation and visualization of level sets](#)

52. [查看详情](#)

53. [Harris Mark J Analysis of error in a CML diffusion operation 2002](#)

54. [Strzodka Robert, Rumpf Martin Nonlinear diffusion in graphics hardware 2001](#)

55. [Batty Chris, Wiebe Mark, Houston Ben High performance production-quality fluid simulation via Nvidia's quadro FX 2003](#)

56. [Krishnan Shankar SIGGRAPH2002 Course](#)

57. [Chan Eric, Ng Ren, Sen Pradeep Efficient partitioning of fragment shaders for multipass rendering on programmable graphics hardware 2002](#)

相似文献(10条)

1. 学位论文 [李东魁 基于可编程硬件的骨骼蒙皮动画的设计与实现 2006](#)

角色动画是最近十年来计算机科学最活跃的领域之一，它广泛应用于虚拟现实、计算机辅助设计、广告媒体和数字娱乐等诸多领域。随着计算机硬件技术的发展，特别是消费级别的带有硬件加速功能的显卡技术的发展，实时角色动画逐渐获得了越来越广泛的应用。实时角色动画具有巨大的经济前景和理论研究意义。本文对可编程图形硬件和实时角色动画做了深入研究，给出了现在使用最为广泛的实时角色动画——骨骼蒙皮动画在可编程图形硬件上的设计实现。

实时角色动画分为三种：关节动画、单一网格模型动画和骨骼蒙皮动画，本文对这三种动画技术做了深入研究。骨骼蒙皮动画可被看作关节动画和单一网格模型动画的结合，它不但具有前两种动画的优点，同时克服了前两种动画的缺点。但是骨骼蒙皮动画使用的顶点混合技术需要大量的矩阵运算，以前这是由CPU来实现的，这给CPU造成了极大的负担，降低了动画的实时性。现在可编程硬件图形硬件的发展为解决这一问题提供了可能。

本文对可编程图形硬件及其编程语言和编程方法做了详细介绍。目前图形硬件中的图形处理器(GPU)计算能力的增长速度已经超过了CPU计算能力的增长速度，图形硬件技术一个重大突破就是在图形硬件中引入了可编程功能，该功能允许开发人员编制自己的着色器程序(Shaderprogram)来替换原来固定流水线中的某些功能模块，以实现更为灵活的功能。虽然GPU具有非常高的计算速度，但不能将以前在CPU中实现的算法直接放到GPU中来执行，这是因为GPU的指令执行方式和CPU不一样，GPU的体系结构是一种高度并行的单指令多数据(SIMD)指令执行体系，所以要在可编程图形硬件上实现在CPU中效率不高的算法，就需要重新制定算法实现的数据结构和步骤，以充分利用GPU并行处理体系结构带来的性能优势。本文首先给出了基于CPU的骨骼蒙皮动画的设计实现，对其中可放到GPU中算法进行了分析，给出了基于GPU的骨骼蒙皮动画的设计实现，实现的测试结果表明基于GPU的骨骼蒙皮动画比基于CPU的骨骼蒙皮动画帧率要高出很多，极大提高了骨骼蒙皮动画的实时性。本文还给出了动画混合技术的实现，这是通过对骨骼的本地变换进行按比例混合来实现的，这一实现不仅增加了动画动作，还提高了动画质量。

2. 学位论文 [杨晓玲 基于GPU的LBM方法计算研究 2008](#)

图形处理器(GPU)最近几年迅速发展，基于GPU的计算作为一个新的研究方向已经引起了越来越多人的关注。当今的图形硬件有着高度的并行性和很高的存储带宽，这使得GPU比CPU更适合于流处理计算。随着GPU性能和可编程性的提高，基于可编程图形硬件的通用计算成为近些年来图形学领域研究的热点。

建立在微观模型上的格子Boltzmann方法是近年来发展起来的一种模拟流体流动新的计算方法。格子Boltzmann方法具有计算简单，能够处理复杂边界问题，以及适合于并行计算等很多优点。

本文主要研究基于GPU的格子Boltzmann方法计算，试图利用图形硬件的最新进展，以新的图形硬件的可编程性和一定程度的并行性加速通用数值运算，实现基于格子Boltzmann方法计算的加速。本文的主要研究内容如下：

