计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2020.20190208
57(1): 17-31, 2020

多光源绘制方法综述

刘逸凡　　徐　昆
(清华大学计算机科学与技术系　北京　100084)
(yf-liu17@mails.tsinghua.edu.cn)

A Survey on Many-Lights Rendering Methods

Liu Yifan and Xu Kun
(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Abstract　Many-lights rendering has always been an important research topic in computer graphics. It is one of the important methods to achieve global illumination effects, and it is a great demand in related industries such as games, movies, and animations, etc. However, efficient many-lights rendering is still a big challenge in both fields of off-line rendering and real-time rendering. This paper reviews the main progress of many-lights rendering methods in recent years. Among those many-lights rendering methods, their major goal is to improve the rendering efficiency. In the field of off-line rendering, we first review the algorithms of accelerating visibility test computation for reducing the average rendering time of a single light source. Then, we discuss light source clustering algorithms, and review accelerated rendering algorithms which are directly based on light source clustering. Different strategies for light source clustering are discussed, including strategies using hierarchical structures and strategies based on matrix analysis. After that, we review important sampling methods based on light clustering. In the field of real-time rendering, we introduce several rendering algorithms based on light culling. We also provide comparison and analysis of the advantages and disadvantages of various methods, and summarize the research trends and challenges of many-lights rendering.

Key words　rendering; ray tracing; direct illumination; many lights; light clustering; light culling

摘　要　多光源场景绘制一直是计算机图形学中的重要研究问题,是实现全局光照效果的重要手段之一,也是游戏、影视、动画等应用领域的重要需求.无论在离线绘制领域还是实时绘制领域,多光源场景的高效绘制仍然是一个巨大的挑战.回顾了近年来图形学在多光源场景绘制方面的主要进展,如何提高多光源绘制的效率是所有相关方法的主要研究问题.在离线绘制领域,首先介绍了如何通过加快可见性测试来提高单个光源的平均计算效率;然后,讨论了光源聚类算法,介绍了基于光源聚类的加速绘制方法,并讨论了不同的光源聚类策略,包括基于层次结构的策略和基于矩阵分析的策略;之后,介绍了基于光源聚类的重要性采样方法.在实时绘制领域,介绍了多种光源剔除绘制方法.对比和分析了各种方法的优缺点,并总结了多光源绘制的研究趋势以及面临的挑战.

关键词　绘制;光线跟踪;直接光照;多光源;光源聚类;光源剔除

中图法分类号　TP391.9

收稿日期:2019-03-26;**修回日期**:2019-11-13

基金项目:国家自然科学基金优秀青年科学基金项目(61822204)
This work was supported by the National Natural Science Foundation of China for Excellent Young Scientists (61822204).

通信作者:徐昆(xukun@tsinghua.edu.cn)

全局光照效果(global illumination)包含直接光照效果(direct illumination)和间接光照效果(indirect illumination),可以表现反射、折射、焦散(caustics)、互反射等效果,绘制结果具有很高的真实感.近年来,随着游戏、影视、动画等图形学相关产业的发展,全局光照效果成为普遍而重要的需求[1-2].

计算某点的全局光照,既要考虑光源对该点的直接光照,又要考虑来自光源的光线与其他物体表面发生1次或多次反射/折射后对该点的间接光照.在光线跟踪框架[3]中,可以通过从视点处反向追踪光路来计算到达成像平面的全局光照.如图1所示,对于成像平面上的每一个像素,从视点发射若干条射线.对于每条射线,尝试与场景中的物体求交:如相交,则可能在交点处发生反射或折射,产生新的反射/折射射线,然后继续与场景中的其他物体求交,如此递归下去.在图1中,点 x_1 的出射光亮度 L_1,由直接光照和间接光照2部分组成:通过依次计算场景中每个光源对点 x_1 的贡献并求和,得到直接光照;通过计算点 x_2 的出射光亮度 L_2 对点 x_1 的贡献,得到间接光照.

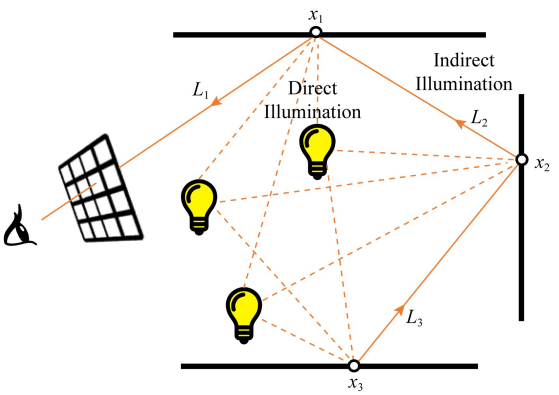


Fig. 1 Global illumination in path tracing

图1 光线跟踪中的全局光照

一方面,通过引入虚拟点光源(virtual point lights, VPLs),全局光照的计算(包括间接光照分量)可以转化为多光源下的直接光照计算;另一方面,随着工业界对于图像真实感诉求的提高,越来越多的绘制场景本身就含有大量光源:夜景的霓虹灯、火星四溅的粒子特效、复杂网格光源的模拟等.这使得多光源绘制成为计算机图形学和绘制领域的一个重要研究内容,涌现出了一大批重要方法.

理论上,多光源绘制计算的时间复杂度与场景中光源的数量成正比.如此一来,随着光源数量的增多,每个点的光照计算量也会成线性增加,造成总体绘制时间的线性增长.例如,图2是动画电影《寻梦环游记》中的一个场景,它包含了约800万个光源,如果不采取更好的优化算法,如此多光源所带来的绘制时间的增加是难以处理的.

Fig. 2 Complex illumination in Coco (© 2017 Disney•Pixar)[2]

图2 《寻梦环游记》中的复杂光照[2]

为了提高多光源场景的绘制效率,研究人员一般从3个方面入手:1)使用蒙特卡洛方法对直接光照进行重要性采样,以缩短绘制时间;2)从占据计算量主体的可见性测试入手,提高单个光源的光照计算效率;3)通过光源聚类等方法实现绘制时间对光源数量的可伸缩性(scalable),使总体绘制的时间复杂度由 $O(n)$ 变为 $O(\log n)$,其中 n 为光源数量.

根据单帧图像的绘制时长,绘制可以分为实时绘制和离线绘制.实时绘制与离线绘制的实现方法有着很大差异,对于多光源问题的处理方式也有所不同.这里,本文着重分析离线绘制的优化算法,简要介绍实时绘制的相关方法.

本文将首先阐述直接光照计算和重要性采样的基本概念;然后从加速可见性计算、通过光源聚类加速光照计算、通过光源聚类优化重要性采样等方面,对现有的离线方法进行总结和分析;最后对实时绘制中的光源剔除方法进行介绍.

1 直接光照与重要性采样

我们将物体表面需要进行绘制的点称作着色点(shading point).如图3所示,仅考虑直接光照,着色点 x 在反射方向 ω 的反射光亮度,可通过对所有光源表面 A 进行积分得到[3-4]

$$L(x,\omega)=\int_A F(y\to x\to\omega)\mathrm{d}y, \tag{1}$$

其中,积分项 $F(\cdot)$ 定义为

$$L_e(y\to x)B(y\to x\to\omega)V(y\leftrightarrow x)G(y\leftrightarrow x), \tag{2}$$

其中,y 为光源表面一点.$L_e(y\to x)$ 为 y 朝 x 发射的光亮度.$B(y\to x\to\omega)$ 为双向反射分布函数

(bidirectional reflectance distribution function, BRDF),描述 x 处的材质反射性质.$V(y\leftrightarrow x)$ 是二值可见性函数:x,y 之间可见为 1,不可见为 0.几何项

$$G(y\leftrightarrow x)=\frac{\cos\theta_y\ \cos\theta_x}{d^2(y,x)},\tag{3}$$

其中 $d(y,x)$ 为 x,y 之间的欧几里得距离;θ_x 为 x,y 连线与 x 法向之间的夹角;θ_y 为 x,y 连线与 y 法向之间的夹角.

