

基于 GPU 的快速色阶映射算子实现

钱银玲^{a,b}

(四川大学 a. 计算机学院; b. 视觉合成图形图像技术国防重点学科实验室, 成都 610064)

摘要: 为了提高色阶映射计算的效率, 设计了基于 GPU 的快速色阶映射算法。首先结合基本规约算法和 GPU 的并行运算特征设计了基于两个核函数的最大亮度计算方法, 然后通过区域中间值共享计算以像素为中心的区域平均亮度, 最后针对视屏处理, 提出利用纹理缓存池解决 CPU 读数据和 GPU 处理数据速度不匹配的问题, 并根据像素子集最大亮度自适应地更新全局最大亮度。实验结果相对相同算法的 CPU 实现得到了 4~5 倍的速度提升, 表明所提出的算法能够充分利用 GPU 的并行性, 并减少了大量重复运算, 满足实时渲染的要求, 并且对不同规模的纹理具有良好的适应性。

关键词: 色阶映射; 高动态范围; 算子; 图像处理; 视屏流

中图分类号: TP391.9

文献标志码: A

文章编号: 1001-3695(2013)02-0595-03

doi:10.3969/j.issn.1001-3695.2013.02.077

Fast tone mapping operator implementation based on GPUs

QIAN Yin-ling^{a,b}

(a. School of Computer Science, b. State Key Laboratory of Fundamental Science on Synthetic Vision, Sichuan University, Chengdu 610064, China)

Abstract: To improve the efficiency of tone mapping computing, this paper designed a fast implementation algorithm based on GPUs. First combined with basic reduce algorithm and the parallel computing characteristic of GPUs, the algorithm proposed two kernel functions to compute maximum luminance. Then it computed pixel-centered area average luminance using shared area intermediate result. Besides, for video stream processing, it proposed a solution to the unmatched problem of CPU reading and GPU computing by a texture pool and a heuristic method to update the global maximum luminance according to the value of a pixel set. Experiment proves that it obtains 4~5 efficiency improvement relative to CPU implementations. The algorithm can take full advantage of GPUs' parallel computing capability and reduce much duplicate computation. It's sufficient for realtime rendering and satiable for textures with different scales.

Key words: tone mapping; high dynamic range; operator; image process; video stream

在自然场景中, 光亮度有很广的范围, 很黑的夜里亮度为 10^{-5} cd/m², 强烈的太阳光亮度可以达到 10^5 cd/m², 动态范围达到了 $10^{10}:1$ 。相比之下, 一般显示设备亮度的动态范围显得很小, 如一般 CRT 的亮度范围为 $1\sim100$ cd/m², 一般 LCD 的亮度范围为 $1\sim400$ cd/m², 动态范围都不超过 $1000:1$ 。虽然两者的动态范围差别很大, 但是人的视觉系统对相对亮度比较敏感, 而对绝对亮度并不敏感, 所以对高动态范围图像进行压缩然后映射到一般显示设备能显示的低动态范围图像, 也能获得令人满意的效果。为了将利用 Mai^[1] 或 Mann 等人^[2] 的方法获得的高动态范围图像在一般的显示设备上显示出来, 就需要采用色阶映射算子进行动态范围压缩和色阶映射。

1 色阶映射算子

Mantiuk 等人提出了色阶映射算子的一般模型如图 1 所示^[1,3,4]。

从图 1 可以看出, 色阶映射算子实现了观察者对真实场景和显示器中展现场景在视觉上的匹配, 即在一般显示设备上体现了更强的真实感。

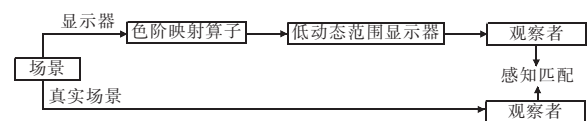


图 1 色阶映射算子模型

根据映射过程是否仅依赖于当前像素亮度可以将色阶映射算子分为全局算子和局部算子两类。

最简单的全局算子就是线性映射算子。映射过程通过式 (1) 实现。

$$L_d = \frac{L_w - L_{wmin}}{L_{wmax} - L_{wmin}} \quad (1)$$

其中: L_d 、 L_w 、 L_{wmax} 、 L_{wmin} 分别表示显示亮度, 真实场景中的当前点亮度、最大亮度和最小亮度。线性映射算子具有运算量小的优点, 对于亮度范围与显示器显示范围近似的场景, 可以达到令人满意的效果; 但对于亮度范围差别过大的场景, 会出现明显的失真现象, 如在夜里拍摄的场景, 亮度最大的像素仍会被映射到显示器上的最大亮度。

