基于动态纹理载入的大规模数据场体绘制

郑 杰 姬红兵
(西安电子科技大学电子工程学院, 西安 710071)

摘 要 为克服图形硬件对传统纹理映射体绘制的限制, 提出了一种在普通 PC上进行大规模数据场体绘制的有效方法。该方法中, 体数据被划分为合适大小的数据块, 这些数据块被动态的载入图形硬件, 并利用 3维纹理映射进行绘制。在整个绘制过程中, 仅有一个数据块存储在图形硬件上, 有效地提高了对大规模体数据的绘制能力。同时, 充分利用目前 PC图形硬件成熟的可编程特性, 通过对梯度的实时计算来减少在传统纹理映射体绘制中巨大的内存消耗。实验结果表明, 该方法在普通 PC上可以对超过纹理内存容量的大规模体数据进行交互式体绘制。

关键词 体绘制 3维纹理映射 纹理分块 图形处理单元
中图法分类号: TP391. 41 **文献标识码**: A **文章编号**: 1006-8961(2008)02-0316-06

Interactive PC Texture-based Volume Rendering for Large Datasets

ZHENG Jie, JI Hong-bing
(*School of Electronic Engineering, Xidian University, Xi'an* 710071)

Abstract A novel technique is presented for rendering large-scale volume datasets interactively on general purpose PC hardware. To circumvent the limited texture memory for texture based volume rendering, the dataset is partitioned into the bricks with reasonable size. The bricks are loaded to the graphics hardware dynamically and rendered using 3D texture mapping. During the rendering only one brick resides on the texture memory. Additionally, the sophisticated PC graphics hardware functionality is utilized to estimate the gradient on the fly avoiding the huge memory consumption in previous approaches. Using a prototype implementation of the algorithm, we are able to perform fast data loading and interactive visualization for the large datasets on a single standard PC.

Keywords volume rendering, 3D texture mapping, texture partitioning, graphics process unit

1 引 言

体绘制对于由各种类型的 3维设备、科学仿真等产生的体数据的可视化是一种非常重要的技术。为了克服绘制中大量的计算和对带宽的极高要求, 基于图形硬件纹理映射的体绘制逐步成为对规则网格数据场绘制的最为实际可行的方法。在体绘制中, 体数据作为 3维纹理被载入图形硬件, 在用切片对纹理采样时, 利用图形硬件的高速并行处理能力进行 3次线形插值, 对规模适中的体数据进行可交互的绘制并得到较高的图像质量。

GPU(graphics processing unit)在 1999年首先由 NVIDIA公司提出, 被定义为“一个单芯片的处理器, 集成了几何变换、光照、三角形构造、裁剪和绘制引擎功能, 并具有每秒至少 1千万个多边形的处理能力”。近几年来, GPU得到了高速的发展, 目前最新的通用图形处理器, 如 NVIDIA公司的 GeForce 7950, 具有每秒可绘制 20亿个顶点, 填充 240亿个像素的处理能力, 在运算速度、图像质量和可编程性方面甚至都超过了专业的图形工作站, 可以对现实世界中的各种材质和光照效果进行更真实的建模仿

收稿日期: 2006-03-17; 改回日期: 2006-08-22

第一作者简介: 郑杰(1976~), 男。现为西安电子科技大学电子工程学院电路与系统专业博士研究生。主要研究领域为图像处理、计算机图形学、科学可视化等。E-mail: zhengjie_ee@yahoo.com.cn

真。相对于以前采用固定绘制管道的图形硬件, 新的图形处理器在顶点处理和像素处理阶段提供了灵活的可编程特性, 在运算上都支持 IEEE32位浮点运算, 并且支持相关纹理采样功能, 可以将纹理作为内存来使用, 以方便数据的索引访问。这些新特征的引入使得大规模数据场在普通 PC上的实时绘制成为可能。