由于式(1)没有解析解,一般使用蒙特卡洛方法将其近似为 N 次采样的积分项的加权平均:

$$L(x,\omega)\approx\frac{1}{N}\sum_{i=1}^{N}\frac{F(y_i\rightarrow x\rightarrow\omega)}{p(y_i\mid x,\omega)},\tag{4}$$

其中,$p(y\mid x,\omega)$ 表示给定着色点 x 和方向 ω,采样光源上一点 y 的概率密度函数(probability density function, PDF).概率密度函数与积分项越相似,蒙特卡洛估计的方差就越小.当概率密度函数正比于积分项时,方差为 0,此时,式(4)可以看作等式.这种采样方法也称作重要性采样(importance sampling).

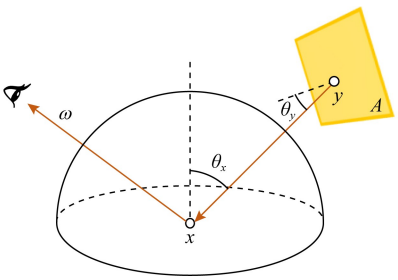


Fig. 3 Illustration of direct illumination

图 3 直接光照示意图

一般地,尤其是对于非环境光源,采样光源上一点 y,通常分解为 2 步[4]:先采样一个光源 l,再在光源上采样一个点 y.则式(4)变换为

$$L(x,\omega)\approx\frac{1}{N}\sum_{i=1}^{N}\frac{F(y_i\rightarrow x\rightarrow\omega)}{p(l_i\mid x,\omega)p(y_i\mid l_i,x,\omega)},\tag{5}$$

关于在给定几何形状的光源上进行点的采样这一问题(即得到 $p(y\mid l,x,\omega)$),许多技术已经提供了近似最优的解决方法[4-5],因此不再是研究重点.而如何合理对光源本身进行采样(即得到 $p(l\mid x,\omega)$),是直接光照计算中一个开放性的研究问题.

在对光源进行重要性采样时,采样的概率分布与光源对场景的贡献越相近,则生成的图片噪点越少.为了提高绘制质量,往往需要为每个着色点分别计算光源采样的概率密度函数:估计每一个光源的重要性程度,从而形成离散的概率分布.关于光源重要性的估计,理想情况是考虑到式(2)中的所有项:发射光亮度、BRDF、可见性和几何项.但为了保证计算效率,往往只能考虑其中的一部分.应该考虑哪几项?又如何进行快速估计?这些问题是重要性采样研究中的主要研究目标.

下面,本文将从提高单个光源的光照估计效率和实现光照估计的亚线性复杂度这 2 个角度,介绍离线绘制对于多光源场景的解决办法.

2 加速可见性计算

2 点之间可见性的判断占据整个绘制计算量的主体.许多工作都是从可见性入手,提高绘制效率.

由于暗光源对场景整体贡献较小,1994 年 Ward[6] 提出在直接光照计算中对暗光源跳过可见性测试以加速绘制.1996 年 Shirley 等人[5] 将该思想拓展到重要性采样中,只对亮度大的光源进行采样.由于不考虑暗光源,这些方法造成了绘制结果的有偏性.

1995 年 Jensen 等人[7] 的方法和 2000 年 Keller 等人[8] 的方法通过扩展光子图(photon map)方法来减少绘制时的可见性计算.然而对于大规模场景,产生的光子常常不处于视点重要区域,难以取得理想效果.

1999 年 Hart 等人[9] 利用图像空间的连贯性提前记录每个“像素-光源”对之间的潜在遮挡物体,后续计算着色点可见性时只需考虑这些记录的潜在遮挡物体.2002 年 Fernandez 等人[10] 将该思想扩展到场景空间.他们对场景分块,预先将区域块与光源之间的可见性关系分为 3 种进行记录:完全可见、完全不可见、不确定.在绘制时,只对那些相对着色点所在区域块可见性关系为不确定的光源进行可见性测试.2003 年 Wald 等人[11] 将类似方法用到了计算采样权重上.

2009 年 Dong 等人[12] 提出了 VPLs 框架下的可见性加速计算方法.他们用 K-Means 方法将虚拟点光源聚类为虚拟面光源,采用虚拟面光源计算可见性的同时,仍使用虚拟点光源进行其他项的计算.

2013 年 Popov 等人[13] 通过自适应分组缓存可见性测试结果来加速可见性计算.如图 4 所示,基于双向路径追踪(bidirectional path tracing)框架,当若干路径的光线子路径端点位于相近区域 $A(\bar{p}_L)$,且视线子路径端点也位于相近区域 $A(\bar{p}_E)$ 时,这些路径会被分到一组,分享相同的可见性结果.为避免

不必要的计算,缓存操作不是在预处理阶段完成,而是在绘制过程中自适应地完成:若当前路径所在组别没有已缓存的可见性结果,则进行可见性测试并记录;否则直接使用缓存结果.

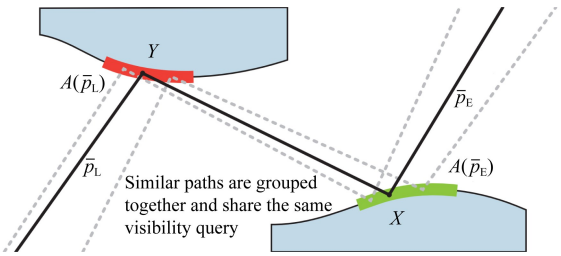


Fig. 4 Algorithm overview of ref [13]

图 4 文献[13]的算法总览

相比于缓存 1 对 1 的可见性,Ulbrich 等人[14]通过使用阴影图记录 1 对多的可见性结果.他们在物体表面离散地分布一些记录点,每个记录点存储 1 张阴影图.估计任意 2 点间的可见性时,寻找最近的阴影图,然后将 2 点间的距离与阴影图中的相应值进行比较.

这类缓存记录的方法,通过牺牲一定的存储,避免了一部分可见性测试的重复计算,从而提高了计算效率.然而对于复杂的场景,由于物体表面的相似性不高,缓存的方法往往难以得到很好的效果,并且会占用大量存储.

同年,Billen 等人[15]提出一种基于随机选取的可见性估计算法:计算 2 点之间可见性时,利用蒙特卡洛方法随机选取一部分场景物体进行测试.理论上,2 点之间的可见性值,是分别只考虑单个潜在遮挡物体时可见性值的累乘.该算法的核心思想是将累乘分解为累加,然后进行蒙特卡洛采样.例如,当只存在 2 个潜在的遮挡物体 A 和 B 时,考察点 x 和点 y 之间的可见性 $V(x,y)$,它等于 $V_A(x,y)\cdot V_B(x,y)$.通过分解得到:

$$V(x,y)=V_A(x,y)\times V_B(x,y)=V_A(x,y)+V_B(x,y)+(\overline{V_A(x,y)}\times\overline{V_B(x,y)}-1), \quad (6)$$

其中,标记 $\bar{a}=1-a$.这样一来,原本的乘式被分解为 3 项的和,可得到该式的蒙特卡洛估计:

$$\widetilde{V}(x,y)=\begin{cases}\dfrac{V_A(x,y)}{p_1}, & \text{以概率 } p_1 \text{ 选取},\\ \dfrac{V_B(x,y)}{p_2}, & \text{以概率 } p_2 \text{ 选取},\\ \dfrac{\overline{V_A(x,y)}\times\overline{V_B(x,y)}-1}{p_3}, & \text{以概率 } p_3 \text{ 选取},\end{cases} \quad (7)$$

假设采样概率 p_1,p_2,p_3 均为 1/3,相比原来需要测试 2 个物体,现在平均只需要测试 $(1+2+3)/3=1.33$ 个物体,减少了可见性测试的次数.