- 1、在分析GPU编程特点的基础上，建立基于GPU计算的编程环境，研究在GPU环境下进行计算的特点。
- 2、在分析格子Boltzmann方法基本原理的基础上，建立基于LBM的数值模型，分析GPU数据存储的特点，通过纹理映射将格子Boltzmann方法的数据映射到GPU的数据结构上，将格子Boltzmann方法的数据储存在纹理中。
- 3、采用Cg语言编制在GPU上实现格子Boltzmann方法计算的程序，在GPU上实现用LBM方法模拟方腔流，完成对软件的测试和数值实验，计算t值不同时的结果分析方腔流的特点。实验结果表明基于GPU的计算有了明显的加速。

3. 学位论文 [郑杰 基于GPU的高质量交互式可视化技术研究 2007](#)

近十年来，可视化技术成为科学研究的一个热点方向。随着医学影像技术的发展，迫切需要有效的可视化技术对海量体数据进行分析。由于体绘制技术可以真实地显示三维物体内部信息，逐渐成为主要的可视化手段。高质量的体绘制效果需要大量的运算，从而导致交互性能的严重下降，不能应用于许多对实时性要求较高的领域。同时，由于缺乏适合于描述和建模真实物体内在特性的物理模型，绘制参数的选择和交互工具的设计对体绘制非常关键。本文重点研究基于通用图形处理器(GPU)的直接体绘制技术，力图在绘制的真实感与交互性能之间寻求一个良好的平衡，最终设计实现能够具有应用价值的可视化处理系统。本文的主要贡献包括以下四个方面：

- 1、在体绘制的光照效果方面，利用GPU的可编程性提出了一种基于per-pixel光照的体绘制算法。在绘制过程中使用归一化梯度对每个像素实时计算光照贡献，明显改善了绘制的明暗效果，在一个绘制通道中完成所有绘制过程，得到可交互的高质量绘制结果。
- 2、在体绘制的传递函数设计方面，利用GPU的纹理特性提出了一种基于空间信息的交互式多维传递函数设计算法。在传递函数中根据体数据值和位置信息对局部空间区域指定绘制参数，并将整个绘制过程映射到GPU上，实现了对数据场中感兴趣区域的自由绘制。
- 3、在体数据的交互式分析方面，利用图像分割理论根据组织结构对数据场分类，提出了一种基于组织分割的多物体混合体绘制算法。在GPU的像素处理阶段完成对分割物体的独立绘制，实现了数据场中不同物体结构的快速分析。另外，利用体素的空间位置完成数据场区域分类，提出了一种基于空间区域标识的交互式体切割算法。绘制中GPU的并行处理能力提高了切割操作的实时性，实现了对数据场中任意区域隐藏信息的分析。

4. 在大规模数据场实时绘制方面,充分利用GPU资源,提出了一种基于动态纹理载入的实时体绘制算法。通过分块绘制的方式改进绘制流程,图形硬件中仅存储一部分体数据,并在绘制中实时计算梯度,减少纹理内存的占用,在普通PC平台上实现了大规模数据场的高质量实时体绘制。

4. 期刊论文 [张杨, 诸昌铃, 何太军. ZHANG Yang, ZHU Chang-qian, HE Tai-jun 图形硬件通用计算技术的应用研究 - 计算机应用2005, 25 \(9\)](#)

在通用计算的图形硬件加速研究中,综合了在OPENGL体系下的计算模型.通过实验,测试了该计算结构的性能并分析了提高计算性能的一些方法.在此基础上,介绍一种基于GPU的并行计算二维离散余弦变换方法.该方法可在GPU上通过一遍绘制,对一幅图像1至4个颜色通道,同时进行8×8大小像素块的离散余弦变换.实验表明在该实验硬件基础上,采用GPU加速的并行离散余弦变换,可比相同算法的CPU实现提高数百倍。

5. 学位论文 [姚远 基于GPU的实时水体模拟及刚体和水体交互的设计与实现 2006](#)

对海面,江水,湖泊等水体效果的模拟,一直以来是真实感图形学的热门技术。为在应用中获得更强的真实感,人们将流体动力学方法引入计算机图形学,并获得成功。然而,流体动力学方法的不易控制性和离线模拟机制,使之不能满足三维电子游戏和虚拟漫游对实时帧速率的要求。随着图形处理器 GPU(Graphics Processing Unit, 图形处理器)的出现,围绕GPU硬件功能来编写渲染结构已经成为新的趋势,而基于可编程GPU硬件加速的实时水体效果模拟也成为可能。

