doi:10.3772/j.issn.1002-0470.2015.10-11.006

基于 GPU 的支持多重散射的体光照技术①

刘 宁②[*,**] 朱登明[*] 陆一峰[*,**] 王兆其[*]

([*]中国科学院计算技术研究所虚拟现实实验室 北京 100190)

([**]中国科学院大学 北京 100049)

摘 要 为了提高体光照——体数据可视分析的重要手段的性能,针对现有的体光照技术存在的光照模型过于简化、只考虑光照的局部特征、收敛速度过慢的问题进行了创新研究,提出了一种基于图形处理单元(GPU)的能够支持多重散射的体光照方法。该方法在进行重要性采样时不仅根据相函数采样重要的方向,同时也考虑各方向上所携带的能量大小,因此能够更快地收敛;为了快速估算出一个方向上所携带的能量,采用了一个新颖的基于哈希桶的体数据代理,利用这个代理能够有效地跳过相似的体素,从而提高光照计算的效率。该方法相比于传统的方法能够取得更显著的加速比,同时能够有效地支持全面光照,产生复杂的光照和阴影效果。

关键词 体光照,多重散射,加速结构,重要性采样,图形处理单元(GPU)

0 引 言

体光照技术是对体数据进行可视分析[1-3]的重要手段,它能够逼真地展示出体数据的内部特征,因此在科学模拟计算和医学数据分析等诸多领域得到了广泛运用。

体光照是一个计算密集型的问题。为了提高体光照算法的效率和支持更真实的光照效果,研究人员做了大量努力。Hadwiger 等[4,5]将深度阴影图技术应用到了体绘制方面。Hernell 等[6,7]则提出了一个模拟环境光遮挡的体光照算法。该算法只考虑了光照的局部特征或者只能在特定表面上产生真实的渲染结果。与此同时,随着硬件技术的进步,体光照算法采用了越来越真实的光照模型,硬阴影、软阴影、单重散射和多重散射等比较高级的光照效果逐步得到了支持[8]。Salama[9]提出了一个基于 Monte-Carlo 重要性采样的算法来模拟多重散射效果,但这个方法在采样时只考虑了重要的方向,而没有考虑各方向上所携带的能量大小,因此它需要比较长的时间才能收敛。Ropinski 和 Sunden 等[10-12]利用一个锥形数据代理来简化多重散射的计算,但该方法不支持多光源的场景。本文提出了一个新颖的基于图形处理器(GPU)的体光照算法,该算法采用的是基于物理的光照模型,没有做不切实际的假设。进行采样时不只考虑相函数的重要方向,同时也考虑了各方向上所携带的能量的大小,因此能够比传统方法收敛得更快。该方法按照数据的空间连续性与数据连续性对体数据进行压缩并产生一个基于哈希桶(Hash bucket)的体数据代理,利用这个代理,可以快速地跳过相似体素(voxel),因此能够高效地计算出一个方向上的能量。体光照通常需要多个渲染迭代过程,在每个迭代过程中都需要构建加速结构,因此加速结构的构建时间是影响渲染效率的重要因素。基于哈希桶的数据代理相对传统的加速结构(如八叉树)的一个主要优点是它的构建时间很短,

① 国家自然科学基金(61173067, 61379085)资助项目。

② 男,1987 年生,博士生;研究方向:科学数据可视化;联系人,E-mail: liuning01@ict.ac.cn

(收稿日期:2015-04-22)

因此更适用于这种场景。因为体光照问题具有很大的并行性,因而该算法全部在 GPU 上实现,以充分利用 GPU 提供的并行计算能力。

1 方法概述

本文提出了一个基于 Monte-Carlo 的体光照多重散射算法。在进行重要性采样时,它不只根据相函数去选择重要的方向,同时也将在方向上所携带的能量考虑进去,因此本文方法相对于传统的基于 Monte-Carlo 的方法能够更快地收敛。本文提出了一个基于哈希桶的体数据代理,它利用体数据的空间一致性和数据一致性对体数据进行匀质剖分。利用这个代理,本文方法能够快速跳过相似体素,因此能够显著地加快光线追踪时能量累积的计算过程。