2014 年 Billen 等人[16]进一步提出利用蒙特卡洛估计来降低可见性测试中潜在遮挡物体的平均网格复杂度.在测试 2 点之间的可见性时,随机地选择遮挡物体的简化模型或原始网格模型.与传统方法相比,该方法并不具有太多优势.主要因为,很多情况下简化网格和原始网格的可见性测试的时间消耗相差不大.

上面 2 种算法虽然在效果提升上不是十分显著,但巧妙地将蒙特卡洛估计运用到可见性计算中,非常具有启发意义.

表 1 从准确性、存储占用、应用范围 3 个方面对上述加速可见性计算的工作进行了分析总结.本节

Table 1 Analysis of Visibility Test Acceleration Methods

表 1 可见性测试加速方法比较

Year	Reference	Accuracy	Additional Memory	Application
1994	Ref [6]	Approximation	Little	Illumination Calculation
1996	Ref [5]	Approximation	Little	Importance Sampling
1995	Ref [7]	Approximation	Large	Illumination Calculation
2000	Ref [8]	Approximation	Large	Importance Sampling
1999	Ref [9]	Exact	Large	Illumination Calculation
2002	Ref [10]	Exact	Large	Illumination Calculation
2003	Ref [11]	Approximation	Large	Importance Sampling
2009	Ref [12]	Approximation	Little	Illumination Calculation
2013	Ref [13]	Approximation	Large	Illumination Calculation
2013	Ref [14]	Approximation	Large	Illumination Calculation
2013	Ref [15]	Unbiased	Little	Illumination Calculation
2014	Ref [16]	Unbiased	Little	Illumination Calculation

介绍的算法,通过加速可见性测试降低了单个光源的计算时间消耗,对于小规模光源场景可以取得较好的效果.然而,绘制时间与光源数量成线性的复杂度并没有降低,随着场景中的光源数量继续增多,仅仅加速可见性测试难以显著提高计算效率.这时,常见的做法是将光源聚类,相似的一组光源用一个单独的大光源替代,从而使得绘制时间相比光源数量的复杂度从线性降为对数.下面,我们将首先介绍光源聚类的基本方法,然后介绍基于光源聚类的重要性采样方法.

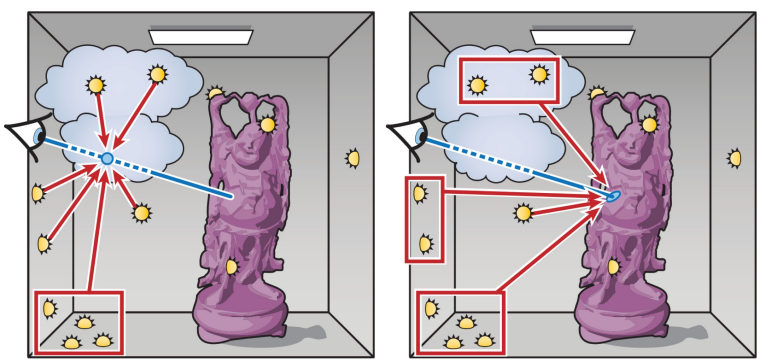
3 使用光源聚类加速直接光照计算

场景中每个光源对绘制结果的贡献往往是不均匀的.部分光源有着较低的重要性,例如,当光源被遮挡时,或光源距离着色区域很远;同时,部分光源是非常重要的,比如对于绘制结果中的高光有贡献的光源,我们需要精确计算它们的光能贡献.通过利用场景中光源重要程度的不均匀性和光源之间的相似性,对光源进行自适应的聚类,并以类为单位进行直接光照计算,是光源聚类加速方法的主要思想.

使用光源聚类来加速直接光照计算的相关研究主要是基于 VPLs 算法框架.VPLs 算法源自 Keller 的即时辐射度方法[17],其核心思想是将复杂的全局光照计算问题近似为大量虚拟点光源(VPLs)下的直接光照计算问题.为了确保图像的真实性,VPLs 算法往往需要生成数量巨大的虚拟点光源.此时,通过依次遍历光源来计算直接光照的蛮力方法不再适用,需要有更优化的算法来降低线性时间复杂度,实现可伸缩性.近 10 多年,出现了许多研究成果,其中 2014 年的 1 篇综述[18]详细介绍了 VPLs 算法的原理以及相关算法,感兴趣的读者可以进行查阅.本文将着重介绍 2014 年之后的相关工作.

需要注意的是,基于 VPLs 框架的光源聚类不一定能直接泛化到一般情形下的直接光照计算中.原因主要有 2 点:1)VPLs 框架中的虚拟光源往往是点光源,基于此的一些光源聚类方法可能不适用于球光源、面光源等其他光源类型;2)VPLs 算法的应用场合决定了它往往不需要过多的像素采样数,基于此的一些光源聚类算法会预先计算和存储所有着色点,用于后续的绘制过程.当像素采样数增多时,这种算法会占用大量的存储空间,不适合泛化到一般情境中.

光源聚类的方法大致可分为 2 类:基于树的方法和基于矩阵的方法.不同聚类方法的主要区别在于,估计光源重要性的标准以及聚类的形式存在差异.但总体来说,光源聚类的核心思路都是将相似的光源聚成一类,形成一个光源簇(light cluster).每个光源簇的总体光能贡献,可以通过将该光源簇看作一个更大、更亮的光源来进行近似估计,如图 5 所示:



(a) Shade a point in the participant media (b) Shade a point on the surface

Fig. 5 Lights clustering when evaluating VPLs lighting[18]

图 5 估计 VPLs 光照时的光源聚类[18]

3.1 基于树的光源聚类

1998 年 Paquette 等人[19]最早提出通过构造光源的层次结构树来加速直接光照的计算,但是没有考虑到遮挡和阴影.2005 年基于 VPLs 框架的 Lightcuts 算法[20]是首个实用的可伸缩性算法.Lightcuts 算法分为 3 步:1)基于空间位置和朝向的相似性,将光源用二叉树组织起来,形成光源树(light tree),即叶节点是单独的光源,中间节点代表一个光源簇,包含其下所有叶节点的光源.2)对于每个着色点,利用光源树搜索一个合适的光源聚类,由于一个光源聚类对应于光源树的一条分割线,所以又称其为光源割(lightcut),如图 6 所示.光源割的计算,是从根节点

出发自顶而下地递归遍历光源树:当遍历到某一节点时,计算当前节点所代表的光源簇对着色点的光能贡献估计值和估计值的误差上界.若误差上界小于设定的阈值,停止当前节点的遍历;否则,递归地同时考察它的 2 个子节点.3)得到光源割后,着色点的直接光照是通过依次估计光源割中每个光源簇对其的光能贡献,并求和得到.这样一来,计算时间与光源割大小成线性关系,而光源割的大小并不随光源数量的增加而线性增大,因此实现了伸缩性,并大大缩短了绘制时间.Lightcuts 算法中,光源树的构造采用的是简单的自顶而下的方法.为了得到更高质量的光源树,也有一些关于如何自底而上建树的工作[21-22].2012 年 Davidovič等人[23]提出了 Lightcuts 算法的 GPU 实现.