本文的主要工作是设计并实现了一个基于GPI加速,并适用于实时三维游戏等应用的水体模拟系统。论文分析了前人模拟流体的几种主流方法,介绍了 GPU和HLSL(High Irevel Shading Language, 高级着色语言),在此基础上提出了基于GPU加速的水体模拟系统的算法理论,渲染架构以及具体实现。论文首先用周期波的叠加来模拟水面的高度场,然后用凹凸纹理映射模拟法线的扰动,生成逼真的水面形状,然后模拟出水面的反射、折射、Fresnel和高光等现象,再现光和水面的相互作用,生成极其生动逼真的水面效果。论文还讨论了合成凹凸贴图,粒子系统等理论及方法在水体模拟中的应用,以实现刚体与水体交互时的波纹、泡沫飞溅等自然现象的模拟。

为能达到实时的水体模拟,论文用基于GPU加速的可编程硬件语言HLSL来编写Vertex shader(顶点着色器)和Pixel shader(像素着色器)。Vertex shader模拟水体的高度场和法线图,Pixel shader用于水体的渲染。经工程实践证明,该水体模拟系统在主流硬件上可达到 100 以上的帧速率(FPS),完全满足三维游戏中对实时帧速率的需求。本文所开发的系统已应用于XBOX360。

6. 期刊论文 [封卫兵, 杨晓玲. FENG Wei-bing, YANG Xiao-ling 基于图形处理器的格子Boltzmann方法计算 - 上海大学学报\(自然科学版\)2009, 15 \(1\)](#)

由于图形处理器(GPU)最近几年迅速发展,基于GPU的计算作为一个新的研究方向已经引起越来越多人关注.在综述国内外最新文献的基础上,从介绍GPU的高性能开始,分析GPU本身的特性,介绍GPU的计算模型并分析其流水线结构,阐述如何对GPU进行编程,并初步实现基于GPU的格子Boltzmann方法(LBM)计算。

7. 学位论文 [牛亚飞 列车模拟器视景仿真系统中实时阴影的生成研究 2009](#)

阴影的实时渲染作为增加虚拟场景真实感的关键技术之一,一直备受关注,它使得计算机产生的图像更逼真。阴影暗示了几何场景的信息,如物体形状、物体间的位置关系及光源的位置和方向等。在各类仿真器、大型三维游戏、地理信息系统和计算机辅助设计、制造中,许多成熟的阴影渲染技术得到了广泛的应用。然而,在动态场景中实时绘制真实感的阴影仍是一种挑战。

近年来,伴随着计算机图形硬件的快速发展,复杂的图形计算已经逐步从CPU转向图形硬件——图形处理器GPU。现代GPU技术的发展极大地推进了计算机实时渲染技术的应用,其主要的的应用包括交互式的图形建模程序和虚拟现实系统以及游戏、计算机动画等。基于图形硬件的实时阴影绘制的研究取得了很大的成果,利用可编程图形硬件加速并提高渲染效果已经成为主要发展趋势。

本文基于现代图形处理器的可编程特性,在吸取三维计算机图形学、计算几何、科学计算可视化、虚拟现实的先进理论和技术成果的基础上,对列车模拟器视景仿真系统中动态实时阴影进行了较为深入的研究和分析。通过对各种典型的实时阴影算法的优缺点进行比较,根据列车驾驶仿真器实时性要求较高的需求,针对不同的光源类型,最终选择了一种基于图像空间的算法进行扩展。在实验的基础上,对基于图像空间的算法作了改进,并在实际项目中得到了很好的应用。该算法较好地平衡了阴影真实感和实时性,实时交互性和阴影真实感效果基本满足了实际的需求。

8. 学位论文 [卜祥磊 基于GPU的医学图像三维可视化技术研究 2009](#)

直接体绘制能够探究体数据内部复杂的解剖结构,与标准的三维面绘制技术相比,该绘制方法最大的优势在于它能够提供更透明绘制,能够提供不同结构间丰富的空间信息。但现代医学影像设备产生的数据量非常大,这就对传统绘制架构和技术提出了极大的挑战。在三维可视化技术和计算机图形学中提到的体绘制算法中,有的为获取高质量的重建图像而放弃与体数据的交互,有的为获取良好的交互性能而牺牲图像的质量。随着当前图形硬件技术的发展,研究人员根据图形硬件架构的特殊结构设计了许多新的算法,这些算法能够兼顾图像质量和交互性能两个方面,对三维重建和可视化技术具有重要的意义。本文在学习图形硬件GPU编程技术的基础上,对医学图像的可视化技术做了一些有意义的探索。