本文方法首先进行单重散射的计算,之后再进行若干次迭代来进行多重散射的计算。在每一次迭代过程中都需要重新构造哈希桶代理并进行 Monte-Carlo 能量重要性采样计算。在达到最大迭代次数之后,算法会终止并输出最终渲染结果。图 1 是本文方法的示意图。

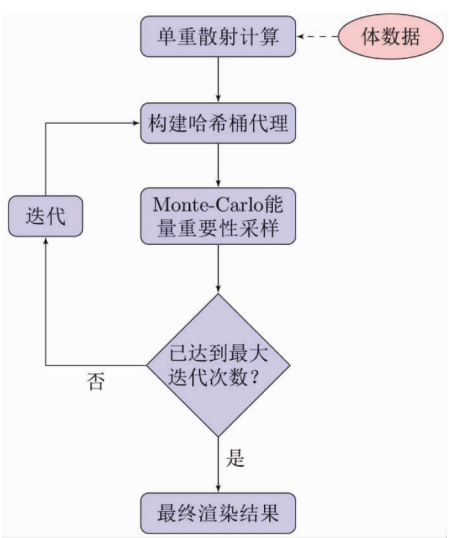


图 1 本文方法流程图

2 基于哈希桶的体数据代理

体数据通常由连续空间的离散采样或者科学模拟计算产生。因此它们通常具有下面两个性质:(1)空间连续性,即在空间中相近的体素通常具有相似的数据值;(2)数据连续性,即在一个大的体数据中,存在大量具有相似数据值的体素。基于这些性质,研究人员发明了许多加速结构来更有效地表示体数据。八叉树是最流行的加速结构之一。在光线投射的计算过程中,相似的体素被视为一个大的体素,因此它可以显著地加速能量积累的计算过程。但八叉树的构建是一个非常耗时的操作。由于体光照通常会包含多个渲染迭代过程,其中在每次迭代过程中都需要进行加速结构的构建,因此八叉树不太适用于这种场景。

针对这个问题,本文提出了一个更为有效的基于哈希桶的体数据代理。八叉树是按照空间对体数据进行均匀剖分,而哈希桶代理则是根据体数据的空间连续性与数据连续性对其进行匀质剖分,因此哈希桶代理能够更好地反映出数据的特征。同时,哈希桶代理的构建时间要远小于八叉树的构建时间,因此它能够更好地适应于这种需要多次迭代计算的场景。

2.1 哈希桶的结构

体数据的哈希桶代理由一个哈希桶的列表组成。哈希桶由 bucketID 标识,它通常包含原始体数据的多个体素的信息。

为了构建哈希桶代理,首先计算每个体素(由 voxelID 标识)的哈希值。在我们的实现中,体数据的颜色值包含 RGBA 四个通道并被存储为一个 32 位的内存单元,其中每个通道占据一个字节大小的空间。每一个颜色通道需要与一个映射强度参数 mask 进行掩模运算。如果 mask 是 11111111,那么颜色将会以无损的方式映射到哈希值。如果 mask 是 11111110,那么颜色通道的最后一位在映射过程中将不被使用。映射完成之后,所得到的数值将会被解释为一个整型的哈希值。可以看出,通过控制映射强度参数 mask,相似但不相同的体素可以被映射到相同的哈希值上。一个哈希桶由下面四个部分组成:

(1) bucketColor,哈希桶的颜色值,可以由映射到该哈希桶的体素的颜色值与映射强度 mask 进行掩模运算得到;

(2)hashValue,哈希桶的哈希值;

(3)bucketBase,映射到该哈希桶的第一个体素的 voxelID;