为了改善场景显示效果, 考虑人眼对亮度反应呈对数变化的现象, Stockham^[5] 提出应用对数函数进行映射, 映射函数如

式(2)所示。

$$L_d = \frac{\log(L_w + 1)}{\log(L_{wmax} + 1)} \quad (2)$$

很明显,式(2)的映射结果仍然存在与线性映射算子同样的问题,即将场景最大亮度映射到显示设备最大亮度,并且由于算法压缩率过高,场景的对比度不能很好地显现出来。

为了解决 Stockham 算法中细节丢失严重的问题,Drago 等人^[6]提出利用局部区域平均亮度自适应地调整映射算子中的参数。该算法提供用户 Bias 系数和曝光系数两个可调参数,通过调整参数,用户可以得到与真实场景很接近的效果。Drago 的算法具有运算量小、亮度范围压缩比高和细节保留完整的优点,所以适用于具有实时要求的系统,本文的实现就是基于这一算法。Lee 和 Bae 等人^[7,8]提出利用 K-均值对纹理进行区域划分,不同区域采用不同参数进行映射计算;Kitaura 等人^[9]则利用亮度信息进行区域划分;Kirk 等人^[10]提出基于感官的专用于低光照纹理的映射算子。以上算法能展示很好的效果,但都需要用户调整参数,为了避免这一问题,Mai^[11]和 Qiu 等人^[11]提出对局部像素的色调能量分布和 HDR/LDR 梯度变化约束建立概率统计模型,通过求解最大后验概率实现色阶映射,达到保留更多视觉信息的效果,使映射计算结果更具真实性。

众所周知,色阶映射算子存在运算量大的问题,对于一般的显示要求,如 1024×768 大小的纹理,需要处理的数据量就已经达到了 768 K。为了提高色阶映射算子的实现效率,Zhao 等人^[12]提出利用 GPU 进行色阶映射算子实现,并且比较了常见的色阶映射算子在 GPU 上实现结果的效率,指出 Pattanaik 等人^[13]提出的算法具有最高的效率,对用户之色阶映射算子的选择上起到了一定的指引作用。这也是目前最新的利用 GPU 进行色阶映射算子实现的文章。Yamauchi 等人^[14]提出结合视线信息进行局部映射计算,以大幅提高计算效率,但一般情况下视线信息未知,因此该算法不具有普适性。

本文结合当前最新的 NVIDIA CUDA API,充分利用 GPU 中的共享存储空间、纹理存储空间,通过设计复杂度近似为 $\log(N)$ 的最大亮度求解算法和高效的局部平均亮度计算以及复杂度为 1 的映射过程,达到 Drago 色阶映射算子的实时效果。

2 基于 GPU 的色阶映射算子实现

本文是基于 Dragon 的自适应色阶映射算子,整体映射函数如式(3)所示。

$$L_d = \frac{0.01L_{dmax}}{\log_{10}(L_{wmax} + 1)} \times \frac{\log(L_w + 1)}{\log(2 + (\frac{L_w}{L_{wmax}})^{\frac{\log(b)}{\log(0.5)}} \times 8)} \quad (3)$$

其中: L_{dmax} 为显示设备的最大亮度,对一般 LCD 显示器可以取 300 cd/m^2 。 L_{wmax} 、 L_w 是通过式(4)中曝光系数 e 和当前像素所处区域的平均亮度 L_{wa} 规范过的场景最大亮度和当前像素亮度。

$$L' = (e \times L) / L_{wa} \quad (4)$$

式(3)中参数 b 源于 Bias 幂函数,如式(5)所示。

$$\text{Bias}_b(x) = x^{\frac{\log(b)}{\log(0.5)}} \quad (5)$$

算子的实现可以分为三个步骤:

a) 计算每个像素亮度,并获取场景最大亮度;

b) 选取合适的区域计算每个像素点的区域平均亮度;