1. 基于GPU的医学图像快速体绘制算法

文中利用图形处理器(GPU)强大的并行计算能力和灵活的可编程性能,将传统的光线投射算法在具有可编程管线的图形处理器上重新实现,将耗时的三线性插值和采样过程放在GPU上进行,从而提高了重建速度。论文首先将体数据和传递函数映射为纹理并将其载入到显存,接着通过对顶点着色程序和像素着色程序的编写将光线进入点、离开点的计算以及图像的合成运算移入GPU中,最后通过调整传递函数来实现不同的绘制效果。论文通过使用渲染到纹理技术,将绘制的中间结果保存到纹理,并以此来避免使用着色器的动态分支功能。实验表明,在同等绘制质量的前提下,该方法的绘制速度显著提高,能够满足医学影像可视化的实时交互需求,具有较好的临床应用前景。

2. 基于GPU的医学图像

为了给医生提供全面、直观和准确的诊断信息,论文提出了一种基于GPU加速的体切割算法,通过将切割算法和基于GPU的光线投射算法结合,实现体数据的快速切割。论文在基于GPU加速的医学图像快速体绘制的基础上,将剖面的空间信息传入着色器,然后通过比较体数据的空间坐标与剖面位置的关系来决定体数据的取舍。该方法不同于以往基于深度模板信息的体切割,在定义好切割平面后,可从任意角度对保留下来的有效体数据的重建结果进行观察。该方法能够精确地按照用户定义的形状对体数据进行切割,并且由于使用了硬件的加速功能,该方法可以达到实时交互的速度,在手术模拟等临床技术中有广泛应用。

3. 基于影响因子的医学图像快速体绘制算法

通过对体数据进行切割,医务人员能够清楚地观察到隐藏在体数据内部的重要器官和组织,但切割平面的确定比较困难,如果切割平面穿过感兴趣器官,则会切除部分感兴趣器官;如果切割平面恰好位于感兴趣器官前方,则部分感兴趣器官将被其他组织器官遮挡,无法清楚地再图像中显示。为了消除体切割算法的这种缺点,论文提出了一种基于影响因子的医学图像快速体绘制算法。论文在应用GPU进行加速的基础上,将影响因子引入到传递函数的构造中,对重要组织、感兴趣器官部分的体数据赋予大的影响因子,相反则为其赋予小的影响因子,通过对体数据影响因子的调节来达到增强重要组织、感兴趣器官抑制次要组织、非感兴趣器官的效果。另外论文通过对球形光照模型映射而来的纹理的索引来进行非真实感绘制,从而实现对艺术风格风格的模拟,增强绘制图像对物体重要特征和细节的绘制能力。该方法能够弥补传统切割算法的不足,在保证交互速度的前提下清楚地显示体数据内部的结构,为医务人员诊断提供尽可能多的信息。

9. 期刊论文 [张延红, 周必水 基于GPU加速的光线与三角面片求交 - 计算机时代2006, "" \(5\)](#)

近年来,随着图形处理器(GPU)能力以及可编程技术的发展,基于图形处理器的通用计算成为一个研究的热点.文章介绍了图形处理器的发展和结构,以及在使用通用计算时所面临的一些问题;具体介绍了利用GPU进行光线和空间三角形面片的求交测试。

10. 学位论文 [何晶 基于GPU的体绘制技术研究 2008](#)

医学图像三维可视化是科学可视化的一个重要研究方向,而直接体绘制技术作为医学图像可视化的关键技术近年来发展迅速,受到国内外学者的广泛关注。直接体绘制技术在绘制时不需要生成中间几何图元,而是根据离散的三维体数据标量值直接绘制成像,绘制过程充分利用了体数据标量值来获取全局信息。然而,体数据的数据量非常大,直接体绘制又涉及大量的插值运算和颜色混合运算,巨大的计算量对计算机的性能要求很高,绘制的速度较慢,限制了体绘制的广泛应用。目前所见到的一些体绘制加速技术,多半是以牺牲图像绘制质量为代价来实现加速的,加速性能和加速的扩展空间有限。因此,在保证图像质量的前提下,单纯依靠软件加速技术很难满足实时成像和交互帧率的需求。

当前,可编程图形处理器GPU发展迅速,它具有强大的并行计算和多数数据流处理能力,数据传输带宽不断提升,显存容量不断增大。较大的显存

容量使得中小规模甚至大规模的体数据可以一次性的以三维纹理形式载入显存，而直接体绘制中顶点和片段属性的相关计算非常适合GPU的并行流处理，尤其是GPU灵活的可编程特性为体绘制算法在图形硬件上的实现提供了可能。

