基于光子映射的辐照度缓存算法①

顾晓玲，王毅刚

(杭州电子科技大学 图形图像研究所，杭州 310018)

摘　要：提出了一种全局光照计算方法，结合了两个知名的技术，光子映射和辐照度缓存。光子映射具有视点无关的优势，辐照度缓存可以快速计算间接光照，但后者是视点相关的，为了使光照缓存记录覆盖整个场景，辐照度缓存算法需要手动设置很多相机。利用这两种技术的各自优势，通过光子图来计算改进后的视点无关的辐照度缓存算法，实现了快速而准确的全局光照计算。

关键词：全局光照；光子映射；辐照度缓存

Irradiance Caching Algorithm Based on Photon Mapping

GU Xiao-Ling, WANG Yi-Gang

(Institute of Graphics and Image, Hangzhou Dianzi University, Hangzhou 310018, China)

Abstract: This paper presents a global illumination method combining two well-known techniques: photon mapping and irradiance caching. The photon mapping method has the advantage of being view independent, irradiance caching can quickly calculate indirect light, but the latter is view-dependent. To compute records covering the entire scene, the irradiance caching method has to be run for many cameras. Our method exploits the advantages of these two methods to achieve fast and accurate calculation of global illumination by computing a refined view-independent irradiance cache from a photon map.

Keywords: global illumination; photon mapping; irradiance caching

1　引言

全局光照计算是场景绘制迈向真实感的一个必要手段，早期的全局光照算法主要是光线跟踪和辐射度算法，当今最常用的全局光照算法是光子映射（Photon Mapping）和辐照度缓存(Irradiance Caching)。

Henrik Wann Jensen 在 1996 年正式提出了光子映射算法[1],该算法从光的物理属性出发，研究光子在场景中的传输，能更有效地模拟一些特殊的光学现象，如焦散(Caustics)、色彩渗透(Color Bleeding)、中间介质(Participating Media)等。由于综合了现有的各种光照技术的优点，光子映射已经成为计算全局光照最流行的算法之一[2]。

辐照度缓存最早由 Ward 在 1988 年提出[3]，通过对漫反射面的间接光照进行稀疏采样和插值，用来加速漫反射面的间接光照的计算，该算法成功地运用于 Radiance 光照模拟系统。现在几乎所有的商业渲染软件和开源渲染软件中全局光照的计算都包含了基于辐照度缓存的算法。

光子映射具有视点无关的优势，而辐照度缓存可以快速计算间接光照，本文利用光子映射和辐照度缓存的各自优势，提出并实现了一种全局光照计算算法。

2　相关工作

光子映射算法主要有两个阶段:第一阶段从光源发射一定数量的光子，并跟踪这些光子在场景中的运动情况，将与物体碰撞的光子信息存储到光子图中。第二阶段利用生成的光子图，在采样点附近查找与之邻近的光子，利用查找到的邻近光子的能量来估计采样点的反射光强。

Jensen 提出了将光子映射和辐照度缓存结合起

① 收稿时间:2010-08-27;收到修改稿时间:2010-10-01

来，将辐照度缓存算法应用到光子映射的绘制过程，作为优化光子映射算法的策略之一[4]。文献[5]对全局光子图中的所有光子进行辐照度的预计算，只需最邻近的几个光子就可以估算采样点的间接光照。文献[6]提出了光子在中间介质中传播时的计算方法。文献[7]在可编程硬件上实现了光子映射算法的加速。文献[8]实现了实时光子映射的并行系统框架，并对内存管理、资源发布进行了讨论。文献[9]实现了在 GPU 上实时构建 kd-tree 的算法。

辐照度缓存算法的思想很简单，如果在采样点附近有已经缓存的记录集，则用这些记录集的光强来插值作为采样点的光强，否则在采样点计算并存储新记录。所以，辐照度算法主要涉及三个方面：辐照度计算，辐照度存储，辐照度插值。

为了改善辐照度插值的质量，文献[10]和文献[11]介绍了辐照度梯度(Irradiance Gradients)，可以提高计算漫反射面的间接光照的精度。文献[12]在 GPU 上实现了辐照度缓存算法，实现了辐照度缓存算法的加速。文献[13]提出了使用改进的辐照度缓存算法来绘制中间介质。