c) 利用式(3)进行色阶映射。

2.1 像素亮度和场景最大亮度计算

2.1.1 像素亮度计算

一般高动态范围图片的像素通过 R 、 G 、 B 三个通道的浮点值表示,Drago 采用 Rec. 709^[15] 颜色模型进行颜色到亮度的转换,转换公式如式(6)所示。

$$L = 0.2126R + 0.7152G + 0.0722B \quad (6)$$

为了后续计算的方便,像素亮度的计算结果存储在一维数组中。

2.1.2 最大亮度计算

最大亮度计算的算法思想是源于 NVIDIA CUDA Reduction^[16] 示例,即每个线程每次取两个亮度值,判断保留较大的亮度值,循环调用核函数直到最终计算得到最大亮度值。为了减少计算过程中的判断以提高运算效率,本文算法实现了两个核函数:

- 像素个数为 2 的整数次方的核函数;
- 像素个数不是 2 的整数次方的核函数。

前者每个线程只需要获取两个亮度值,并将较大亮度存储到对应空间,后者在取值过程中需要进行一次越界判断。

算法基本步骤为:

- 调用核函数 2,得到个数为 2 的整数次方的亮度数组;
- 循环调用核函数 1,计算最大亮度。

长度为 7 的亮度数组处理过程如图 2 所示。

2.2 区域平均亮度计算

为了更好地保持绘制结果中相邻像素的对比度,使细节更加突出,本文没有采用 Drago 算法中的区域共享区域平均亮度,而采用的是以像素为中心的区域平均亮度,两者之间的差别如图 3 所示。

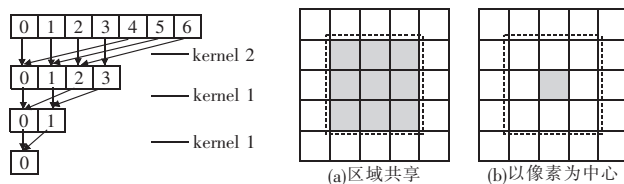


图 2 最大亮度计算

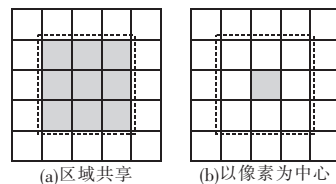


图 3 两种区域平均亮度

图 3(a)中虚线范围内的所有像素具有相同的区域平均亮度,即虚线框内九个像素的亮度均值,而图 3(b)中只有中心的像素区域具有这一平均亮度。对于前者,相邻区域平均亮度的计算具有不相关性,可以直接取一定范围内的像素计算亮度均值即可;而对于后者,相邻像素的区域平均亮度计算具有很强的相关性,为了充分利用相关性以提高运算效率,本文设计的计算步骤如图 4 所示。

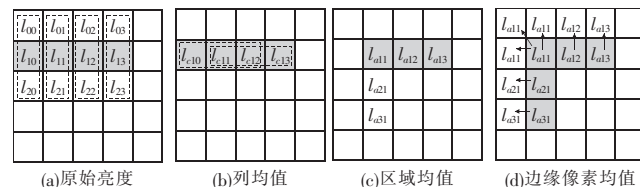


图 4 区域均值计算过程

图 4 中以 3×3 区域为例显示了以像素为中点的区域均值计算过程,算法分为三个步骤:

a) 计算列均值(column average luminance)。为了避免图

像边界判断对效率的影响,对于图像行数为 N 、区域大小为 $n \times n$ (n 为奇数) 的区域均值计算,需要计算 $\frac{n-1}{2} \sim N - \frac{n+1}{2}$ 行中元素的列均值,图 4(b) 中 l_{cij} 的计算如式(7)所示。

$$l_{cij} = \frac{1}{3}(l_{ij} + l_{(i-1)j} + l_{i(j-1)}) \quad (7)$$

b) 计算图像中心部分的区域平均亮度。同列均值计算过程一样,这一步骤也不计算边界部分的平均亮度,计算的范围是 $\frac{n-1}{2} \sim N - \frac{n+1}{2}$ 行, $\frac{n-1}{2} \sim M - \frac{n+1}{2}$ (M 为图像列数) 列的所有像素区域平均亮度。图 4(c) 中计算 l_{aij} 如式(8)所示。

$$l_{aij} = \frac{1}{3}(l_{ci(j-1)} + l_{cij} + l_{ci(j+1)}) \quad (8)$$

c) 平均亮度扩展到边界。为了获取完整的像素区域平均亮度,算法将最邻近非边界点的区域平均亮度赋值给边界部分像素的区域平均亮度,也就是在边界部分采用共享区域平均亮度。因为边界本身不能计算出以像素为中心的区域平均亮度,这种处理方法是合理的。