本文对现有体绘制软硬件技术进行分析总结，比较了各种体绘制算法的优缺点，并根据图形硬件的可编程特性，提出一种基于GPU的快速光线投射算法。以光线投射算法为基础，用预积分分类法对体数据标量值进行分类转换，然后编写顶点程序和片段程序将光线进入点/离开点的计算和光线遍历的计算移入GPU 中执行，在通用PC的图形硬件上实现了基于GPU的快速光线投射体绘制算法。其关键之处在于能够利用GPU 的可编程特性在片段程序中完成对体数据的遍历、采样和计算，并将得到的采样值分类后进行混合，以产生最终的重建结果。其优势在于：避免了传统体绘制算法CPU 与GPU 之间的多次数据交换；通过计算出光线的离开点来避免动态分支；只需要绘制一个填充四边形即可完成光线投射计算。此外，通过预定义阈值，在片段中实时计算梯度，可以实现添加光照的等值面绘制。

论文最后对所提出的方法进行了实验验证。通过实验表明：本文算法充分利用了通用图形硬件灵活的纹理操作和强大的并行处理能力，解决了体数据的巨大计算量所造成的体绘制瓶颈问题，绘制速度和性能得到极大的提高，能够满足中小规模体数据实时交互的绘制需求。

## 引证文献(98条)

1. [王世元, 温柳英](#) GPU光线跟踪算法加速结构研究[期刊论文]-[技术与市场](#) 2010(5)
2. [过洁, 徐晓阳, 潘金贵](#) 基于阴影图的阴影生成算法研究现状[期刊论文]-[计算机辅助设计与图形学学报](#) 2010(4)
3. [刘晓平, 谢文军](#) 实时水面模拟方法研究[期刊论文]-[工程图学学报](#) 2010(1)
4. [徐少平, 李洋, 江顺亮, 熊宇虹, 叶发茂](#) 基于混合编程的真实感图形生成课件[期刊论文]-[计算机技术与发展](#) 2010(2)
5. [王宗跃, 马洪超, 徐宏根, 张建伟, 彭检贵](#) 多核CPU的海量点云并行kNN算法[期刊论文]-[测绘科学技术学报](#) 2010(1)
6. [王建明, 张树斌](#) 浅谈GPU在遥感影像融合中的应用[期刊论文]-[太原科技](#) 2010(1)
7. [丁鹏, 陈利学, 龚捷, 张岩](#) GPU通用计算研究[期刊论文]-[计算机与现代化](#) 2010(1)
8. [叶剑, 李立新](#) 基于GPU的AES快速实现[期刊论文]-[计算机工程与设计](#) 2010(2)
9. [邹岩, 杨志义, 张凯龙](#) CUDA并程序序的内存访问优化技术研究[期刊论文]-[计算机测量与控制](#) 2009(12)
10. [刘世光, 柴佳伟, 闻媛](#) 三维动态云快速模拟的新方法[期刊论文]-[计算机研究与发展](#) 2009(9)
11. [韩博, 周秉锋](#) GPGPU性能模型及应用实例分析[期刊论文]-[计算机辅助设计与图形学学报](#) 2009(9)
12. [张静, 陈晓军, 龚捷, 陈汶滨, 陈亚丽](#) 阴影图算法的改进与实现[期刊论文]-[黑龙江科技信息](#) 2009(29)
