基于PC硬件的大规模数据场快速可视化方法

陈世浩[1,2] 郝重阳[1]

(西北工业大学电子信息学院 西安 710072)[1] (慕尼黑工业大学计算机学院 德国慕尼黑 85748)[2]

摘 要 科学可视化技术在众多领域具有十分广泛的应用,然而直接体绘制技术却有着计算量大、计算时间长的缺点,在普通的PC机上很难实现对大规模数据的实时交互绘制。目前的三维可视化系统通常需要架构在高端的图形工作站或转用计算机上。随着计算机软硬件技术的发展,普通的PC机图形处理器GPU(Graphic Processing Unit)具有了可编程功能。正是借助GPU的可编程功能及其强大的并行处理能力,研究并实现了一种基于普通PC硬件的体会之方法。最后应用该方法对工业、医学等体数据进行可视化,结果证明该方法可以在普通PC上实现较大规模数据的快速可视化。

关键词 图形处理器GPU(Graphic Processing Unit),大规模数据场,光线投射算法

中图法分类号 TP391 **文献标识码** A

Rapid PC Hardware Based Visualization Method for Large-scale Datasets

CHEN Shi-hao[1,2] HAO Chong-yang[2]

(Institute of Electronic and Information Engineering, Northwestern Polytechnical University, Xi'an 710072, China)[1]

(Technical University Munich, Munich 85748, Germany)[2]

Abstract As the 3D datasets are usually in large scalar, the capability of a single CPU to rendering is not sufficient to achieve interactivity. Direct volume rendering via GPU has positioned itself as an efficient tool for the display and visual analysis of volumetric scalar fields. A rapid PC hardware based visualization method for large-scale datasets method was proposed. At last we demonstrated the effectiveness of our method with several data sets. It was proved that the proposed method can generate high-quality visual representations on normal PC.

Keywords GPU(Graphic Processing Unit), Large-scale datasets, Ray casting

大规模数据场的可视化一直是科学计算可视化中极具挑战性的一个研究领域。而直接体绘制技术是三维数据场可视化的主要技术手段之一,它不仅能够表现出三维对象的表面特征,也能显示其内部信息,因此被广泛地应用于三维数据场的可视化中。然而,直接体绘制技术却有着计算量大、计算时间长的缺点。实际生产过程中的三维数据场的数据量往往十分巨大,目前对大规模体数据需要应用高性能计算机进行交互式可视化以达到实时显示的效果。为了能够快速可视化大规模体数据场,以往的研究主要集中在基于高端图形工作站和计算机集群上的并行算法。近年来个人计算机的性能大幅提高,特别是可编程图形硬件飞速发展,利用PC机可视化大规模体数据场成为可能。

1 PC图形处理器

直接体绘制的计算量很大,尽管已经提出多种加速算法,但仍然很难在普通工作站和PC机上实现中等规模数据的实时可视化,只能利用高档图形工作站提供的由硬件实现的3D纹理映射功能来进行体数据的直接体绘制,如SGI的Reality Engine。近年来PC图形处理器GPU(Graphics Processing Unit)[1]发展很快,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中央处理器(CPU)计算能力的增长速度。主流图形硬件制造商声称,现在每隔12个月GPU的性能就会增长一倍。图形硬件技术一个最主要的突破就是在图形硬件中引入了可编程功能,此功能允许用户编制自定义的着色器程序(Shader program)来替换原来固定流水线中的某些功能模块,使得GPU在功能上更像一个通用处理器。虽然GPU具有非常高的计算速度,但并不能直接将以前在CPU中实现的算法照搬到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,这种流处理机的并行结构实现了指令的并行处理。目前绝大部分的GPU都拥有多条可以并行的Shader管线,这种体系结构使得其不仅可以用于高效图形绘制,而且可以成为通用并行计算平台。所以要基于可编程图形硬件实现一些在CPU中效率较低的算法,就必须重新组织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。图形硬件处理管道的顶点处理和像素处理模块中引入了可编程性,使得用户可以通过程序方式控制图形流水线的执行,从而极大地扩

到稿日期:2009-07-01 返修日期:2009-09-04 本文受国家教育部博士点基金(20040699015)资助。

陈世浩(1979-),男,博士生,主要研究方向为基于GPU的通用计算、图像处理等,E-mail:peternpu@hotmail.com;郝重阳(1942-),男,教授,主要研究方向为图像处理等。