为了降低计算过程中对全局存储空间的访问代价,算法实现中将亮度数组映射到二维纹理,通过 GPU 自带的纹理访问缓存机制降低访存时间。

2.3 色阶映射

色阶映射的过程比较简单,对每个像素利用式(3),代入相应的参数计算最终像素的亮度。

需要注意的是,因为计算的结果是亮度值,需要进行一次转换将映射反映到 R, G, B 三个颜色通道上。对原来颜色为 (R, G, B) 、原亮度为 L_w 、映射后亮度为 L_d 的像素,最终颜色值通过式(9)计算。

$$(R' \ G' \ B') = \frac{L_d}{L_w}(R \ G \ B) \quad (9)$$

2.4 针对视屏处理的加速方法

2.4.1 纹理缓存池

在整个算法中,CPU 主要负责数据的准备,GPU 负责最大亮度、区域平均亮度和映射计算。在处理静态纹理时,纹理的读取只需要进行一次,不会对 GPU 的计算过程产生影响。但在处理视屏流时,因为 GPU 的处理时间要比纹理读取时间短,会出现 GPU 等待数据的现象,为了消除这种现象,可以采用以下方法:

a) 建立 GPU 纹理缓存池,每一块纹理缓存对应一个 GPU 全局数据指针,建立可用性数组标志数据是否已经读取完成,初始化为 false,利用多线程读取视屏流存储到 GPU 纹理缓存中,每读取一帧数据就将对应数据可用标志改为 true。

b) 调用 GPU 核函数进行计算时,将可用的纹理数据指针作为参数传递到核函数中,处理完之后重新将可用标志改为 false 以读取新的数据。

2.4.2 最大亮度更新

最大亮度计算是色阶映射算子实现中最耗时的部分,每一帧都计算最大亮度势必会对效率产生很大的影响。受到 Lauterbach 等人^[17]提出的用于对光线跟踪 BVH 的更新方法的启发,本文提出一种自适应的最大亮度更新算法。

程序保留一个当前全局最大亮度和一个像素子集最大亮度。在每一帧计算最大亮度之前,先计算出像素子集的最大亮

度,当这一亮度与前一帧的像素子集最大亮度相差超过 10% 时才进行全局最大亮度的更新,否则继续使用之前保留的最大亮度值。

3 实验结果及分析

3.1 实验结果

本文采用的实验平台为 Intel Core2 Duo E7400(2.80 GHz) CPU, GTX260 GPU, NVIDIA CUDA API 4.0 Release。实验结果如图 5 所示。本文算法与文献[7]中算法结果对比如表 1 所示。

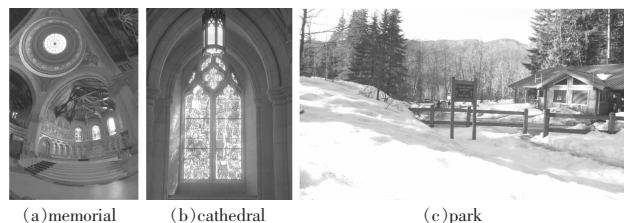


图 5 实验结果

表 1 实验结果及对比

纹理名称	纹理大小	文献[7]算法	本文算法	帧率比值
memorial	512 × 768	59	232	3.93
cathedral	767 × 1024	30	161	5.36
price western	3272 × 1280	6	-	-
park	3840 × 2160	-	17	-

注:表中 - 表示没有相关数据。

因为使用平台的不同,不能直接将本文算法与文献[8]算法在帧率上作直接的对比,但从表 1 可以看出,本文算法的平均帧率能达到文献[8]中的 4 倍,即使对超大纹理(如 park),本文算法也能具有很好的适应性。