13. [崔雪冰, 张延红, 李国徽](#) 基于通用计算的GPU-CPU协作计算模式研究[期刊论文]-[微电子学与计算机](#) 2009(8)
14. [盛玲, 姜晓彤](#) 基于Cg高级着色器语言的阴影贴图的算法研究[期刊论文]-[中国科技信息](#) 2009(15)
15. [陈红倩, 李凤霞, 黄天羽, 战守义](#) 一种基于动态纹理的运动场景可视化方法[期刊论文]-[北京理工大学学报](#) 2009(6)
16. [王祥远, 王兴东, 宋利](#) DVCPRO HD并行解码算法的研究与实现[期刊论文]-[信息技术](#) 2009(7)
17. [张健, 陈瑞](#) 图形处理器在通用计算中的应用[期刊论文]-[计算机工程与设计](#) 2009(14)
18. [徐品, 蓝善祯, 刘兰兰](#) 利用GPU进行通用数值计算的研究[期刊论文]-[中国传媒大学学报\(自然科学版\)](#) 2009(2)
19. [宋晓丽, 王庆](#) 基于GPGPU的数字图像并行化预处理[期刊论文]-[计算机测量与控制](#) 2009(6)
20. [崔雪冰, 张延红, 王康平](#) 基于GPU的通用计算模型[期刊论文]-[河南科技大学学报\(自然科学版\)](#) 2009(3)
21. [王磊, 张春燕](#) 基于图形处理器的通用计算模式[期刊论文]-[计算机应用研究](#) 2009(6)
22. [胡香, 张巍巍, 周玉柱, 刘芳](#) 利用GPU实现基于物理模型的流体运动仿真[期刊论文]-[测绘科学技术学报](#) 2009(3)
23. [高辉, 张茂军, 熊志辉](#) 基于GPU编程的地形纹理快速渲染方法研究[期刊论文]-[小型微型计算机系统](#) 2009(4)
24. [倪炜](#) GPU通用计算研究[期刊论文]-[黑龙江科技信息](#) 2009(16)
25. [苏畅, 付忠良, 谭雨辰](#) 一种在GPU上高精度大型矩阵快速运算的实现[期刊论文]-[计算机应用](#) 2009(4)
26. [章浩, 庞振山, 姚长利, 张明华](#) 基于CUDA技术的矿产储量计算[期刊论文]-[地质通报](#) 2009(2)
27. [赵健, 徐凯, 吴玲达](#) 微分域网格变形的GPU加速算法[期刊论文]-[小型微型计算机系统](#) 2009(3)
28. [张健, 陈瑞, 芮雄丽](#) 安全散列算法SHA-1在图形处理器上的实现[期刊论文]-[网络安全技术与应用](#) 2009(1)
29. [薛盖超, 吕伟伟, 刘学慧](#) 方差阴影图中的光渗现象消除算法[期刊论文]-[计算机辅助设计与图形学学报](#) 2009(2)
30. [罗月童, 薛晔, 刘晓平](#) 基于GPU的多分辨率体数据重构和渲染[期刊论文]-[计算机辅助设计与图形学学报](#) 2009(1)