展了图形处理器的能力和应用范围。在顶点级操作上，引入了顶点着色器(vertex shader)来处理每个顶点，用户可以自己编写代码来实现专门的光照明模型。经过光栅化后，在像素处理阶段引入的像素着色器(pixel shader)可以实现对每个像素的可编程操作。

2 基于 GPU 的直接体绘制

三维显示可以通过多种算法来实现，主要可以分为表面绘制技术[2,3]和体积绘制技术[4]。表面绘制技术通过数据分析寻找感兴趣区域的轮廓线边界，进而绘制出多边形表面。体积绘制技术包括等 Multi-planar Reformatting 算法、Splatting 算法[5]、Ray Casting 算法[6]及硬件加速三维纹理算法[7,8]等。较之表面绘制技术，体积绘制能够高质量地绘制出三维物体的表面以及内部细节，但运算量较大，实时交互性较差。

2.1 光线投射算法

图像空间的体绘制算法以 Ray Casting 算法为代表。其原理是沿视线方向向物体投射一条光线，在它穿越体数据的过程中，每隔一段距离便执行一次采样。每个采样点的数值要通过邻近的离散体素点的插值来产生。在执行完插值和采样操作之后，可以按照从前向后或从后向前的顺序对光线通路上的采样结果进行积分，形成对应于某种物理或几何性质的图像。完整的 Ray Casting 算法需要考虑较多光学效应，且由于人体解剖形态的复杂性，这种算法的运算量很大，利用软件运算很难达到实时交互性。由于插值和积分这两步大运算量的操作可以利用现代图形硬件的特性来高速完成，硬件加速三维纹理算法便成为 PC 平台下高质量实时体绘制的较佳选择。我们将 CT，MRI 等医学影像以纹理形式载入显存后，使用代理几何面对插值之后的纹理图像进行采样，采样产生的切片可以通过透明度混合的方法绘制，从而在屏幕上显示出三维物体。

在体积绘制中最重要和最大的工作量就是采样和插值计算。体积数据本身就是一种采样后得到的离散数据，这个过程也称为重采样。为了绘制出一幅高质量的体积图像，这些重采样点的对应数值需要仔细地计算。把体积数据作为三维纹理，可以利用图形硬件的纹理插值功能，快速得到重采样点的数值。

在目前通用显示硬件上，图形卡的显存空间已经达到 768MB 以上，三维纹理存放在显存上，大大提高了读写速度。Direct X10 提供了许多对三维纹理操作的功能，本文使用 Direct X10 三维纹理映射函数，用代理几何面获取经过插值的三维纹理中的体数据。如图 1 所示，代理几何面取与视线方向垂直的等分三维纹理空间的平面切片，可以通过改变中间平面的层数来改变成像的精细程度。直接体绘制积分公式近似为

$$C(X_{s_i})=\begin{cases} f_C(X_{s_i})f_a(X_{s_i}) \\ C(X_{s_{i-1}})+f_c(X_{s_i})f_a(X_{s_i})(1-a(X_{s_{i-1}})),i>0 \end{cases} \tag{1}$$

$$a(X_{s_i})=\begin{cases} f_a(X_{s_i}),i=0 \\ a(X_{s_{i-1}})+f_a(X_{s_i})(1-a(X_{s_{i-1}})),i>0 \end{cases} \tag{2}$$

式中，C_{s_i} 和 a_{s_i} 分别表示 s_i 点的颜色值与不透明度值，$f_c(X_{s_i})$ 和 $f_a(X_{s_i})$ 是颜色和灰度传递函数。$C(X_{s_n})$ 和 $a(X_{s_n})$ 分别表示最终的颜色值与透明度值。从前到后的投射方法需要使用一个 α-buffer 来存储不透明度累积值，以便当不透明度达到一定值时可以判断放弃对该条光线的继续计算，以减少计算量。

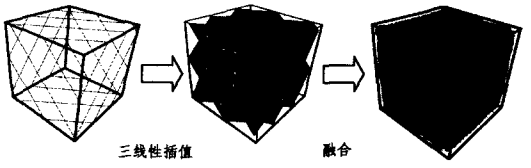

图 1 基于三维纹理的体绘制

对平面从后向前绘制的过程实际是当前平面提取的纹理图像与缓存中已经绘制好的图像相融合的过程。如果只取某个域值上的点，就构成了等值面的显示；如果采用前后累加，就相当于光线跟踪中对光线上的采样点的积分。