辐照度缓存算法可以快速计算间接光照，但却是视点相关的，为了使光照缓存记录覆盖整个场景，需要手动设置很多相机，这需要花费很多时间。而光子映射生成的光子图有视点无关的优势，本文提出了一种全局光照算法，利用光子映射和辐照度缓存的各自优势，通过对全局光子图中存储的光子的位置来生成光照缓存记录，即实现了视点无关的辐照度缓存算法，另外，对光子进行辐照度的预计算，这样辐照度缓存算法在采样间接光照时，只需根据离交点最邻近光子的辐照度计算即可，实现了快速而准确的全局光照计算。

3 算法实现

我们的算法主要为四个部分：

(1) 创建光子图(Building Photon Map)。

(2) 对全局光子图中存储的光子进行辐照度的预计算(Precomputation of Irradiances)。

(3) 根据光子图生成光照缓存记录(Sampling Irradiance From Photon Map)。

(4) 根据生成的光照缓存记录来计算间接光照(Computing Indirect Lighting)。

图 1 描述了渲染最终图片的整体算法流程，黑框部分是本文的主要内容。

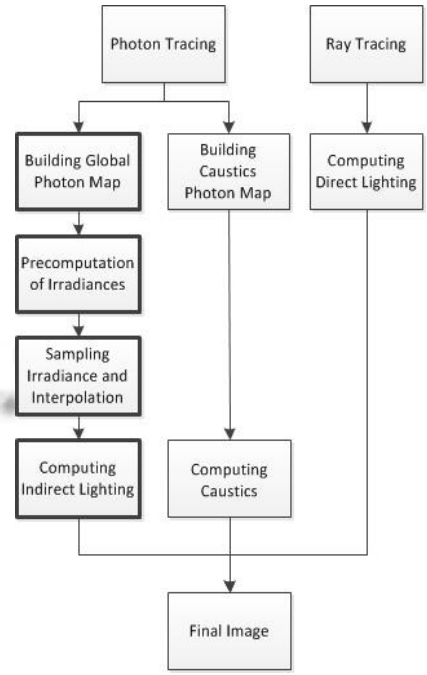


图 1 整体算法流程

3.1 创建光子图

这个阶段为光子追踪，创建全局光子图和焦散光子图。通常，光子信息包括光子的位置、入射方向、光子的能量信息等。我们的光子图存储了额外的信息，辐照度值和表面法向，因为在算法的第二个阶段要对全局光子图中存储的光子进行辐照度的预计算。

3.2 预计算辐照度

这个阶段对全局光子图中存储的光子进行辐照度的预计算。我们对光子的位置处进行密度估计，将预计算的辐照度值和表面方向存储在光子中。在一个光子的位置预计算辐照度，需要花费 O(logN),N 为光子图中的光子数，对 N 个光子进行辐照度的预计算需要花费 O(NlogN)。尽管预计算辐照度需要额外的时间代价，但对整体算法的加速效果显著。

3.3 根据光子图生成光照缓存记录

当光子覆盖所有表面，即表面被间接光照明，它们的位置有可能被作为辐照记录的位置。从一个空的缓存开始，我们通过检查光子图中的每个光子逐步增加新记录。图 2 描述了辐照度记录的创建过程。

```
GenerateIrradianceCacheFrom(Photon Map) {
    for(all photon p in Photon Map) {
        find the record set S
        if(S is empty)
        {
            compute a new record
            store the new record in octree
        }
    }
}
```