尽管目前的 PC图形硬件广泛提供了对 3维纹理的支持, 但基于 3维纹理映射的体绘制仍有一些局限性。当使用纹理映射对体数据进行绘制时, 硬件没有提供对梯度计算的直接支持。为了克服这个缺陷, 梯度数据要预先进行计算, 并和体数据一同载入图形硬件中。这就需要 4倍于体数据量的纹理显存来完成绘制。另一方面, 由于目前通用图形硬件有限的纹理内存容量及设计上的限制, 如 NVIDIA 的 GeForce系列图形硬件[1]最大支持 512×512×512的 3维纹理, 这就制约了大规模体数据的体绘制。在医学领域中, 随着多排螺旋 CT的广泛应用, 在一次扫描过程中可以产生多达 1024层的体数据, 仅体数据就需要 256MB的纹理内存空间, 传统的基于纹理映射的体绘制是无法满足这样的应用。

本文基于 PC图形硬件提出了一种对大规模数据场的体绘制算法。在绘制前, 将体数据预先划分为大小合适的数据块, 通过动态纹理载入技术对每个数据块进行绘制。在整个体绘制过程中, 纹理内存仅存储一个数据块, 有效地提高了对大规模体数据的绘制能力。同时, 利用目前图形硬件的可编程特性提出了一种梯度的实时计算方法, 避免了梯度的预先计算和绘制时的存储, 进一步减少了纹理内存的开销, 也加快了数据载入的速度。这样在普通 PC硬件上实现了对大规模体数据的交互体绘制。

2　相关研究

最重要的基于纹理映射的体绘制方法[2]由 Cabral等人提出, 其利用了高端图形工作站中的 3维纹理映射特性。近年来通用图形硬件的快速发展可以达到实时的 2维、3维纹理映射, 基于硬件加速的纹理映射逐渐成为进行逼真光照计算以及其他体绘制技术的重要手段。然而, 纹理内存容量是制约纹理映射技术应用的主要瓶颈. 在利用 2维或 3维纹理[3,4]进行体绘制时, 有限的纹理内存限制了对大规模体数据的交互绘制。

Guthe等人利用小波变换对体数据进行无损压缩[5], 在绘制时对编码数据实时解码。但每个纹理需要在 CPU上完成解码后载入图形硬件进行绘制。Kraus和 Ertl将体数据进行分块, 然后再合成为大小一致的纹理[6]。利用图形硬件的可编程特性完成最终的解码过程, 重建出绘制所需的纹理信息。Li等人利用数据相识性进行纹理组合来减少原始体数据纹理的大小[7]。然而梯度信息需要预先计算并存储在图形硬件中, 尽管对梯度纹理也进行了组合但仍然占用了大量的纹理内存。Schneider等人提出采用矢量量化 (vector quantization)技术对体数据进行压缩[8], 在一定的误差内可以自动得到压缩后的数据来近似表示原始数据。但仅能采用最近邻插值进行绘制, 影响了绘制图像的质量。

另外可以利用 OpenGL 和 D3D 等标准图形 APIs提供的纹理压缩技术对体数据进行压缩, 然而到目前, 两种 APIs都不支持无损压缩。由于有损压缩带来了一定的纹理误差, 尤其是对梯度信息影响更大, 会显著的降低绘制质量[9]。

本文的主要贡献是提出了一种新方法, 利用动态纹理载入技术实现了在普通 PC硬件上对大规模体数据的交互体绘制。这种方法没有使用复杂的编码方式对体数据进行压缩, 避免了在绘制中耗时的解码操作对交互性的影响以及编解码带来的误差。在绘制中仅有一个合适大小的体数据块被载入 3维纹理, 克服了纹理内存对体数据规模的限制。同时, 通过利用目前通用 PC图形硬件的可编程特性对光照计算中的梯度进行实时的计算和归一化处理, 该方法同时降低了 CPU和 GPU (图形处理器)的内存消耗, 也加快了体数据的预处理过程。因此, 这种方法非常适用于对绘制质量有较高要求的大规模体数据可视化应用中。