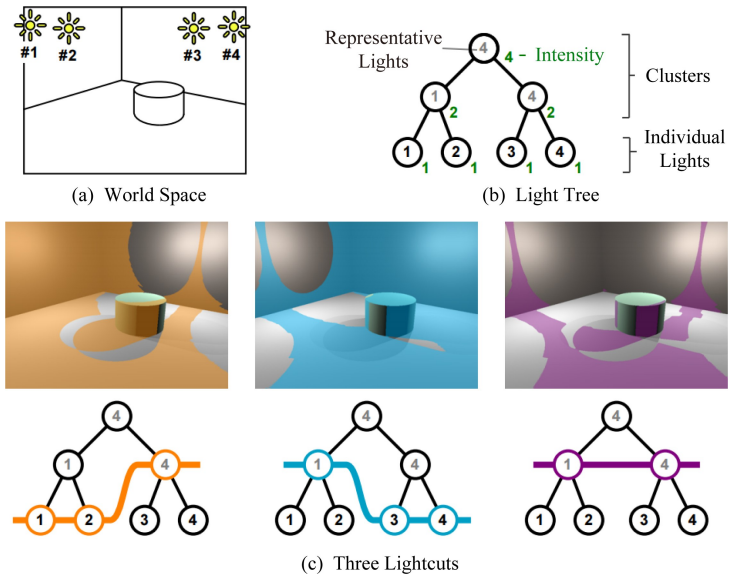


Fig. 6 A light tree and three example lightcuts[20]

图 6 1 个光源树和 3 个光源割[20]

Lightcuts 算法需要为每个着色点选择一个合适的光源割.当光源数量很多且分布复杂时,从光源树中搜索合适的光源割也需要花费不少的计算时间.针对这一问题,一些工作提出了改进方法.

2006 年 Walter 等人[24]将 Lightcuts 算法扩展,提出多维 Lightcuts 算法(multidimensional lightcuts),以便能够处理高维空间的绘制积分,例如景深、运动模糊、空间抗锯齿等.对于每个像素,他们将高维积分离散成一系列着色点的光照求和.由此产生的着色点数量非常多,对每个着色点都重新计算一个光

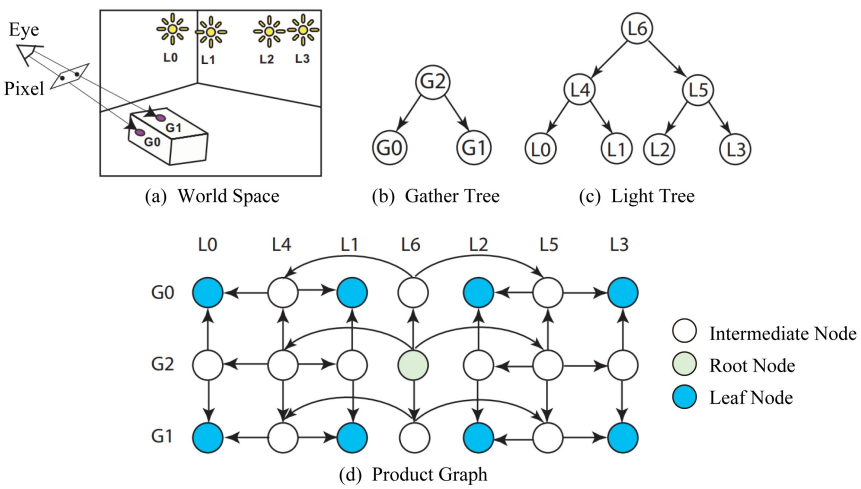


Fig. 7 Product graph [24]

图 7 乘积图[24]

源割将带来巨大的时间成本.他们分别将同一个像素内的着色点用二叉树组织起来,并通过使用图 7 所示的乘积图(product graph)来隐式地表达“着色点-光源”对的层次结构,从而提高了效率.

2013 年王光伟等人[25]提出对所有的着色点进行空间聚类,同一类中的着色点使用同一个光源割作为初始光源割,然后对同一类中的每个着色点进一步细化光源割.这样的做法避免了每次都从光源树的根节点开始搜索光源割,一定程度上减少了计算量.

2015 年 Bus 等人[26]提出的 IlluminationCut 沿用多维 Lightcuts[24]的思路,但不再只考虑同一像素内的着色点,而是将所有着色点都用一个二叉树组织起来,并用乘积图(图 7 所示)隐性地对“着色点-光源”对进行聚类.这样一来,只需遍历一次乘积图(等价于同时遍历一次着色点树和光源树),即可得到每个着色点的光源割,平摊了计算时间.

2016 年 Rehfeld 等人[27]提出一种 Lightcuts 插值算法.他们利用光源割的局部连贯性,先选取一部分着色点按照原始的 Lightcuts 算法计算光源割,然后通过插值来得到其余着色点的光源割.如此一来,减少了光源割的计算次数,提高了计算效率.

上面介绍的算法都是基于 Lightcuts 的聚类思路,即先建立光源二叉树,从光源树根节点自顶向下通过误差上界标准来得到光源割.除此之外,一些工作提出了其他的基于树的聚类方法.

2015 年 Bus 等人[28]基于 VPLs 框架提出了一种不依赖视角信息的点光源聚类算法.它的核心思路是:1)在预处理过程中,针对每个点光源,将其余点光源聚类;2)绘制过程中,对着色点计算直接光照时,使用最近邻的点光源的聚类作为该着色点的初始聚类,然后进行细微调整.这样一来,由于预处理过程中的聚类操作只依赖场景物体信息和光源信息,视点的变换不需要重新进行聚类操作,节省了大量的计算时间.在具体的聚类方法上,他们以文献[29]作为理论依据,使得聚类后的光源簇与对应点满足完全分离(well-separated)标准.如图 8 所示,这保证了对应点到光源簇内任意一点的距离和朝向都是相似的.然而,由于这种聚类标准没有考虑到着色点的材质信息,该算法更适用于漫反射表面.另外需要注意的是,VPLs 框架中,虚拟点光源的分布与场景几何分布非常接近,因此对着色点的光源聚类使用最近邻点光源对应的光源聚类来代替在直观上是合理的.但这一思路不易推广到光源分布与场景几何分布不一致的一般情形.

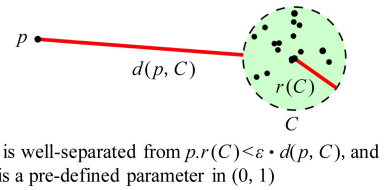


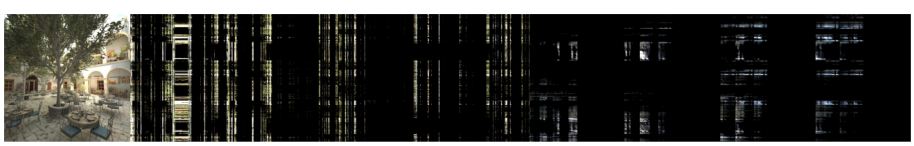
Fig. 8 Definition of well-separated[28]

图 8 完全分离的定义[28]

2018 年 Maria 等人[30]基于 Bus 等人[28]算法做了进一步的改进,在聚类时考虑了可见性.他们提出在构造完全分离对分解(well-separated pair decomposition, WSPD)结构时直接查询点对之间的可见性,这样虽然增加了 WSPD 结构的构造时间,但总体上节约了 45%左右的绘制时间,效果提升明显.

3.2 基于矩阵的光源聚类

这一类方法,将多光源场景的绘制问题用光能传输矩阵(如图 9 所示)来形式化地表示:矩阵的行索引指向着色点,列索引指向光源,第 i 行第 j 列的元素表示第 j 个光源对第 i 个着色点的光能贡献.第 i 行所有元素之和代表了所有光源对第 i 个着色点的总贡献.朴素的蛮力算法需要计算矩阵中每个



(a) Scene 1

(b) Scene 2

Fig. 9 The light transport matrices of two complex scenes (subsampled from the original)[31]

图 9 2 个样例场景的光能传输矩阵(欠采样之后的结果)[31]

元素的值,然后将它们沿着列方向进行累加以得到每个着色点的光照.当存在 m 个着色点、n 个光源时,蛮力算法所需的时间复杂度为 $O(mn)$.为了减少时间成本,这类方法通常的做法是仅计算矩阵的部分元素再估计矩阵的其余部分,而不是直接计算全部元素.需要注意的是,此类方法常常需要占据较多的存储空间,当像素数量非常大时,往往不太适用.