(4)bucketSize,映射到该哈希桶的体素数量。

2.2　GPU 上哈希桶代理的构建

在 GPU 上构建哈希桶代理的过程可以概括为“压缩-排序-压缩-解压缩”的过程,其中第一次压缩是为了将在空间上连续的相似体素压缩到一个哈希桶中;排序是将压缩得到的哈希桶按照哈希值进行排序,这样在原始体数据中即使不相邻但具有相似颜色值的体素也会被聚到一起;第二次压缩是针对排序后的哈希桶的,经过这次压缩之后,哈希桶代理不但考虑了体数据的空间连续性,也考虑了数据连续性;最后的解压缩过程是根据哈希桶计算出每个体素的对应的 bucketID。体光照算法可以利用该代理结构快速地跳过相似体素,因此能够提高渲染的效率。图 2 是在 GPU 上构建哈希桶代理的示意图。

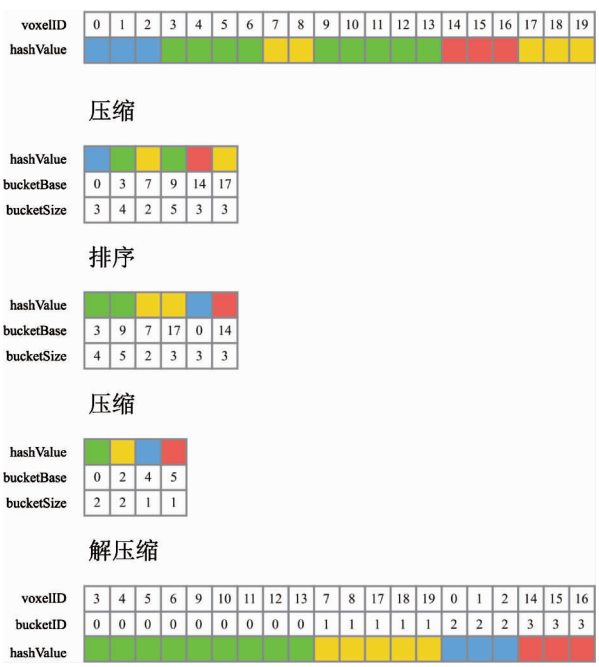


图 2　GPU 上构建哈希桶代理示意图

具体的构建步骤如下:

(1)哈希计算。利用 GPU 上的 map 操作可以并行地计算出每个体素的哈希值。在执行 map 操作时,可以通过改变映射强度参数 mask 来调节体素相似性的阈值,因此具有相似但不相同颜色值的体素也可能映射到同一个哈希值上。

(2)第一次压缩。压缩对象是体素的哈希值数组,目的是将空间中连续的具有相同哈希值的体素压缩到一个哈希桶内。该压缩过程的核心运算是 GPU 上的 scan 操作。具体的计算过程如下:*flag* 标记数组的长度与体素个数相同,其中若体素 i 的哈希值与前面体素不同,则 $flag(i)$ 置为 1;否则,$flag(i)$ 置为 0。然后对 *flag* 数组执行 scan 操作并产生一个积累数组 *accFlag*。在构建哈希桶时,如果 $flag(i)$ 为 1,则新建一个哈希桶,其中 *hashBase* 由 $accFlag(i)$ 确定,*hashSize* 则由相邻两个哈希桶的 *hashBase* 的差值决定。

(3)排序。对第一次压缩产生的哈希桶进行排序操作,使得具有相同哈希值的哈希桶会被放在一起。排序过程可以使用 radix-sort,它在 GPU 上有高效的实现。第一次压缩时只能处理空间上相邻的体素,因为只考虑了体数据的空间连续性,没有考虑数据连续性,对于那些空间上不相邻但具有相似颜色值的体素则没有办法处理。排序的主要作用是利用数据连续性将那些不相邻但具有相似颜色值的体素聚合到一起,并为第二次压缩做准备。

(4)第二次压缩。对于经过排序过的哈希桶执行第二次压缩操作。第一次压缩利用体数据的空间连续性将相邻的具有相同哈希值的体素压缩到一起,第二次压缩则利用体数据的数据连续性,将那些空间上不相邻但具有相同哈希值的体素压缩到一起。