3　基于 PC硬件大规模数据场体绘制

3.1　直接体绘制

直接体绘制技术通常被分为基于像空间的绘制, 如射线投射法 (ray-casting)和基于物空间的绘制, 如足迹法 (splatting)、剪切-曲变法 (shear-warp)和基于纹理映射的方法。体绘制的主要任务是在每个像素上对体绘制积分进行近似计算, 即沿着入射

光线对逐渐衰减的颜色和衰减系数进行累加。

假设入射光线表示为 $\boldsymbol{x}(\lambda)$, 其中 λ为到观察点的距离。对空间中的任意一点 $\boldsymbol{x}$, 颜色强度为 $\boldsymbol{c}(\boldsymbol{x})$, 衰减系数为 $\tau(\boldsymbol{x})$, 这样体绘制积分可以表示为

$$I = \int_0^D \boldsymbol{c}(\boldsymbol{x}(\lambda))\, \mathrm{e}^{-\int_0^{\lambda}\tau(\boldsymbol{x}(\lambda))\,\mathrm{d}\lambda}\,\mathrm{d}\lambda$$

D 为最大观察距离。在空间点 $\boldsymbol{x}$ 处, 体素根据函数 $\boldsymbol{c}(\boldsymbol{x})$进行光线发射, 同时衰减系数 $\tau(\boldsymbol{x})$在观察点和体素位置上的累加对发射的能量进行减弱。

但是, 由于颜色和衰减系数没有指定, 对于连续标量场 $s(\boldsymbol{x})$上述的积分是无法进行计算的。首先要对体数据进行分类来指定颜色和衰减系数。通过引入传递函数 (transfer function) 来将体数据强度 $s=s(\boldsymbol{x})$映射为颜色强度 $\boldsymbol{c}(s)$和衰减系数 $\tau(s)$, $\boldsymbol{C}$ 一般为颜色空间中的颜色矢量函数。

这样体绘制积分可以表示为

$$I = \int_0^D \boldsymbol{c}(s(\boldsymbol{x}(\lambda)))\, \mathrm{e}^{-\int_0^{\lambda}\tau(s(\boldsymbol{x}(\lambda)))\,\mathrm{d}\lambda}\,\mathrm{d}\lambda \qquad (1)$$

沿入射光线采样, 对体绘制积分进行离散化, 引入阻光度 A, 定义为

$$A_i = 1 - \mathrm{e}^{-\tau(s(\boldsymbol{x}(id)))d} \qquad (2)$$

同样, 第 i个采样光线间隔上的颜色矢量可以近似表示为

$$\boldsymbol{C}_i = \boldsymbol{c}(s(\boldsymbol{x}(id)))d \qquad (3)$$

这样, 离散化的体绘制积分可以表示如下, 其中 $N=[D/d]$为采样点的个数, d 为采样点间隔。

$$\boldsymbol{C} = \sum_{i=0}^{N} \boldsymbol{C}_i \prod_{j=0}^{i-1} (1 - A_i) \qquad (4)$$

将最大观察距离为 D 的入射光线 $\boldsymbol{x}(\lambda)$分为连续的光线段, 这样第 i个采样光线间隔上的颜色矢量 (如式 (4)所示)可以由每个光线段的颜色来近似表示

$$\boldsymbol{C} = \sum_{i=0}^{n_0} \boldsymbol{C}_i \prod_{j=0}^{i-1} (1 - A_j) + \sum_{i=n_1}^{n_2} \boldsymbol{C}_i \prod_{j=0}^{i-1} (1 - A_j) + \cdots + \sum_{i=n_{L-1}}^{n_L} \boldsymbol{C}_i \prod_{j=0}^{i-1} (1 - A_j) \qquad (5)$$