3.2 算法分析

本文算法设计的映射过程分为最大亮度计算、区域平均亮度计算和映射计算三个步骤。

对最大亮度计算,本文算法充分利用 GPU 的多线程并行计算,并将 GPU 所不擅长的分支运算减少到最小。对大小为 $M \times N$ 的纹理,算法时间复杂度为 $\log(M \times N)$ 。

本文结合相邻像素区域平均亮度计算的相关性,利用三个步骤实现对区域平均亮度的计算。对于实验中选取的 7×7 的区域亮度计算,每个像素需要进行 12 次加法运算,相比直接计算所需的 48 次运算,效率提高了 4 倍。

整个算法是基于 GPU 实现的,所以具有良好的扩展性,算法的效率与 GPU 并行线程数成正比。

4 结束语

本文提出一种基于 GPU 的快速色阶映射算子实现算法。算法设计了高效的最高亮度计算和区域平均亮度计算算法,极大地减少了 GPU 中的分支运算,充分挖掘了 GPU 并行计算性能,对超大纹理的色阶映射也具有很好的适用性。另外文中还提出了处理视屏流时可以采用的加速方法,但对视屏流处理中的曝光度调整问题还需要进一步研究。

参考文献:

- [1] MAI Z. Tone-mapping high dynamic range images and videos for bit-depth scalable coding and 3D displaying [M]. [S. l.]: University of British Columbia, 2012. (下转第 626 页)

- ternational Journal of Computer Vision, 2001, 45(2): 83-105.
- [5] GESU V D, VALENTI C, STRINATI L. Local operators to detect regions of interest [J]. *Pattern Recognition Letters*, 1997, 18(11-13): 1077-1081.
- [6] WAI W Y K, TSOTSOS J K. Directing attention to onset and offset of image events for eye-head movement control [C]//Proc of International Association for Pattern Recognition Workshop on Visual Behaviors. 1994.
- [7] STENTIFORD F W M. An evolutionary programming approach to the simulation of visual attention [C]//Proc of Congress on Evolutionary Computation. 2001.
- [8] ITTI L, KOCH C, NIEBUR E. A model of saliency-based visual attention for rapid scene analysis [J]. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 1998, 20(11): 1254-1259.
- [9] ELAZARY L, ITTI L. Interesting objects are visually salient [J]. *Journal of Vision*, 2009, 8(3): 1-15.
- [10] GUO Chen-lei, ZHANG Li-ming. A novel multiresolution spatiotemporal saliency detection model and its applications in image and video compression [J]. *IEEE Trans on Image Processing*, 2010, 19(1): 185-198.
- [11] GAO D, MAHADEVAN V, VASCONCELOS N. On the plausibility of the discriminant center-surround hypothesis for visual saliency [J]. *Journal of Vision*, 2008, 8(7): 1-18.
- [12] SEO H J, MILANFAR P. Static and space-time visual saliency detection by self-resemblance [J]. *Journal of Vision*, 2009, 9(12): 1-27.
- [13] GUO C, ZHANG L. Spatio-temporal saliency detection using phase spectrum of quaternion Fourier transform [C]//Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2008.
- [14] BRUCE N D B, TSOTSOS J K. Saliency, attention and visual search: an information theoretic approach [J]. *Journal of Vision*, 2009, 9(3): 1-24.
- [15] ROSIN P L. A simple method for detecting salient regions [J]. *Pattern Recognition*, 2009, 42(11): 2363-2371.
- [16] GAO D, VASCONCELOS N. Decision-theoretic saliency: computational principles, biological plausibility, and implications for neurophysiology and psychophysics [J]. *Neural Computation*, 2009, 21(1): 239-271.
- [17] GOFERMAN S, ZELNIK-MANOR L, TAL A. Context-aware saliency detection [C]//Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2010.
- [18] HOU Xiao-di, ZHANG Li-qing. Saliency detection: a spectral residual approach [C]//Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2007.
- [19] HAREL J, KOCH C, PERONA P. Graph-based visual saliency [C]//Advances in Neural Information Processing Systems. 2003.
- [20] ACHANTA R, HEMAMI S, ESTRADA F, *et al.* Frequency-tuned salient region detection [C]//Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2009.
- [21] ACHANTA R, SÜSSTRUNK S. Saliency detection using maximum symmetric surround [C]//Proc of IEEE International Conference on Image Processing. 2010.
- [22] YANG J, ZHANG D, FRANQI A F, *et al.* Two-dimensional PCA: a new approach to appearance-based face representation and recognition [J]. *IEEE Trans on Pattern Analysis and Machine Intelligence*, 2004, 26(1): 131-137.
- [23] GEISLER W S, PERRY J S. A real-time foveated multi-resolution system for low-bandwidth video communication [C]//Proc of Micro Fluidic Devices and Systems. 1998.
- [24] FANG Yu-ming, LIN Wei-si, LEE B S, *et al.* Bottom-up saliency detection model based on human visual sensitivity and amplitude spectrum [J]. *IEEE Trans on Multimedia*, 2012, 14(1): 187-198.
- [25] GREENSPAN H, BELONGIE S, GOODMAN R, *et al.* Overcomplete steerable pyramid filters and rotation invariance [C]//Proc of IEEE Conference on Computer Vision and Pattern Recognition. 1994: 222-228.