2.2 光照模型

为了增强可视化效果，需要引入光照模型。本文采用 Phong 光照模型：

$$I=I_a+k_dI_p(n\cdot l)+k_sI_p(m\cdot h)^n \tag{3}$$

式中，I_a 是环境光，I_p 是点光源强度，n 为绘制面法线，l 为入射光方向，k_d 是漫反射系数，k_s 镜面反射系数，m 为镜面高光系数。其中 n 用体数据的梯度替代。

3 算法在 PC 上的实现

目前的显卡可以支持 DirectX Pixel Shader4.0 的渲染技术。这些技术可以支持同时对多个纹理的读写操作；支持不同通道间的通信；纹理坐标生成以及代数运算等的支持，可以使我们在 GPU 上实现多通道(Multi passes)的并行光线投射算法。

首先需要绘制代理几何面[9]来辅助绘制，其目的是通过设计恰当的代理几何面直接把大多数外围的空区域隔开，减少光线的数量以及采样长度。这里选择立方体为代理几何面。

代理几何面的前面、后面的每个像素点的几何信息分别存储于该点的颜色通道中，这样只需要这两个面做减法运算，就可以得到光线向量。光线投射法按照以下的步骤实现：

1)绘制代理面的前面，将其存储于一个纹理中；

2)绘制代理面后面，对后面和前面做减法运算，再对结果单位化并存储于一个新的纹理中；

3)对体数据应用光线投射算法：重新绘制代理面前面，沿着光线方向(光线向量)进行采样计算，直到光线离开代理面或者不透明度达到一定值；

4)最后将这些结果合成以得到最后的图像。混合方程为

$$C_{out,i}=C_{out,i-1}+a_i(1-A_{i-1})C_{in,i} \tag{4}$$

$$A_i=A_{i-1}+a(1-A_{i-1}) \tag{5}$$

式中，A_i 是累积不透明度，a_i 为采样分配的不透明度，$C_{in,i}$ 为采样点的颜色值，$C_{out,i}$ 为混合后的累积颜色。

可以看出以上的过程非常适合现代 GPU 的处理，可以将代理面的前、后面分别存储于一个纹理中。各个步骤所得到的结果也可以存储于纹理中。由于现在 GPU 的渲染程序中 Shader 可以高速访问多个纹理，因此这种方法过程中不需要与 CPU 通信，便可大大提高运算速度。其次，当今 GPU 支

持循环与条件选择，所以可以很简单地实现这些描述的方法。

4 实验与分析

将本文提出的算法应用于大规模医学、工业等数据，首先给出绘制图像，然后以标格形式列出绘制速度以及数据的规模，验证本文方法的效能。

实验条件：使用一台普通的 PC 作为平台，具体如下：Intel(R) Core(TM)2 CPU 6600@ 2.40GHZ 2.39GHZ 配备 GeForce 8800 GTX GPU，768 MB 显存。支持 DirectX4.0，整个系统在 VC++环境下运行，使用 HLSL 语言编写。

图 2 给出了绘制的结果，其中 A 是战斗机喷气发动机尾部喷气的三维数据，使用本文方法对其进行了体绘制；B 为一条冷冻鱼的 CT 扫描三维数据集的绘制结果；C 为 Teddy 熊的三维 CT 扫描数据集所绘制的结果；D 为一块云彩的三位扫描数据的体绘制图像。各个数据集的规模在表 1 中列出。

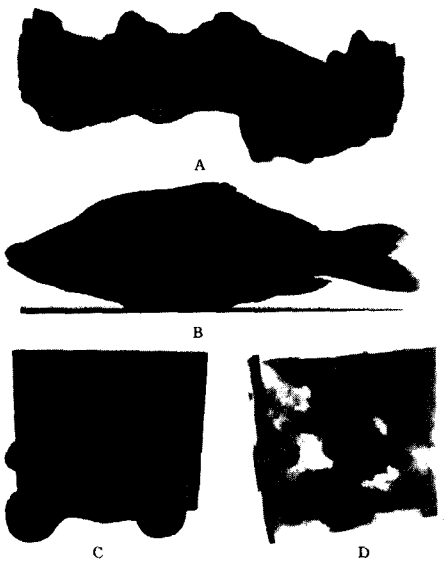

图 2 数据体绘制结果

从图 2 可以看出本文的方法可以很好地应用于多种类型的体数据，绘制结果较为理想。

从表 1 的结果可以看出，当数据集大小为 512×512×512 时，本文的算法仍然可以在普通 PC 上以 16.7fps 的速度进行绘制，这个速度足以满足实时交互绘制的需求。绘制结果图像和绘制速度都验证了本文所提方法的有效性。