式中, $N = \sum_{k=0}^{L} n_k$, 每个部分都可以通过从前向后或从后向前进行 Alpha混合迭代进行实现。

3.2 数据分块和纹理动态载入

平行于视平面的多边形切片是在基于 3维纹理映射的绘制中最常用的方法。体数据被装载到一个3维纹理中, 在切片多边形内部上进行插值得到顶点的 3个纹理坐标, 在光栅化过程中, 这些纹理坐标被用来对 3维纹理进行采样, 利用硬件的 3次线形插值得到相应的体数据。

将体数据划分为 L 个连续的数据块, 对第 k个数据块, 其对体绘制积分中第 i个采样光线间隔上的颜色矢量 $\boldsymbol{C}$ 的贡献可以表示为

$$\boldsymbol{C}_k = \sum_{i=n_k}^{n_{k+1}} \boldsymbol{C}_i \prod_{j=0}^{i-1} (1 - A_i) \qquad (6)$$

采样光线在数据块上的采样为 $[n_k, n_{k+1}]$。在基于 3维纹理映射的方法中, 每个数据块可以表示为一个 3维纹理, 采用平行于视平面的多边形切片对数据块采样, 并按照由前向后或由后向前的顺序混合得到该数据块的颜色贡献, 这些临时的颜色和阻光度信息在图形硬件的帧缓存中进行累加, 当完成对所有数据块的绘制时就生成了最终的绘制结果。

在数据的预处理过程中, 体数据被划分为一系列大小不同的数据块。这些数据块都是连续却又相互不交叠的, 在一次完整的绘制中, 每个数据块最多被采样一次, 以保证没有体素被进行多次混合。

在绘制中, 由式 (5)可知, 每次仅绘制一个数据块, 并将每个数据块的绘制结果组合起来得到的图像和同时对所有数据块进行绘制得到的结果是一致的。这样可以按照正确的绘制顺序将一个数据块动态的载入 3维纹理进行绘制, 而将其余的数据块存放在系统内存中。每次绘制完当前的数据块时, 下一个数据块被载入图形硬件并卸载当前的数据块。在整个绘制过程中, 只需在纹理内存中驻留一个数据块。这种动态纹理载入的特性使得提出的算法克服了目前通用图形硬件对绘制数据容量的限制。整个绘制流程如图 1所示。

在进行数据分块时, 数据块数目的确定需要特别注意。数据块越多, 用在纹理切换的开销就会越高。然而增加数据块的大小, 相当于减少数据块的数目, 由于带宽的限制会增加数据由系统向图形硬件传输的延时。对于配有 256MB内存的图形硬件, 一般数据块的大小为 $512\times512\times256$。

另外, 由于图形硬件不支持跨越数据块边界的插值计算, 这会在数据块之间的边界部分产生明显的伪影。为了保证 3次线形插值能够在边界区域正确的进行, 需要重复数据块边界的体素。在进行数据分块时, 对每个数据块, 还要同时采样其在 x, y 和