31. [邱书波, 吕荫平](#) [基于DSP的纸张缺陷实时检测系统设计](#) [期刊论文] - [中国造纸学报](#) 2008 (4)
32. [闾跃龙, 贾金原, 刘金义](#) [基于GPU加速的TIP技术](#) [期刊论文] - [计算机辅助工程](#) 2008 (4)
33. [杨娜, 杨钦, 曹龙](#) [基于GPU的加速网格求交算法分析与实现](#) [期刊论文] - [微计算机信息](#) 2008 (30)
34. [李海燕, 张春元, 李礼, 任巨](#) [图形处理器的流执行模型](#) [期刊论文] - [计算机工程](#) 2008 (22)
35. [季卓尔, 张景峤](#) [基于可编程图形处理器的骨骼动画算法及其比较](#) [期刊论文] - [计算机工程与设计](#) 2008 (21)
36. [李伟伟, 王健, 陈轶, 王钺旋](#) [火焰实时模拟的新算法](#) [期刊论文] - [吉林大学学报 \(信息科学版\)](#) 2008 (6)
37. [吴玲达, 杨超, 陈鹏](#) [基于GPU的等值面提取与绘制](#) [期刊论文] - [计算机应用研究](#) 2008 (11)
38. [李蔚清, 吴慧中](#) [一种基于GPU的雷达探测区域快速可视化方法](#) [期刊论文] - [系统仿真学报](#) 2008 (z1)
39. [杨珂, 罗琼, 石教英](#) [平行散点图: 基于GPU的可视化分析方法](#) [期刊论文] - [计算机辅助设计与图形学学报](#) 2008 (9)
40. [曹问鱼, 杨耀明, 汤晓安](#) [港口三维GIS系统的设计与关键技术研究](#) [期刊论文] - [科技信息 \(学术版\)](#) 2008 (12)
41. [CHEN Peng, 杨超, WU Ling-da](#) [硬件加速的等值面提取与绘制](#) [期刊论文] - [小型微型计算机系统](#) 2008 (8)
42. [CHEN Peng, 杨超, WU Ling-da](#) [硬件加速的等值面提取与绘制](#) [期刊论文] - [小型微型计算机系统](#) 2008 (8)
43. [季卓尔, 张景峤](#) [基于可编程GPU的骨骼动画](#) [期刊论文] - [计算机工程与应用](#) 2008 (22)
44. [程甜甜, 陈阁, 陈新, 李云飞](#) [基于GPU的大规模地形场景实时渲染](#) [期刊论文] - [苏州大学学报 \(工科版\)](#) 2008 (3)
45. [周季夫, 钟诚文, 尹世群, 解建飞, 张勇](#) [基于GPGPU的Lattice-Boltzmann数值模拟算法](#) [期刊论文] - [计算机辅助设计与图形学学报](#) 2008 (7)
46. [包达, 罗立民](#) [基于可编程图形处理器的快速傅立叶变换](#) [期刊论文] - [生物医学工程研究](#) 2008 (2)
47. [张雨浓, 马伟木, 李克讷, 易称福](#) [简述协处理器发展历程及前景展望](#) [期刊论文] - [中国科技信息](#) 2008 (13)
48. [刘耀林, 邱飞岳, 王丽萍](#) [基于GPU的图像快速旋转算法的研究及实现](#) [期刊论文] - [计算机工程与科学](#) 2008 (6)
49. [张立民, 邓向阳, 赵瑞行](#) [通用加速计算在环幕失真校正中的应用研究](#) [期刊论文] - [海军航空工程学院学报](#) 2008 (3)
50. [史胜伟, 姜昱明, 朱新蕾](#) [基于GPU的真实感地形绘制](#) [期刊论文] - [计算机系统应用](#) 2008 (6)
51. [史胜伟, 姜昱明, 朱新蕾](#) [基于GPU的真实感地形绘制](#) [期刊论文] - [电脑开发与应用](#) 2008 (6)
52. [邓向阳, 张立民, 曹新建](#) [基于硬件加速的几何失真校正研究](#) [期刊论文] - [现代计算机 \(专业版\)](#) 2008 (3)
53. [王家华, 刘琳](#) [油藏建模三维显示的交互式设计与研究——基于GPGPU技术](#) [期刊论文] - [科技资讯](#) 2008 (2)
54. [陈鹏, 魏迎梅, 吴玲达, 杨超](#) [硬件加速的雷达作用范围三维可视化研究与实现](#) [期刊论文] - [计算机工程与科学](#) 2008 (4)
55. [徐少平, 文喜, 肖建, 曾文](#) [一种基于Cg语言在图形处理器GPU上实现加密的方法](#) [期刊论文] - [计算机应用与软件](#) 2008 (4)
56. [王磊, 杨新, 朱磊](#) [基于图形硬件加速的Freehand三维超声图像滤波与插值技术](#) [期刊论文] - [微型电脑应用](#) 2008 (3)
57. [李建明, 万单领, 何荣盛, 钱昆明](#) [一种基于GPU加速的图像颜色传递算法](#) [期刊论文] - [大连理工大学学报](#) 2008 (2)
58. [曲洋, 黄永忠, 王磊](#) [流式缩减技术在GPU上的研究与应用](#) [期刊论文] - [计算机工程与设计](#) 2008 (5)
59. [刘世光, 彭群生](#) [气象景观的真实感模拟技术综述](#) [期刊论文] - [计算机辅助设计与图形学学报](#) 2008 (4)
60. [郑浩, 柴学梁, 王毅刚](#) [三维家具虚拟展示系统的设计与实现](#) [期刊论文] - [中国水运 \(学术版\)](#) 2008 (1)
61. [李勋祥, 陈定方](#) [分布式汽车驾驶交互仿真系统的实现](#) [期刊论文] - [计算机辅助设计与图形学学报](#) 2008 (1)
62. [张经宇](#) [用图形处理器 \(GPU\) 实现矩阵乘法的方法](#) [期刊论文] - [职大学报](#) 2007 (4)
63. [孟亮, 刘传耀](#) [LOD算法中的高度预测](#) [期刊论文] - [科技情报开发与经济](#) 2007 (36)
64. [李大禹, 胡立发, 穆全全, 宣丽](#) [GPU计算液晶自适应光学波前重构的并行性研究](#) [期刊论文] - [液晶与显示](#) 2007 (5)