(5)解压缩。对于每个哈希桶而言,会通过相应的 hashBase 与 hashSize 展开为一系列的 voxelID,即每个体素都会有一个与之对应的 bucketID。在沿着光线累积能量时,如果当前体素的 bucketID 与前面体素的相同,则只需要简单地累加能量,而不需要从原始体数据中获取,因此可以快速地跳过相似体素从而提高能量计算的效率。

3　Monte-Carlo 能量重要性采样

体光照的核心是求解体渲染方程,它的一般形式是:

$$L(x,\omega_o) = L_0(x_0,\omega_o)T(x_0,x) + \int_{x_0}^{x}(\sigma_a(x')L_e(x,\omega_o) + \sigma_s(x')\int_{\Omega}s(x',\omega_i,\omega_o)L(x',\omega_i)\mathrm{d}\omega_i)T(x',x)\mathrm{d}x' \quad (1)$$

在这里 $L(x,\omega_o)$ 是在点 x 向方向 ω_o 发射的能量；$L_0(x_0,\omega_o)$ 是背景能量；$L_e(x,\omega_o)$ 是点 x 朝方向 ω_o 辐射的能量值；$T(x_i,x_j)$ 是从点 x_i 到点 x_j 的衰减系数；$\sigma_a(x)$ 是点 x 处的吸收系数；$\sigma_s(x)$ 是点 x 处的散射系数；$s(x,\omega_i,\omega_o)$ 是点 x 处的相函数，它描述了从方向 ω_i 进来的能量散射到方向 ω_o 的比例。由于 $L(x,\omega_o)$ 同时出现在方程的两边，因此该渲染方程一般没有解析形式的解。

求解体渲染方程的一种常用技术是采用 Monte-Carlo 重要性采样。它根据相函数采样一组重要的方向，并沿着这些方向累积能量。Henyey-Greenstein 相函数是一个流行的相函数，它定义为

$$s(\varphi,g) = \frac{1-g^2}{4\pi(1+g^2-2g\cos\varphi)^{3/2}} \quad (2)$$

在这里 g 是一个常数，它用来控制相函数的形状；φ 是方向 ω_i 与 ω_o 之间的夹角。

传统的 Monte-Carlo 方法依据相函数对重要的方向进行采样。但是这些方向所携带的能量可能非常小，对最终的渲染结果的贡献会微乎其微，因此可能需要很长时间才能收敛。在采样方向时，同时也考虑这些方向中所携带的能量大小会更加合理，这正是本文方法的出发点。首先，本文方法利用 Monte-Carlo 重要性采样（MIS）依据相函数去采样重要的方向，但增加采样方向的数量，比如从原来的 N 个增加到 M 个（$M > N$，在具体实现中我们取 $M = 2N$）。然后我们利用哈希桶代理快速地估计这 M 个方向所携带的能量，并从中选取携带能量最多的 N 个方向。最后我们在原始数据中追踪这 N 个方向，并产生渲染结果。

因为增加了需要追踪的方向的数目，因此如何快速地估算出每个方向上能量的大小对算法整体效率有很大的影响。这正是本文所提出的基于哈希桶的体数据代理的主要作用。在沿着光线方向累积能量时，如果当前体素的 bucketID 与前面的体素相同，则算法只是简单地积累能量，而无需从原始体数据中去获取，因此能够快速地跳过相似的体素，从而提高渲染效率。

4 实现细节

本文方法在 GPU 上利用通用并行计算架构 CUDA 和 OpenGL 得到了实现。算法的核心是 ray-casting。针对每个像素生成一条光线，并由一个 CUDA 线程去负责一条光线。每一个 CUDA 线程会积累所经过体素的辐射、单重散射与多重散射的贡献。之后，本文方法会产生一个与视口大小相同的像素纹理并利用 OpenGL 将这个纹理渲染到屏幕上去。

在实现传统的 Monte-Carlo 重要性采样（MIS）方法时，一般通过采样 8 个重要的方向来估算多重散射的贡献。通常在 3 到 5 次迭代之后，就可以得到最终的渲染结果。