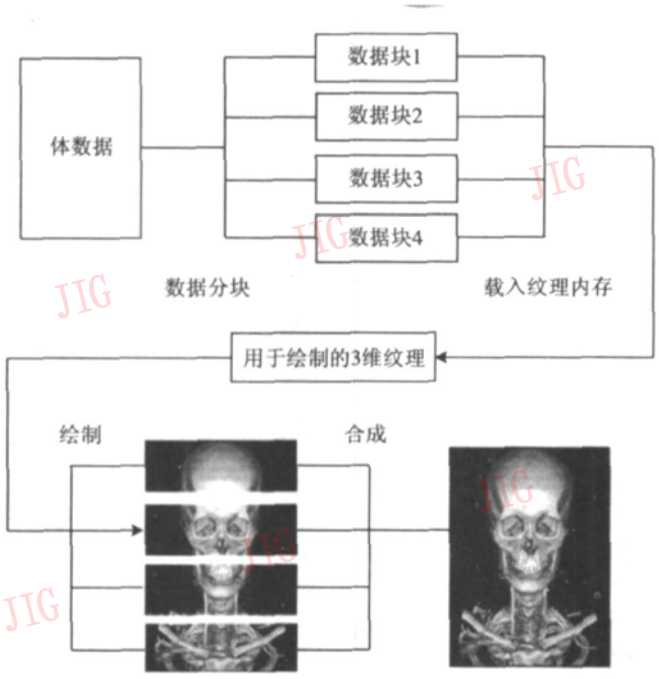


图 1　绘制流程

Fig 1　Rendering pipeline

z方向上相邻的数据块。如果数据块在整个体数据的边界上, 则不在该侧进行采样。这样数据块相当于在 x, y和 z方向上增加了一个体素来保存边界邻域体素。在绘制中, 利用这些邻域体素就可以在数据块边界处完成连续 3次线形插值。同时由于这种技术使得数据块之间有一个体素的交叠, 会带来内存使用效率的降低。

3 3　梯度的实时计算

在利用纹理映射绘制体数据时, 通用图形硬件没有对梯度的计算提供支持。为了实现光照处理, 通常要预先对体数据的表面梯度进行计算, 梯度和体数据以 RGBA 的格式存储在纹理内存中。这样极大的占用了通用图形硬件中有限的内存空间, 对一个 512 × 512 × 512 规模的体数据, 最少需要 512MB的 3维纹理内存来装载这样的数据, 这已远远超出了目前大多数通用图形硬件的内存限制。

本文基于对梯度进行实时计算的思想来有效的减少纹理内存的巨大占用。这种方法仅在进行光照计算时, 完成该点处梯度的计算。这样只需在纹理内存中存储原有传统纹理绘制 1/4的数据, 极大的增加了应用范围。同时也无需在预处理过程中进行耗时的梯度计算, 直接载入体数据, 提高了交互性。

为了在保证图像质量的同时兼顾运算的复杂性, 采用计算简单的基于中心差分的梯度计算方法。对一个数据场 s 在体素 (x, y, z)处的梯度为

$$g(x, y, z) = \nabla s(x_i, y_j, z_k) = \frac{1}{2}(s(x_{i+1}, y_j, z_k) - s(x_{i-1}, y_j, z_k)),$$
$$\frac{1}{2}(s(x_i, y_{j+1}, z_k) - s(x_i, y_{j-1}, z_k)),$$
$$\frac{1}{2}(s(x_i, y_j, z_{k+1}) - s(x_i, y_j, z_{k-1}))$$

近几年来, 通用图形硬件在顶点处理和像素处理阶段都提供了灵活的可编程性。OpenGL 的 fragment program 特性应用在光栅化阶段, 对每个像素进行处理。为了实时的计算体数据的归一化梯度, 梯度“ fragment program”应完成下面的任务:

(1)根据当前纹理坐标, 得到 6邻域纹理坐标;

(2)利用相关纹理采样得到所需 7个体数据;

(3)计算并归一化梯度。

实现中使用了 20条指令, 其中 7条为 3维纹理寻址操作, 6条矢量运算和 7条标量运算。在得到规一化的梯度后, 就可以利用 Phong-Blinn光照模型来计算采样点处的光照强度[2]。

4　实验结果

用大量的医学临床数据对上述算法进行了测试实验, 所有实验都是在一台 P4, 2.8GHz, 2GB 内存, 操作系统为 Windows XP的 PC机上完成的, 显卡为配有 256MB显存的 NVIDIA Geforce 7800 GT。采用 C++和 OpenGL对提出的算法进行了实现。