2007 年的 MRCS(matrix row-column sampling)算法[32]是首个基于矩阵的多光源绘制方法.他们观察到,由于着色点以及光源的空间连贯性,光能传输矩阵往往是一个低秩矩阵(例如图 9 中大部分区域都是黑色),它的信息可以通过计算低维矩阵来近似.算法的主要思路如图 10 所示:首先对传输矩阵的行做降采样,把这些行合并为一个简化矩阵(reduced matrix),然后对这个简化矩阵的列索引(即对应的光源)做聚类.根据传输矩阵的低秩假设,简化矩阵的列聚类结果可以近似为传输矩阵的列聚类结果.接下来在每一列簇中随机选择一个代表列进行绘制(绘制结果需要乘以列簇大小)来近似整个列簇的值.最后通过累加这些代表列即可得到最终绘制结果.该算法的优点在于,部分计算步骤可以用阴影图方法实现,通过 GPU 运算提高了计算效率.

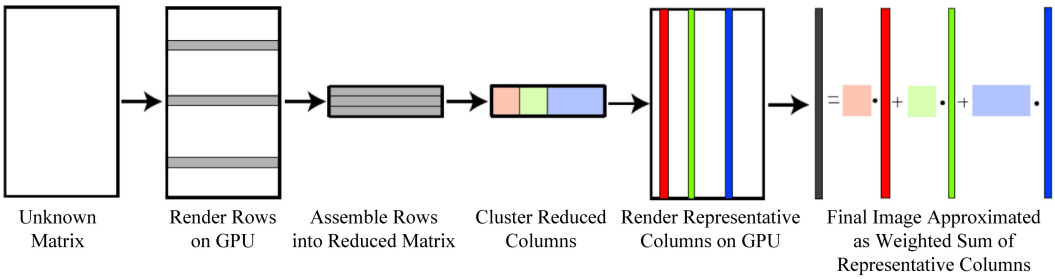


Fig. 10 Overview of MRCS algorithm[32]

图 10 MRCS 算法总览[32]

将 MRCS 与 Lightcuts 算法进行比较:前者让所有着色点共享相同的光源聚类,平摊了聚类计算量,但容易忽视只对局部区域有贡献的光源,难以捕捉光照细节;后者对每个着色点都分别进行光源聚类,虽然能捕捉局部光照细节,但存在不必要的重复计算,时间成本高.2011 年提出的 LightSlice 算法[31]结合了两者的思想,先对所有着色点划分一个粗糙的光源聚类,然后以此为起始点,对每一部分着色点继续细分光源聚类.

2015 年 Huo 等人[33]通过求解一个矩阵恢复问题来加速聚类之后的计算效率.他们观察到,即使采用之前的 LightSlice 等方法进行聚类,最后要绘制的矩阵仍然很大,需要不少的计算时间.由于可见性测试是整个计算过程中最耗时的部分,他们决定从这部分入手.他们提出只对一小部分的矩阵元素的可见性进行精确计算,其余部分采用矩阵分离技术[34]恢复得到.具体的算法流程如图 11 所示:1)将矩阵按行聚类得到一系列切片(slice),对每个切片按列聚类,在每个列簇中选择一个代表列合并成简化矩阵.计算简化矩阵中每个元素不包含可见性的光照值,由于不包含可见性,这部分的计算很快.2)离散采样一些元素进行可见性测试,并利用可见性的局部性特点大致估计其余未采样元素的可见性.3)得到的矩阵可看作是低秩目标矩阵(希望恢复的矩阵)与稀疏误差矩阵(估计值与实际值之差)的和,通过矩阵分离算法,恢复得到目标矩阵.

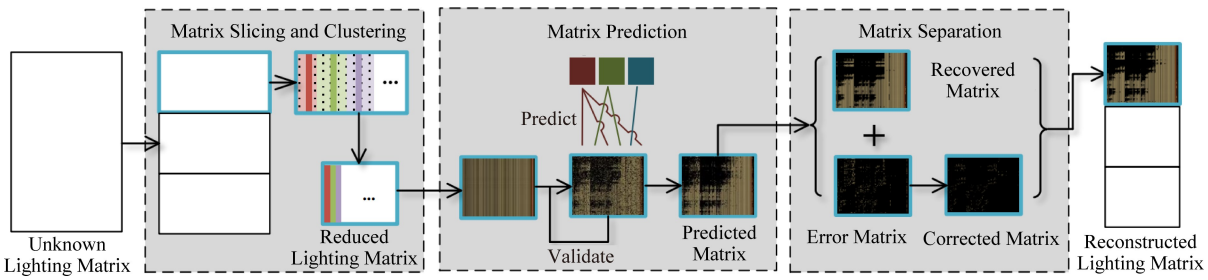


Fig. 11 Algorithm overview of ref [33]

图 11 文献[33]的算法总览

在 VPLs 框架中,为了生成云、雾等体绘制效果,一些工作将虚拟点光源替换为虚拟线光源(virtual ray lights, VRLs)[35]或虚拟束光源(virtual beam lights, VBLs)[36].学者们提出了 LightSlice 的改进算法[37]和自适应的矩阵恢复方法[38]以应对这类光源.

表 2 从存储占用、GPU 友好性、对一般情形的可泛化性这 3 个方面对基于树的光源聚类方法和基于矩阵的光源聚类方法进行对比.相比基于树的算法,基于矩阵的聚类算法由于需要构建光能传输矩阵,内存占用量大.但由于传输矩阵的一整列能通过阴影图的方法进行绘制,基于矩阵的方法 GPU 实现方便.基于矩阵的方法往往需要提前知晓并存储所有着色点,可泛化程度较低.

Table 2 Analysis of Light Clustering Methods

表 2 光源聚类方法比较

Method	Memory	GPU-Friendly	Generalization Ability	Typical References
Tree-Based	Low	No	Easy	Lightcuts[20], MdLightcuts[22], IlluminationCut[26]
Matrix-Based	High	Yes	Hard	MRCS[31], LightSlice[32]

总的来说,本节所介绍的方法均没有使用重要性采样,这样虽然减少了噪声的引入、降低了时间成本,但造成了图像结果的不准确性和有偏性.光源聚类也可以用来加速重要性采样的概率分布的计算,从而提高计算效率,这部分工作将在第 4 节进行介绍.

4 基于光源聚类的重要性采样

当光源数量比较多时,如果对每个着色点都遍历一遍所有光源来计算采样概率分布,会花费巨大的时间成本.对此,许多工作将光源聚类之后再进行重要性采样,使得直接光照采样过程由 2 步变为 3 步:

1) 采样一个光源簇 C;

2) 在光源簇 C 的内部采样一个光源 l;

3) 在光源 l 上采样一个点 y.

由于光源簇的数量远少于光源数量,建立光源簇的采样概率分布只需要消耗亚线性的时间复杂度;位于同一光源簇内部的光源往往很相似,因此在光源簇内部采样光源可以使用较简单的采样策略,例如均匀采样或只根据光源发射光亮度采样.如此一来,通过聚类的方法,对每个着色点进行光源采样的时间复杂度与光源数量呈亚线性,实现了伸缩性.