(上接第 597 页)

- [2] MANN S, PICARD R W. On being undigital with digital cameras: extending dynamic range by combining differently exposed pictures [C]//Proc of IS&T's 48th Annual Conference. 1995: 422-428.
- [3] MANTIUK R, MYSZKOWSKI K, SEIDEL H P. A perceptual framework for contrast processing of high dynamic range images [C]//Proc of the 2nd Symposium on Applied Perception in Graphics and Visualization. 2005: 87-94.
- [4] REINHARD E, HEIDRICH W, DEBEVEC P, *et al.* High dynamic range imaging: acquisition, display, and image-based lighting [M]. San Francisco: Morgan Kaufmann Publisher, 2010.
- [5] STOCKHAM T G. Image processing in the context of a visual model [J]. *Proceedings of the IEEE*, 1972, 60(7): 828-842.
- [6] DRAGO F, MYSZKOWSKI K, ANNEN T, *et al.* Adaptive logarithmic mapping for displaying high contrast scenes [J]. *Computer Graphics Forum*, 2003, 22(3): 419-426.
- [7] LEE J W, PARK R H, CHANG S, *et al.* Local tone mapping using the K-means algorithm and automatic gamma setting [J]. *IEEE Trans on Consumer Electronics*, 2011, 57(1): 209-217.
- [8] BAE J, KIM K, YUN Yu-jin, *et al.* Adaptive tone-mapping operator for HDR images based on image statistics [C]//Proc of IEEE Region 10 Conference. 2011: 1435-1438.
- [9] KITAURA M, OKURA F, KANBARA M, *et al.* Tone mapping for HDR images with dimidiate luminance and spatial distributions of bright and dark regions [C]//Proc of SPIE. 2012: 3.
- [10] KIRK A G, O'BRIEN J F. Perceptually based tone mapping for low-light conditions [J]. *ACM Trans on Graphics*, 2011, 30(4): 42.
- [11] QIU Guo-ping, MEI Yu-jie, LAM K M, *et al.* Tone mapping HDR images using optimization: a general framework [C]//Proc of the 17th IEEE International Conference on Image Processing. [S. l.]: IEEE, 2010: 3129-3132.
- [12] ZHAO Han-li, JIN Xiao-gang, SHEN Jian-bing. Real-time tone mapping for high-resolution HDR images [C]//Proc of International Conference on Cyberworlds. 2008: 256-262.
- [13] PATTANAIK S, YEE H. Adaptive gain control for high dynamic range image display [C]//Proc of the 18th Spring Conference on Computer Graphics. New York: ACM Press, 2002: 83-87.
- [14] YAMAUCHI T, MIKAMI T, OUDA O, *et al.* Improvement and evaluation of real-time tone mapping for high dynamic range images using gaze information [C]//Proc of International Conference on Computer Vision. Berlin: Springer-Verlag, 2011: 440-449.
- [15] Rec. 709 [EB/OL]. http://en.wikipedia.org/wiki/Rec._709.
- [16] NVIDIA CUDA toolkit [EB/OL]. <http://developer.nvidia.com/cuda-downloads>.
- [17] LAUTERBACH C, YOON S E, TUFT D, *et al.* RT-DEFORM: interactive ray tracing of dynamic scenes using BVHs [C]//Proc of IEEE Symposium on Interactive Ray Tracing. 2006: 39-46.