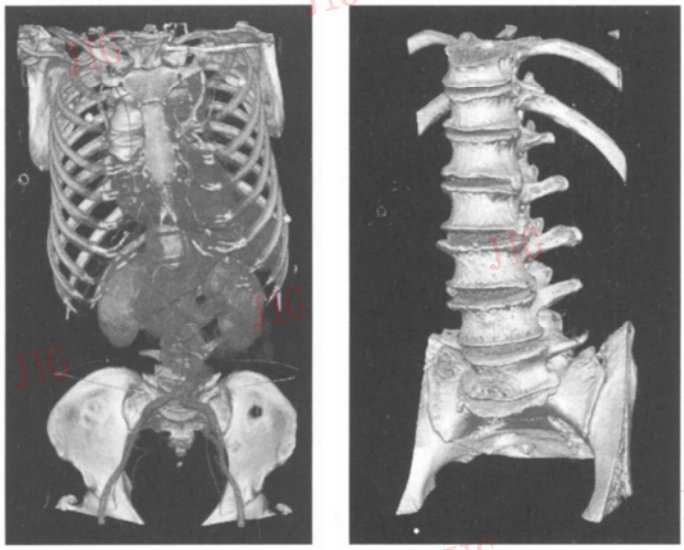
表 1给出了实验数据的运行时间, 包括体数据分块的预处理时间, 数据块数目, 3维纹理消耗以及绘制的帧速率。因为在预处理过程中无需对梯度进行计算, 也没有涉及数据编码过程, 使得数据能够快速的载入图形硬件。数据块的大小为 512 × 512 × 256, 3维纹理占用均为 64MB。

表 1　性能参数, 绘制窗口大小为 512 × 512

Tab 1　Performance statistics for a 512 × 512 viewport

体数据	数据规模	预处理 (s)	数据块数目	帧速率 (fps)
腹部	512 × 512 × 527	0.95	3	8.1
脊柱	512 × 512 × 567	0.99	3	8.1
人体-1	512 × 512 × 690	1.29	3	7.8
人体-2	512 × 512 × 957	1.71	4	5.6
下肢	512 × 512 × 1 156	2.13	5	4.5
人体-3	512 × 512 × 1 582	2.83	7	3.3

实验所采用的体数据均来自医学临床的 CT扫描数据。图 2为腹部器官和脊柱骨骼绘制效果,可见本文提出的算法提供了理想的绘制质量。在绘制结果中可以清晰的看到血管分支,而且在高密度部位,如骨骼上没有出现“阶梯伪影”。

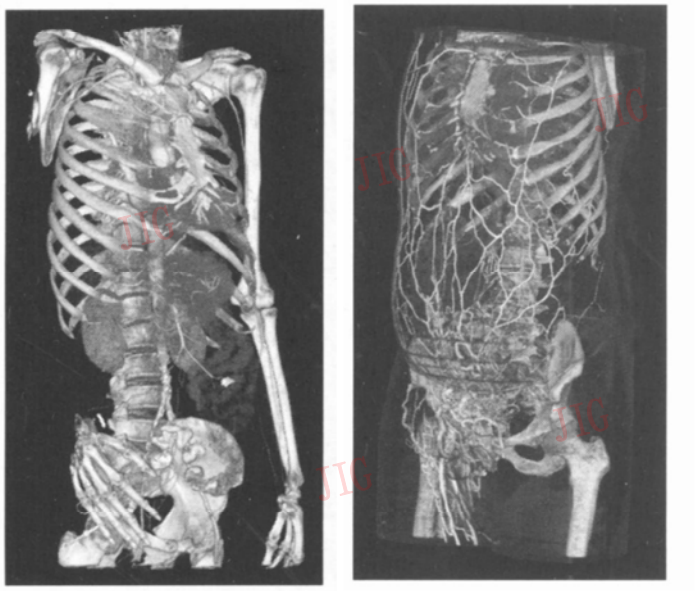


(a) 腹部 (b) 脊柱

图 2 腹部和脊柱绘制效果

Fig 2 Rendering of abdomenand spine

图 3和图 4为 4个超出图形硬件限制的临床实际数据的绘制结果,表现了算法对大规模体数据的绘制能力。从绘制图像上,可清楚地显示出体数据中的各种内部结构,不同器官的空间关系也非常直观的进行了展示。