实现本文方法时，首先生成原始体数据的哈希桶代理，然后采样 16 个重要的方向。利用哈希桶代理，可以快速地估计这些方向上所携带的能量大小，并从中选择能量最大的 8 个方向。之后在原始体数据中追踪这 8 个方向，并将累积的能量作为该位置的渲染结果。每个迭代过程中，本文方法的时间耗费要多于传统的 MIS 方法，但本文方法的迭代次数较小，因此最终本文方法能够取得更好的渲染效率。

5 结 果

试验环境的硬件信息如表 1 所示。

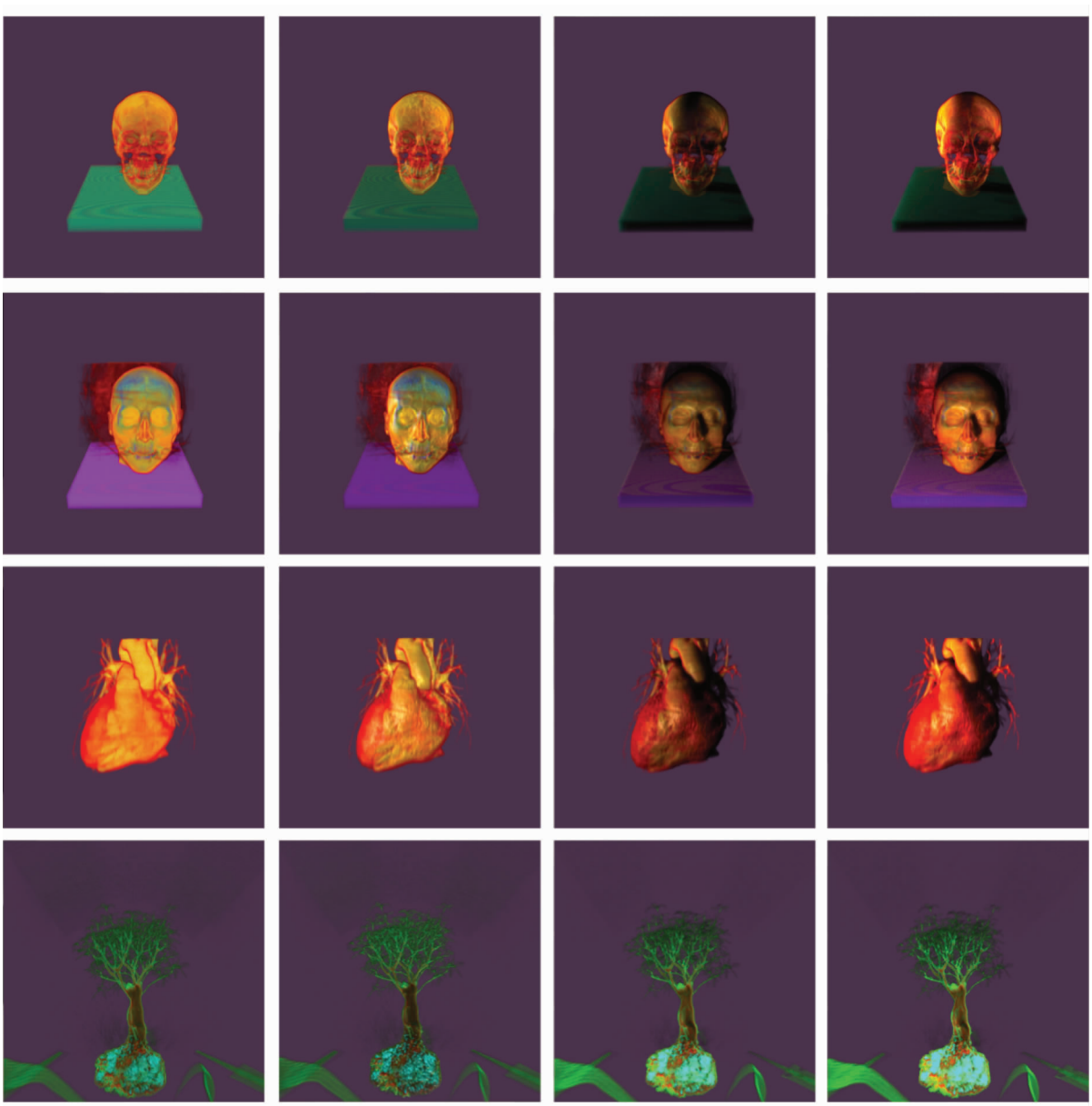
表 1 实验环境的硬件参数

GPU	CUDA 核数	内存带宽
NVIDIA GT 540M	96	128 位
NVIDIAGTX550Ti	192	192 位

我们在四组不同的数据上比较了本文方法与传统的 MIS 方法，结果如表 2 所示。表 2 中 AVP 代表活跃体素比例（active voxels percentage）。AVP 低的

表 2　本文方法与传统 MIS 方法渲染效率对比

数据名称	数据尺寸	AVP	平台	MIS(fps)	本文方法(fps)	加速比
Skull	256×256×256	18%	540M	0.037	0.111	2.99
			550Ti	0.077	0.215	2.79
Head	256×256×256	51%	540M	0.015	0.029	1.96
			550Ti	0.030	0.069	2.19
Heart	256×256×256	23%	540M	0.023	0.064	2.80
			550Ti	0.048	0.130	2.72
Tree	512×512×182	32%	540M	0.0067	0.0147	2.20
			550Ti	0.0137	0.0304	2.22



各列分别代表不同的光照模型,直接体绘制直接将数据值映射为颜色,其结果缺乏立体感,也不支持阴影或者更高级的光照效果;Phong 光照模型能够增强结果的立体感;单重散射能够支持硬阴影;多重散射则能够很好地支持软阴影与全局光照效果,因此它的效果最为逼真

图 3　不同光照模型渲染结果对比