为了在直接光照采样中同时考虑光源光亮度(这里代指光源发射光亮度与几何项的乘积)和 BRDF 的信息,常常使用 1997 年由 Veach[39]提出的多重重要性采样 (multiple importance sampling, MIS)方法:分别按照光源光亮度和着色点 BRDF 的分布进行直接光照的采样计算,再将结果进行加权平均.这种方法虽然实现简单方便,但会造成采样的浪费.

2007 年 Akerlund 等人[40]在计算光源簇的重要性时直接考虑了 BRDF 信息.他们假设着色点的 BRDF 值在同一个光源簇内的变化非常平滑且可以忽略,从而通过在光源簇内随机选择一个光源计算 BRDF 值来作为权重,乘以该光源簇的光亮度估计,作为采样的重要性.这种随机选取光源簇内部光源进行 BRDF 信息估计的方法计算量虽小,但精确度较低.

BRDF 常常具有高频特性,在光源簇内平滑变化的假设往往难以成立.2009 年 Wang 等人[41]提出了更加实际的结合 BRDF 信息的光源簇采样方法.如图 12 所示,计算光源簇的重要性时,在着色点位置按照 BRDF 的分布采样若干条射线,记录与每个光源簇所在包围盒发生相交的射线数量,以此近似 BRDF 在光源簇内的积分,将其再除以包围盒相对于着色点的立体角,得出 BRDF 在光源簇内的均值,最后乘以光源簇的光亮度估计,作为该光源簇的重要性.这种 BRDF 的估计方法,相比 Akerlund 等人[40]的方法更加精确,但它依赖于 BRDF 采样射线

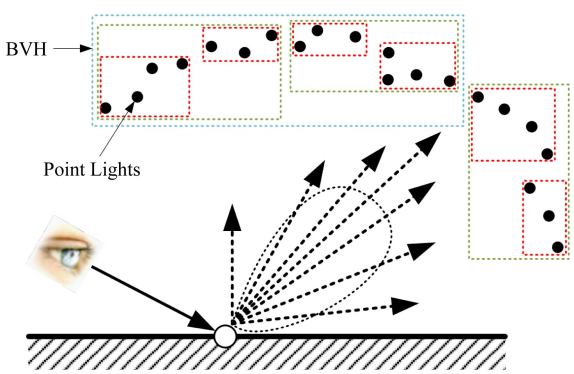


Fig. 12 Detect the number of BRDF sample intersects for each cluster[41]

图 12 检查每个光源簇与 BRDF 采样光线发生相交的数量[41]

的命中率,并且难以拓展到在光源簇内部采样光源的过程中.

2013 年提出的 VisibilityCluster 方法[42] 使用基于矩阵的聚类思想,在光源簇采样中考虑可见性因素.他们按图 13 所示的步骤来估计可见性:首先,根据几何相似性分别对光源和着色点进行聚类;然后,随机进行可见性测试,估计每个小矩阵块的平均可见性;再通过分离光源簇,进一步细分矩阵块;在估计得到可见性信息之后,结合前面上述采样方法[41],即可在光源簇采样中同时考虑光源光亮度、BRDF 和可见性 3 项.然而,基于矩阵的聚类思想需要占用大量的存储,难以适用于大规模场景的绘制.

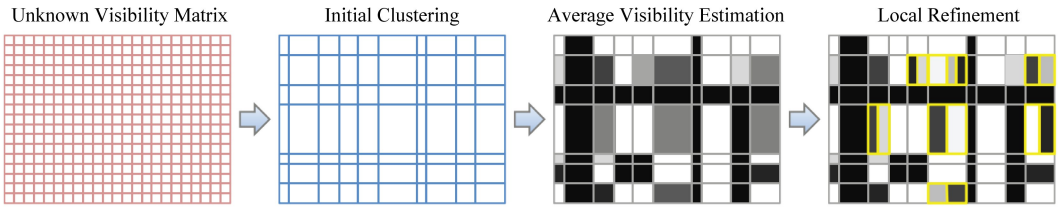


Fig. 13 Construction of VisibilityClusters[42]

图 13 可见性簇的构建[42]

2006 年 Donikian 等人[43] 提出在绘制过程中迭代渐近地学习光源簇的采样概率分布.他们首先用 Lightcuts 算法[20] 对光源聚类,并将图像空间划分为若干块.在绘制过程中,通过极大似然估计迭代地学习光源簇对每个块以及每个像素的重要性,形成相对块的采样概率分布和相对像素的采样概率分布.每一次迭代后,按照一定的经验性权重,将相对块的采样分布、相对像素的采样分布以及均匀采样分布进行加权平均,作为下一次迭代中光源簇的采样概率密度函数.对于光源簇内部,他们根据光源发射亮度进行采样.对于常见的光源,其发射亮度与着色点无关,只需计算 1 次,不影响算法的可伸缩性.该算法的主要缺点在于,3 个概率分布的结合权重是经验性的设定,不够鲁棒和可靠.

2016 年 Vévoda 等人[44] 将上面的思想从图像空间扩展到场景空间:他们将整个绘制场景划分为均匀的网格,对每个网格块单独进行自适应的光源聚类,并以此进行重要性采样.为了避免不必要的计算,他们延迟到绘制过程中才进行聚类:对每个着色点,判断其所属网格块是否已经进行了光源聚类,如果没有则立即进行聚类操作.当区域块包含聚类结果后,简单估计每个光源簇到着色点的光能贡献,以此生成离散的概率分布来采样光源簇.

2016 年 Pharr 等人[4] 也给出了一种针对多光源采样问题的实现.他们也将绘制场景划分为均匀的网格,与前面方法不同的是,他们没有选择聚类,而是在绘制过程中自适应地构建对所有光源的采样概率分布:在着色点所属区域块中随机采样一些点,简单计算每个光源到这些采样点的光能贡献,从而形成对该区域块的光源采样分布.该算法实现起来很简单,但由于没有进行光源聚类,当光源数量增多时,每个区域快的光源采样分布的构造都需要花费大量时间,且采样没有考虑可见性以及 BRDF 的信息.

2018 年 Vévoda 等人[45] 继续研究改进他们在 2016 年提出的方法[44],提出使用贝叶斯回归模型在线学习每个光源簇的采样重要性.与之前相同,他们首先使用类似 Lightcuts[7] 的算法将光源用二叉树组织起来,然后对场景区域均匀划分,在绘制过程中对每个场景块自适应地进行光源聚类.不同的是,类似 Donikian 等人的方法[43],他们也在绘制过程中渐近地学习光源簇的采样概率分布.

1) 通过理论推导,他们发现光源簇的采样重要性既与光源簇内部光源采样点的光能贡献的均值有关,又与光能贡献的方差有关.

2) 他们观察到,不考虑 BRDF 信息和采样点与着色点之间的距离时:若可见性为 1,光源簇内采样点的光能贡献近似平滑,可以用高斯函数来拟合;若可见性为 0,光能贡献为常值 0,可以用狄拉克 δ 函数来拟合.因此,光源簇对着色点的光能贡献可以建模为高斯函数与狄拉克 δ 函数的混合模型.

3) 对每个着色点和每个光源簇:参考 Lightcuts 算法对光源簇简单估计的重要性作为先验,用当前着色点所属场景块内记录的其他已绘制的着色点的光照信息作为后验,通过最大后验估计推断模型得到均值与方差,进而计算出光源簇的采样重要性.

该方法的主要优势有 2 点:一是估计光源簇的重要性时,不仅考虑了光能贡献的均值,还考虑到光能贡献的方差,即内部光源差异性越大的光源簇应

该更高概率地被采样到;二是通过渐进式学习,光源簇的重要性中包含了可见性信息,在遮挡关系复杂的场景中也能够取得很好的绘制效果.