65. [张怡, 张加万, 孙济洲, 柯永振](#) [基于可编程图形加速硬件的实时光线投射算法](#)[期刊论文]-[系统仿真学报](#) 2007(18)
66. [王华, 朱丽华, 顾耀林](#) [一种基于阴影图的实时软阴影算法](#)[期刊论文]-[计算机应用](#) 2007(10)
67. [唐兆, 邬平波](#) [三维实时云建模与渲染在工业仿真中的应用](#)[期刊论文]-[计算机辅助设计与图形学学报](#) 2007(8)
68. [杨慧, 鲍旭东](#) [基于硬件加速体绘制的切割方法](#)[期刊论文]-[生物医学工程研究](#) 2007(2)
69. [李起成, 陈昊罡, 汪国平, 董士海](#) [动态天空环境下的实时海洋渲染](#)[期刊论文]-[计算机辅助设计与图形学学报](#) 2007(2)
70. [黄鑫, 李胜, 程惠阁, 汪国平](#) [基于GPU的B样条曲面加速计算](#)[期刊论文]-[系统仿真学报](#) 2006(z1)
71. [李蔚清, 苏智勇, 杨正龙, 吴慧中](#) [一种基于GPU的复杂目标电磁散射快速算法](#)[期刊论文]-[系统仿真学报](#) 2006(8)
72. [李宏海, 肖建海](#) [CPU+GPU技术在非编系统中的应用](#)[期刊论文]-[现代电视技术](#) 2006(6)
73. [柳有权, 刘学慧, 吴恩华](#) [基于GPU带有复杂边界的三维实时流体模拟](#)[期刊论文]-[软件学报](#) 2006(3)
74. [柳有权, 刘学慧, 吴恩华](#) [基于GPU带有复杂边界的三维实时流体模拟](#)[期刊论文]-[软件学报](#) 2006(3)
75. [梁亮, 张定华, 毛海鹏, 顾娟](#) [一种基于可编程图形硬件的快速三维图像重建算法](#)[期刊论文]-[计算机应用研究](#) 2006(1)
76. [张庆丹, 戴正华, 冯圣中, 孙凝晖](#) [基于GPU的串匹配算法研究](#)[期刊论文]-[计算机应用](#) 2006(7)
77. [冯中心, 唐敏, 董金祥](#) [卡通风格雨的实时模拟](#)[期刊论文]-[计算机辅助设计与图形学学报](#) 2006(12)
78. [赵乃良, 陈艳军, 潘志庚](#) [基于数据修正的实时阴影反走样算法](#)[期刊论文]-[计算机辅助设计与图形学学报](#) 2006(8)
79. [李笑盈, 吴恩华](#) [过程性纹理映射的FPGA动态生成](#)[期刊论文]-[计算机辅助设计与图形学学报](#) 2006(5)
80. [孔渊, 陆虎敏, 周坚锋, 郭凡](#) [计算机图形系统发展简述](#)[期刊论文]-[航空电子技术](#) 2006(2)
81. [郝立巍, 陈武凡](#) [医学三维动态超声实时体绘制](#)[期刊论文]-[南方医科大学学报](#) 2006(3)
82. [吴宇钦, 张丽, 陈志强](#) [基于GPU的锥束CT体数据等值面重构和显示的改进](#)[期刊论文]-[CT理论与应用研究](#) 2006(4)
83. [刘伟峰](#) [基于Internet的三维实时图形引擎的关键技术研究](#)[学位论文]硕士 2006
84. [樊翠](#) [三维游戏引擎的设计及关键技术的实现](#)[学位论文]硕士 2006
85. [甘小方](#) [基于GPU的颜色传递算法在视频处理中的应用](#)[学位论文]硕士 2006
86. [张应利](#) [一种基于GPU加速的二维图像铅笔化算法](#)[学位论文]硕士 2006
87. [张明](#) [GPU加速的实时三维海洋漫游系统](#)[学位论文]硕士 2006
88. [戴康](#) [自然晶体光学现象的实时计算机图形学研究](#)[学位论文]硕士 2006
89. [苏智勇](#) [基于GPU的实时绘制及其应用](#)[学位论文]硕士 2006
90. [沈潇](#) [三维场景实时阴影算法的研究](#)[学位论文]硕士 2006
91. [张杨, 诸昌铃, 何太军](#) [图形硬件通用计算技术的应用研究](#)[期刊论文]-[计算机应用](#) 2005(9)
92. [杨正平](#) [基于GPU计算的直接体视化和遗传算法研究](#)[学位论文]硕士 2005
93. [于守秋](#) [基于GPU的实时视频艺术风格化系统](#)[学位论文]硕士 2005
94. [梁亮](#) [基于可编程图形硬件的三维图像快速重建算法研究](#)[学位论文]硕士 2005
95. [董朝](#) [基于可编程图形硬件加速的若干技术研究](#)[学位论文]硕士 2005
96. [房波](#) [基于通用可编程GPU的视频编解码器——架构、算法与实现](#)[学位论文]硕士 2005
97. [李海](#) [基于硬件加速的虚拟场景绘制](#)[学位论文]硕士 2005
98. [吴恩华](#) [图形处理器用于通用计算的技术、现状及其挑战](#)[期刊论文]-[软件学报](#) 2004(10)

本文链接: [http://d.g.wanfangdata.com.cn/Periodical\\_jsjfsjytxxb200405002.aspx](http://d.g.wanfangdata.com.cn/Periodical_jsjfsjytxxb200405002.aspx)

授权使用: 西安电子科技大学(xadzkj), 授权号: 1715e8a1-3985-4a79-9d34-9e17017e42a1

下载时间: 2010年10月22日