表 1 绘制速度(帧/s)

数据集	A	B	C	D
数据规模	256×256×256	512×512×512	128×128×62	128×128×128
视窗尺寸	1264×958	1024×968	1264×958	1264×958
绘制速度(fps)	18.34	16.7	65.78	109.4

结束语 本文研究并实现了一个基于普通 PC 平台的大规模数据快速可视化算法。算法是基于三维纹理硬件的直接体绘制算法，并在第 3 节中详细描述了算法具体实现的步骤。实验结果证明了算法的有效性。算法可以应用于多种类型的体数据。在普通 PC 平台上就可以实现大规模数据快速可视化。

参 考 文 献

[1] 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，15(10)：1493-1504

[2] Lorensen W E, Cline H E. Marching Cube: A High Resolution 3D Surface Construction Algorithm [J]. Computer Graphics, 1987, 21(4)

[3] Westover L. Footprint Evaluation for Volume Rendering [J]. Computer Graphics, 1990, 24(4): 367-376

[4] Mueller K, Shareef N, Huang J, et al. High quality splatting on rectilinear grids with efficient culling of occluded voxels [J]. IEEE Trans Vis and Comp Graph, 1999, 5(2): 116-134

[5] Tuy H K, et al. Direct 2-D Display of 3-D objects [J]. IEEE Computer Graphics and Applications, 1984, 4(10): 29-33

[6] Levoy M. Display of Surfaces from Volume Data[J]. IEEE Computer Graphics & Applications, 1988, 8(5): 29-37

[7] Cullip T, Neumann U. Accelerating volume reconstruction with 3D texture hardware[R]. UNC Technical Report. 1993

[8] Cabral B, Cam N, Foran J. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware[C]// Proceedings of the 1994 Symposium on Volume Visualization. 1994: 91-98

[9] Kruger J, Westermann R. Acceleration techniques for GPU-based volume rendering[C]// Visualization, VIS 2003. IEEE. 2003: 287-292

(上接第 205 页)

程往往也是被误导的。通过对优质候选解的重要成分进行识别，然后基于重要解成分更新信息素，使得信息素更能反映优质解的特点，消除欺骗吸引子的影响，最终找到全局最优解。

参 考 文 献

[1] Dorigo M, Socha K. An Introduction to Ant Colony Optimization [R]. IRIDIA/2006-10. Université Libre de Bruxelles, Belgium, 2006

[2] Dorigo M, Stützle T. Ant Colony Optimization [M]. London: MIT Press, 2004

[3] Stützle T, Hoos H. Max-min ant system[J]. Future Generation Computer Systems, 2000, 16(9): 889-914

[4] 朱庆保，杨志军. 基于变异和动态信息素更新的蚁群优化算法[J]. 软件学报，2004，15(2)：185-192

[5] 柯良军，冯祖仁，冯远静. 有限级信息素蚁群算法[J]. 自动化学报，2006，32(2)：296-303

[6] Ding J, Tang W, Wang L. Parallel Combination of Genetic Algorithm and Ant Algorithm Based on Dynamic K-Means Cluster [C]// Lecture Notes in Artificial Intelligence. Vol. 4114, Berlin: Springer, 2006: 825-830

[7] 丁建立，陈增强，袁著祉. 遗传算法与蚂蚁算法的融合[J]. 计算机研究与发展，2003，40(9)：1351-1356

[8] Blum C, Dorigo M. Deception in Ant Colony Optimization[C]// Lecture Notes in Artificial Intelligence. Vol. 3172, Berlin: Springer, 2004: 118-129

[9] Eiben A E, Smith J E. Introduction to evolutionary computing [M]. Berlin Heidelberg: Springer- Verlag, 2003: 191-193