图 2　创建辐照度记录的伪代码

3.4 计算间接光照

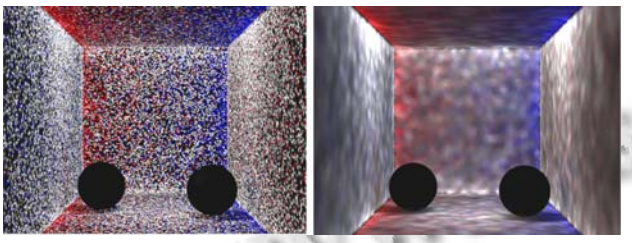
这个阶段非常简单，只需根据生成的光照缓存记录来计算间接光照。

4　算法结果与总结

4.1 算法结果

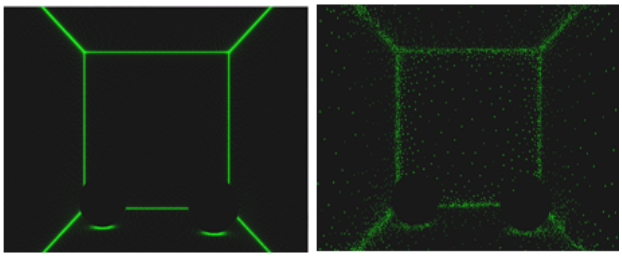
实验环境：CPU 为 Intel Core Duo E8400 3.0 GHz，2.0G 内存，显卡为 NVIDIA GeForce 9800 GT,操作系统 Windows 7,开发平台为 MS Visual Studio 2008.

图 3(a)为生成的全局光子图(全局光子数为 100000)，图 3(b)为对光子进行辐照度预计算后的效果图。对于 100000 个光子的全局光子图进行辐照度预计算需要花费 15.6 秒，虽然这个预处理步骤花费了一定的时间，但大大加速了整个算法。



(a)全局光子图　　(b)对光子进行辐照度预计算

图 3　根据光子图预计算辐照度

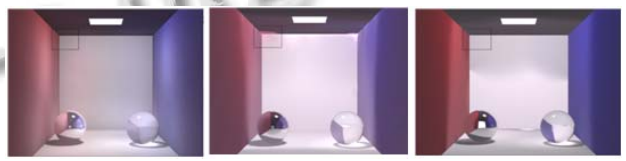


(a)基于视点生成缓存记录　　(b)基于光子图生成缓存记录

图 4　生成光照缓存记录的比较

图 4(a)为通过传统的辐照度缓存算法（基于视点）生成的光照缓存记录图(最大误差为 0.1)，图 4(b)为通过光子图生成光照缓存记录图(最大误差为 0.1)，基于光子图生成的光照缓存记录图的优点是与视点无关。

图 5 利用不同算法对同一场景进行渲染的结果比较，其中从光源发射的全局光子数为 10000 个。图 5(a)采用了传统的光子映射算法绘制，光强估计采用的光子数为 500；图 5(b)采用了传统的辐照度缓存算法绘制，最大误差为 1.0；图 5(c)采用了本文提出的算法绘制，最大误差为 1.0；图 5 的(d)(e)(f)分别是对图(a)(b)(c)矩形区域的特写。



(a)光子映射算法　(b)辐照度缓存算法　(c)本文的算法

(d)图(a)缩放　　(e)图(b)缩放　　(f)图(c)缩放

图 5　渲染结果比较

表 1 说明了改变全局光子数目对 3 种算法速度的影响。

表 1　3 种算法性能比较

全局光子数目	10000	20000	30000	40000	50000
光子映射	398.3s	555.1s	725.3s	888.1s	1028.6s
辐照度缓存	144.2s	162.8s	182.7s	193.5s	201.3s
本文算法	94.3s	163.5s	234.2s	301.5s	357.7s

观察以上实验结果，可以看到由光子映射算法渲染的图片相对真实和准确，但速度较慢，尤其是当增加光子数目时算法的性能会急剧下降。由辐照度缓存算法生成的图片，因为插值存在的误差（辐照度缓存算法存在的内在缺陷），除了在场景的角落和边缘处会有一些比较明显的错误，渲染结果基本准确，可以看到辐照度缓存算法相对光子映射算法在渲染的速度上有明显的优势，而且增加光子数目对此算法的影响不大。本文提出的算法，尽管在有些区域明亮过渡比较明显，场景的角落和边缘处的错误相对辐照度缓存要小，渲染的总体效果要比辐照度缓存算法好。由于本文算法对全局光子图

中存储的光子进行了辐照度预计算，当光子数目较小时，本文算法无论在速度上，还是在渲染的结果上，相对辐照度缓存算法更具优势，但随着光子数目的增加，辐照度缓存算法在速度上更具优势。