数据往往比较稀疏,因此相应的渲染结果看起来更透明。可以看出,随着 AVP 的增加,本文方法相对于传统 MIS 方法的加速比在减小。这是因为当 AVP 增加的时候,哈希桶代理将会包含更多的信息,因此相应的处理时间会增加,也就导致了最终的渲染时间会延长。值得注意的是,在 GTX 550Ti 平台上那些 AVP 较大的数据的加速比比 GT 540M 平台更大一些。这是因为 GTX 550Ti 具有较大的内存带宽,因此内存拷贝操作要更高效一些。

这些数据的渲染结果如图 3 所示。第 1 列展示的是直接体绘制的效果,这个光照模型只考虑吸收和辐散的效果,因此阴影及更高级的全局光照效果都不受支持。第 2 列展示的是利用 Phong 光照模型的渲染结果。Phong 光照模型只考虑了点的局部特征,因此阴影及更高级的全局光照效果也不受支持,但 Phong 模型能够明显改善三维视觉效果。第 3 列展示的是单重散射的渲染效果。可以看出硬阴影开始出现,并且可以随着光源位置改变而改变。最后一列展示是多重散射的渲染结果。软阴影开始出现,并且更好地支持了全局光照的效果,因此最终结果看起来更加逼真。

6 结 论

传统的体光照技术要么采用过于简化的光照模型,要么收敛速度过慢。本文提出了一个基于 GPU 的高效的体光照多重散射算法,它采用的是基于物理的光照模型,而且没有做不切实际的假设。在进行重要性采样时,该方法不只按照相函数去考虑重要的方向,还同时将各方向所携带的能量也考虑进去,因此能够更快地收敛。为了达到这个目的,本文提出了一个新颖的基于哈希桶的体数据代理。利用这个代理,可以快速地估算出一个方向上所携带的能量。相对于传统的加速结构(如八叉树),哈希桶数据代理能够更好地反映出体数据的结构特征,同时它在构建速度与内存占用方面也有优势。在结果部分,本文方法与传统的 MIS 方法进行了详细的对比。试验结果表明本文方法相对于传统的 MIS 方法能够取得 2 到 3 倍的加速比。在渲染效果方面,本文的方法与直接体绘制、Phong 光照模型和单重散射进行了比较。本文的方法能够有效地支持全局光照效果,因此能够产生最为逼真的渲染结果。在下一步的工作中,我们计划研究体素颜色值到哈希值的自适应映射方案。同时,我们也会寻求方法来提高体光照算法的效率,使其可以运用在交互式图形领域。