(a) 人体-1 (b) 人体-2

图 3 人体-1和人体-2绘制效果

Fig 3 Rendering of body1 and body2

(a) 下肢 (b) 人体-3

图 4 下肢和人体-3绘制效果

Fig 4 Rendering of lower limbs and body3

5 结 论

为克服传统基于 3维纹理映射的体绘制方法中纹理内存对体数据规模的限制,本文提出了一种新的体绘制算法,在目前 PC通用图形硬件上实现了大规模数据场的可视化。

本文提出的方法在数据预处理时,将体数据划分为大小合适的数据块,通过动态纹理载入技术将每个数据块按照正确的绘制顺序依次载入 3维纹理,并利用纹理映射进行绘制,在图形硬件帧缓存中对数据块的绘制结果进行组合,得到最终的绘制图像。在整个绘制过程中,仅有一个数据块存储在图形硬件上,实现了对超过纹理内存容量的大规模体数据的绘制。同时,利用 OpenGL的 fragment program扩展将梯度的实时计算结合到绘制过程中,明显的减少了传统纹理映射体绘制方法的巨大内存占用。实验结果表明,利用目前通用图形硬件灵活的纹理操作和强大的光栅化能力,可以得到具有较高质量的可交互的绘制结果。

在将来的工作中,将利用本文提出的方法对时变(time-varying)体数据的可视化进行研究,同时采

用更加灵活的数据分块算法, 提高纹理内存的利用效率。由于梯度对绘制质量非常重要, 但为了保证绘制交互的实时性, 本文采用了中心差分法来计算梯度。另一个可以进行的工作是采用更准确的梯度计算方法来提高精度, 同时可以使用目前最新的 GPU 提供的 32位浮点运算能力进一步改善绘制质量。

参考文献 (References)

1 Zheng Jie, Ji Hong-bing. High quality per-pixel shading in texture-based volume rendering [A]. In: Proceedings of IEEE the International Conference on Neural Networks and Brain[C], Beijing, 2005: 1258~ 1261.

2 Cabral B, Cam N, Foran J. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware[A]. In: Proceedings of ACM Symposium on Volume Visualization' 94[C], New York, USA, 1994: 91~ 98.

3 Engel K, Kraus M, Ertl T. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading [A]. In: Proceedings of ACM Eurographics/Siggraph Workshop Graphics Hardware 2001[C], Los Angeles, CA, USA, 2001: 9~ 16.

4 Kniss J, Kindlmann G, Hansen C. Multidimensional transfer functions for interactive volume rendering[J]. IEEE Transactions on Visualization and Computer Graphics, 2002, **8**(3): 270~ 285.

5 Guthe S, Wand M, Gonser J, Strasser W. Interactive rendering of large volume data sets[A]. In: Proceedings of IEEE Visualization 2002[C], Boston, MA, USA, 2002: 104~ 115.

6 Krauss M, Ertl T. Adaptive texture maps[A]. In: Proceedings of the ACM Siggraph/Eurographics Graphics Hardware Workshop [C], Saarbrucken Germany, 2002: 7~ 15.

7 Li W, Mueller K, Kaufman A. Empty space skipping and occlusion clipping for texture-based volume rendering[A]. In: Proceedings of the 14th IEEE Visualization 2003[C], Seattle, WA, USA, 2003: 317~ 324.

8 Schneider J, Westermann R. Compression domain volume rendering [A]. In: Proceedings of the 14th IEEE Visualization 2003[C], Seattle, WA, USA, 2003: 293~ 300.

9 Meißner M, Guthe S, Straßer W. Interactive lighting models and pre-integration for volume rendering on PC graphics accelerators[A]. In: Proceedings of Conference on Human-Computer Interaction and Computer Graphics [C], Calgary, Alberta, Canada, 2002: 209~ 218.