4.2 总结

本文利用光子映射和辐照度缓存的各自优势，通过对全局光子图中存储的光子的位置来生成光照缓存记录，即实现了视点无关的辐照度缓存算法。另外，对光子进行辐照度的预计算，这样辐照度缓存算法在采样间接光照时，只需根据离交点最邻近光子的辐照度计算即可，实现了快速而准确的全局光照计算。当光子数目较小时(对于一些简单的场景光子数目已经足够)，本文的算法渲染的图片在速度和质量更优于辐照度缓存，但随着光子数目的增加，辐照度缓存算法在速度上更具优势。由于本文实验采用的场景比较简单，在复杂场景下，本文的算法在在计算速度方面的优势应会更加明显。我们接下会针对复杂场景对现有的算法进行改进，同时尝试在 GPU 上实现现有的算法。

参考文献

1 Jensen HW. Global illumination using Photon Maps. Rendering Techniques’96. Eds. X. Pueyo and P.Schr der, 1996.

2 陈皓.基于光子映射的虚拟现实真实感渲染算法研究[博士学位论文].合肥:合肥工业大学,2008.

3 Ward GJ, Rubinstein FM, Clear RD. A ray tracing solution for diffuse interreflection. Computer Graphics (Proc. SIGGRAPH ’88), 1988,22(4):85−92.

4 Jensen HW. Realistic Image Synthesis using Photon Mapping. ISBN:1-56881 − 140 − 7. A K Peters, July 2001.

5 Christensen PH. Faster photon map global illumination. Journal of Graphics Tools, 2000,4(3):1−10.

6 Anson O, Sundstedt V, Gutierrez D, Chalmers A. Efficient selective rending of participating media. ACM International Conference Proceeding Series, 2006.153:135−142.

7 Purcell T, Donner C, Cammarano M, Jensen HW. Photon Mapping on Programmable Graphics Hardware. Proc. of the ACM SIGGRAPH/ EUROGRAPHICS conference on Graphics hardware. 2003:41−50.

8 Günther J, Wald I, Slusallek P. Realtime caustics using distributed photon mapping. Proc. of the Eurographics Symposium on Rendering, 2004:111−121.

9 Zhou K, Hou QM, Wang R, Guo BN. Real-Time KD-Tree Construction on Graphics Hardware. SIGGRAPH Asia 2008.

10 Ward G, Heckbert P. Irradiance gradients. Proc. of EGWR, 1992:85−98.

11 Krivanek J, Gautron P, Bouatouch K, Pattanaik S. Improved radiance gradient computation. Proc. of SCCG, 2005: 155−159.

12 Gautron P, Krivanek J, Bouatouch K, Pattanaik S. Radiance cache splatting: a gpu friendly global illumination algorithm. Proc. of EGSR. 2005:36−39.

13 Jarosz W, Donner C, Zwicker M, Jensen HW. Radiance caching for participating media. ACM Trans. Graph. 27, 1, 2008:1−11.

（上接第 183 页）

6 总结

本文主要研究了基于网络书签的个性化信息推荐方法，主要依据 Web 用户收藏的 Web 资源之间的关系建立 Web 用户关系网络，利用派系过滤算法进行社团结构划分，来实现社团内基于协作过滤的个性化信息推荐和社团间基于“信息桥”的个性化信息推荐，并通过实验验证了此方法在推荐中是有效的。

参考文献

1 余力,刘鲁.电子商务个性化推荐研究.计算机集成制造系统, 2004,(10):1306−1313.

2 Mobasher B, et al. Integrating Web Usage and Content Mining for More Effective Personalization. Proc. of the EC-WEB Conference. Springer, 2000:165−176.

3 张树人.从社会性软件、Web2.0 到复杂适应信息系统研究[博士学位论文].北京:中国人民大学,2006.

4 Palla G, et al. Uncovering the overlapping community structure of complex networks in nature and society. Nature, 2005,7043:814−818.

5 曾庆辉,邱玉辉.一种基于协作过滤的电子图书推荐系统.计算机科学, 2005,(6):147−150.