同年,Estevez 等人[46]从另一个角度对多光源的重要性采样问题进行研究.算法的基本步骤非常简洁:对光源建树,然后对每个着色点采样一个光源.他们的贡献主要有 3 点:

1) 对光源进行层次结构的组织时,不再采用以往方法通常使用的 Lightcuts[7]建树方法,而是提出了表面积朝向准则(surface area orientation heuristic, SAOH)来引导光源层次结构的构建,使得相似空间位置和朝向的光源位于同一节点之下,生成的光源树质量更好,并且能够很好地适用于网格光源(mesh lights).

2) 相比计算所有光源的重要性,构建概率密度函数进行采样,他们提出通过遍历光源树来采样光源,即从根节点出发估计左右子节点的重要性,随机采样一个子节点,如此递归直到抵达叶节点.这样一来,光源采样消耗的时间与光源数量从线性关系降低为对数关系,实现了可伸缩性.

3) 对于深度较低的光源树节点,其包含的光源差异可能较大,由此估计的重要性程度会存在不准确性;另外,当着色点位于某个节点包围盒内部时,也无法较好地估计该节点的重要性程度.对于这些问题,他们提出在遍历光源树时,先不采样子节点,而是进行分离(split)操作,同时对左右子节点进行遍历.只有当节点的光能贡献误差小于一定阈值,且着色点位于节点包围盒之外时,才停止分离操作开始子节点的采样,如图 14 所示.其实从另一个角度来看,他们采取的分离操作等价于寻找一个光源割并对光源进行了聚类.不同的是,他们不是随机选择一个光源簇再选择一个光源,而是对每个光源簇都分别采样一个光源,计算光能贡献并相加.从这样的角度来看,该方法提出的遍历树采样方法是一种光源簇内采样光源的策略.

2019 年"Ray Tracing Gems"中[47]介绍了该算法的 GPU 实现,并比较分析了多种光源树构建方法.

2019 年 Liu 等人[48]基于 Estevez 等人的工作,提出在光源采样中考虑 BRDF 信息.他们用现有的面光源解析方法[49-50]估计 BRDF 在光源树节点对应包围盒的立体角下的积分,以此作为该光源树节点的 BRDF 重要性权重.他们的方法可以有效解决在同时包含多光源和光泽材质的场景中传统 BRDF 采样方法失效的问题.然而,由于在光源上进行点的采样时无法考虑 BRDF 信息,他们的方法仍然需要与 BRDF 采样方法进行结合.

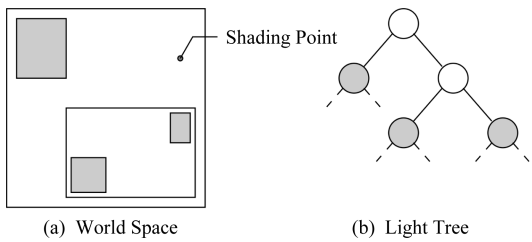


Fig. 14 Importance sampling with adaptive tree splitting[46]

图 14 带有自适应树分离的重要性采样[46]

基于光源聚类的重要性采样方法,近年来已经成为离线绘制中处理多光源场景的主流手段.本文之前介绍的加快可见性测试、光源聚类进行直接光照计算等工作,都可对光源聚类的重要性采样算法起到贡献作用.例如,加快可见性测试方面的工作可用于加快光源(簇)可见性项的重要性估计;光源聚类进行直接光照计算的相关工作可以用于构建更好的光源树和光源割,提高重要性采样的效率.

表 3 对本节介绍的光源采样方法进行了比较.可以发现,在采样时考虑可见性项的方法往往是通过

Table 3 Analysis of Lights Sampling Methods

表 3 光源采样方法比较

Year	Reference	Consider BRDF Term	Consider Visibility Term	Memory	Additional Features
2006	Ref [40]	√	√	Large	Precomputed Visibility via PRT
2009	Ref [41]	√	×	Little	Better BRDF Evaluation
2006	Ref [43]	×	√	Medium	On-line Learning PDF
2016	Ref [44]	×	×	Medium	Per-region Clustering
2017	Ref [38]	×	×	Large	Per-region PDF; No Clustering
2018	Ref [45]	×	√	Medium	On-line Learning PDF
2018	Ref [11]	×	×	Little	Better Light Tree; Tree Traversal Sampling
2019	Ref [48]	√	×	Little	Better BRDF Evaluation

预处理或者数据驱动地在线学习.而考虑 BRDF 项的方法仍然具有很大的改进空间.如何在采样过程中又快又好地同时考虑光源的可见性和着色点的 BRDF,仍然是一个值得探讨的话题.

5 实时绘制中的光源剔除(light culling)

实时绘制往往将光源的影响范围看作是有限的,这里将简要介绍通过光源剔除来加速绘制速率的方法,感兴趣的读者可以查阅 2015 年的 SIGGRAPH 课程[51]进行深入了解.

早先为了处理多光源场景,许多工作[52-54]提出使用延迟着色法(deferred shading)来替代正向着色法(forward shading),其分为 2 步操作:在几何处理阶段用 G 缓存(G-Buffer)存储可见片元的几何信息;在绘制阶段,遍历每个光源,对其影响范围内的每个可见片元进行绘制.由于每次绘制一个片元在一个光源下的光照时都需要读写 G-Buffer,这类方法会占用非常高的带宽.

为了解决高带宽占用的问题,一些工作[55-57]提出切片式延迟着色法(tiled deferred shading 或 deferred lighting),将原本绘制阶段的内外循环颠倒:对 G-Buffer 中的每个片元,查找可能受影响的光源,分别进行绘制.为了给片元分配受影响的光源,他们选择将屏幕空间用网格划分,每个网格共享相同的光源分配结果,从而摊销计算时间.光源分配的方法为:分别计算每个光源投影到屏幕空间的包围盒,对每个包围盒内所有网格区域记录对应的光源索引.

为了避免延迟着色法本身所具有的高存储要求,切片式的思想被运用到正向着色法,出现了切片式正向着色法(tiled forward shading, forward+)[58-59].

2012 年 Harada[60]提出在使用切片式着色法时对每个切片保守地估计最小最大深度,裁去影响区域为空白的光源.

2012 年 Olsson 等人[61]将切片式着色法扩展为聚类着色法(clustered shading).他们发现,原先的切片式方法是在屏幕空间进行光源分配,由于不同视角下屏幕空间的光源密度不同,绘制时间充满不确定性.对此,他们提出为原本屏幕空间的切片(tile)引入深度,将视锥体用网格划分(屏幕坐标方向均匀划分,深度方向对数划分),然后对每个非空的网格单元进行光源分配,如图 15 所示.之后,一些工作对具体的实现方法进行了探索和改进[62-63].

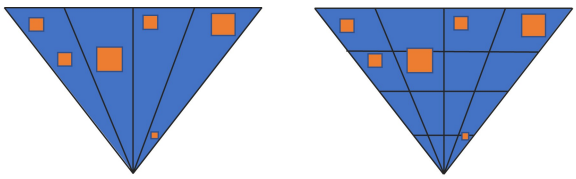


Fig. 15 View frustum in tiled shading and clustered shading[63]

图 15 切片式着色和聚类着色中的视锥体[63]

2017 年 O'Donnell 等人[64]提出通过沿深度方向建立一维的完全二叉树来存储每个屏幕切片或者视锥体网格块中的光源,并且给出了 GPU 友好的构造和遍历算法.该算法与聚类着色法结合,可以很好地弥补视锥体后方网格块深度范围过大、光源数量过多的问题.

上面的光源剔除算法只考虑有限的光源影响范围,这会造成图像结果的有偏性.2016 年 Tokuyoshi 等人[65]提出随机光源采样方法,使得绘制结果保持无偏.他们分别为每个光源设定一个固定的随机数,通过俄罗斯轮盘赌(Russian roulette)决定每个光源的影响范围,然后使用现有的光源剔除方法进行绘制.