参考文献

[1] Drebin A, Carpenter L, Hanrahan P. Volume rendering. In: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, Atlanta, USA, 1988. 65-74

[2] Garanzha K, Loop C. Fast ray sorting and breadth-first packet traversal for GPU ray tracing. *Computer Graphics Forum*, 2010, 29: 289-298

[3] Guthe S, Wand M, Gonser J, et al. Interactive rendering of large volume data sets. In: Proceedings of IEEE/SIGGRAPH Symposium on Volume Visualization and Graphics, Boston, USA, 2002. 53-60

[4] Hadwiger M, Kratz A, Sigg C, et al. GPU-accelerated deep shadow maps for direct volume rendering. In: Proceedings of ACM SIGGRAPH/Eurographics Symposium on Graphics Hardware, Vienna, Austria, 2006. 49-52

[5] Hadwiger M, Ljung P, Salama R, et al. Advanced illumination techniques for GPU volume raycasting. In: ACM SIGGRAPH ASIA 2008 Courses, Singapore, 2008

[6] Hernell F, Ljung P, Ynnerman A. Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering. In: Proceedings of the 6th Eurographics/IEEE VGTC Conference on Volume Graphics, Prague, Czech Republic, 2007. 1-8

[7] Hernell F, Ljung P, Ynnerman A. Local ambient occlusion in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 2010, 16(4): 548-559

[8] Jonsson D, Sunden E, Ynnerman A, et al. Interactive volume rendering with volumetric illumination. In: Proceedings of the 33rd Annual Conference of the European Association for Computer Graphics, Cagliari, Italy, 2012. 53-74

[9] Salama R. GPU-based monte-carlo volume raycasting.

In: Proceedings of the Pacific Conference on Computer Graphics and Applications, Maui, USA, 2007. 411-414

[10] Ropinski T, Doring C, Rezk-Salama C. Interactive volumetric lighting simulating scattering and shadowing. In: Proceedings of the 3rd IEEE Pacific Visualization Symposium, Taipei, China, 2010. 169-176

[11] Ropinski T, Kasten J, Hinrichs H. Efficient shadows for GPU-based volume raycasting. In: Proceedingsof the 16th International Conference in Central Europeon Computer Graphics, Visualization and Computer Vision, Bory, Czech Republic, 2008. 17-24

[12] Sunden E, Ynnerman A, Ropinski T. Image plane sweep volume illumination. *IEEE Transactions on Visualization and Computer Graphics*, 2011, 17(12): 2125-2134

A GPU-based volume illumination technique supporting multiple scattering

Liu Ning[*,**], Zhu Dengming[*], Lu Yifeng[*,**], Wang Zhaoqi[*]

([*]Virtual Reality Lab, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**]University of Chinese Academy of Sciences, Beijing 100049)

Abstract

To improve the performance of volume illumination, an important technique to visualize volume data, an efficient GPU-based volume illumination method supporting multiple scattering was presented based on an innovational study conducted to avoid existing volume illuminations' problems such as using an over-simplified lighting model, only considering local features and taking long time to converge. The new method not only considers the phase function direction when doing importance sampling, but also takes the associated energy in each direction into account, so it can converge more rapidly. In order to fast estimate the energy in a direction, it adopts a novel Hash bucket-based volume data proxy, and based on this proxy, similar voxels can be efficiently skipped and the illumination cost can be reduced. The proposed method can gain significant speedup over conventional methods and can produce realistic rendering results with advanced lighting and shadowing effects.

Key words: volume illumination, multiple scattering, acceleration structure, importance sampling, GPU