表 4 给出了本节介绍的 5 种光源剔除方法的特点.可以看出,传统的延迟着色法会占用较多带宽.相比切片式着色,聚类式着色不太依赖视角,不同视角的效果相近.另外,所有的延迟着色类算法都需要花费较高的存储.

Table 4 Analysis of Lights Culling Methods

表 4 光源剔除方法比较

Method	Innermost Loop	Bandwidth Use	View Dependence	Memory
(Traditional) Deferred	Pixels	High	Low	High
Tiled Deferred	Lights	Low	High	High
Tiled Forward	Lights	Low	High	Low
Clustered Deferred	Lights	Low	Low	High
Clustered Forward	Lights	Low	Low	Low

6 总结与展望

本文对近年来离线绘制和实时绘制中的多光源绘制方法进行了综述.

在离线绘制方面,本文首先介绍了通过加速可见性测试以提高单个光源绘制效率的一类方法.这类方法通常可以运用在直接光照采样的重要性估计中;然后介绍了基于 VPLs 框架的光源聚类方法,可以实现计算时间相对光源数量的可伸缩性;最后介绍了使用光源聚类思想的重要性采样方法,这方面的工作往往受益于前 2 个领域的工作成果.如何在采样中高效地同时考虑光源的可见性和着色点的 BRDF 仍然是一个开放性的问题.

在实时绘制方面,本文介绍了光源剔除方面的工作.对于这一部分,如何结合 GPU 硬件更好地优化具体的实现算法,往往起到非常关键的作用.

针对多光源绘制问题,未来的研究重点可能是 3 个方向:

1) 如何建立更优的光源树.提出更好的建树启发准测,尝试自底向上建立光源树.

2) 如何制定更优的重要性采样策略,尽可能地包含 BRDF 项和可见性项.一方面继续优化光源簇的采样方法,另一方面也要考虑如何高效地在光源簇内部采样光源.

3) 如何利用空间连贯性,让相似的着色点共享重要性采样的概率分布,摊销计算量.

参 考 文 献

[1] Fascione L, Hanika J, Pieké R, et al. Path tracing in production [C] //Proc of ACM SIGGRAPH 2018 Courses. New York: ACM, 2018: No.15

[2] Christensen P, Fong J, Shade J, et al. RenderMan: An advanced path-tracing architecture for movie rendering [J]. ACM Transactions on Graphics, 2018, 37(3): No.30

[3] Kajiya J T. The rendering equation [C] //Proc of the 13th Annual Conf on Computer graphics and Interactive Techniques. New York: ACM, 1986: 143-150

[4] Pharr M, Jakob W, Humphreys G. Physically Based Rendering: From Theory to Implementation [M]. San Francisco, CA: Morgan Kaufmann, 2016

[5] Shirley P, Wang Changyaw, Zimmerman K. Monte Carlo techniques for direct lighting calculations [J]. ACM Transactions on Graphics, 1996, 15(1): 1-36

[6] Ward G J. Adaptive shadow testing for ray tracing [G] // Photorealistic Rendering in Computer Graphics. Berlin: Springer, 1994: 11-20

[7] Jensen H W, Christensen N J. Efficiently rendering shadows using the photon map [C/OL] //Proc of Compugraphics. 1995 [2019-11-07]. http://citeseerx.ist.psu.edu/viewdoc/download? doi=10.1.1.47.2582&rep=rep1&type=pdf

[8] Keller A, Wald I. Efficient importance sampling techniques for the photon map [C/OL] //Proc of Int Workshop of Vision, Modelling, and Visualization. 2000 [2019-11-07]. http://www.sci.utah.edu/~ wald/Publications/2000/ispm/ispm.pdf

[9] Hart D, Dutré P, Greenberg D P. Direct illumination with lazy visibility evaluation [C] //Proc of the 26th Annual Conf on Computer Graphics and Interactive Techniques. New York: ACM, 1999: 147-154

[10] Fernandez S, Bala K, Greenberg D P. Local illumination environments for direct lighting acceleration [C] //Proc of the 13th Eurographics Workshop on Rendering. Aire-la-Ville, Switzerland: Eurographics Association, 2002: 7-14

[11] Wald I, Benthin C, Slusallek P. Interactive global illumination in complex and highly occluded environments [C] //Proc of the 14th Eurographics Workshop on Rendering. Aire-la-Ville, Switzerland: Eurographics Association, 2003: 74-81

[12] Dong Zhao, Grosch T, Ritschel T, et al. Real-time indirect illumination with clustered visibility [C/OL] //Proc of Int Workshop of Vision, Modelling, and Visualization. 2009 [2019-11-08]. http://jankautz.com/publications/ClusteredVis_VMV09.pdf

[13] Popov S, Georgiev I, Slusallek P, et al. Adaptive quantization visibility caching [J]. Computer Graphics Forum, 2013, 32(2): 399-408

[14] Ulbrich J, Novák J, Rehfeld H, et al. Progressive visibility caching for fast indirect illumination [C/OL] //Proc of Int Workshop of Vision, Modelling, and Visualization. 2013 [2019-11-08]. https://cg.ivd.kit.edu/publications/p2013/PVCFID/PVCFID_ulbrich_2013.pdf

[15] Billen N, Engelen B, Lagae A, et al. Probabilistic visibility evaluation for direct illumination [J]. Computer Graphics Forum, 2013, 32(4): 39-47

[16] Billen N, Lagae A, Dutre P. Probabilistic visibility evaluation using geometry proxies [J]. Computer Graphics Forum, 2014, 33(4): 143-152

[17] Keller A. Instant radiosity [C] //Proc of the 24th Annual Conf on Computer Graphics and Interactive Techniques. New York: ACM, 1997: 49-56

[18] Dachsbacher C, Křivánek J, Hašan M, et al. Scalable realistic rendering with many-light methods [J]. Computer Graphics Forum, 2014, 33(1): 88-104

[19] Paquette E, Poulin P, Drettakis G. A light hierarchy for fast rendering of scenes with many lights [J]. Computer Graphics Forum, 1998, 17(3): 63-74

[20] Walter B, Fernandez S, Arbree A, et al. Lightcuts: A scalable approach to illumination [J]. ACM Transactions on Graphics, 2005, 24(3): 1098-1107

[21] Walter B, Bala K, Kulkarni M, et al. Fast agglomerative clustering for rendering [C] //Proc of 2008 IEEE Symp on Interactive Ray Tracing. Piscataway, NJ: IEEE, 2008: 81-86

[22] Gu Yan, He Yong, Fatahalian K, et al. Efficient BVH construction via approximate agglomerative clustering [C] //Proc of the 5th High-Performance Graphics Conf. New York: ACM, 2013: 81-88

[23] Davidovič T, Georgiev I, Slusallek P. Progressive lightcuts for GPU [C] //Proc of ACM SIGGRAPH 2012 Talks. New York: ACM, 2012: No.1

[24] Walter B, Arbree A, Bala K, et al. Multidimensional lightcuts [J]. ACM Transactions on graphics, 2006, 25(3): 1081-1088

[25] Wang Guangwei, Xie Guofu, Wang Wencheng. Enhancing illumination computation of lightcuts with spatial coherence [J]. Chinese Journal of Computers, 2013, 36(11): 2364-2370 (in Chinese)
(王光伟, 谢国富, 王文成. 基于空间聚类增强 Lightcuts 的光照计算[J]. 计算机学报, 2013, 36(11): 2364-2370)

[26] Bus N, Mustafa N H, Biri V. IlluminationCut [J]. Computer Graphics Forum, 2015, 34(2): 561-573

[27] Rehfeld H, Dachsbacher C. Lightcut interpolation [C] //Proc of High Performance Graphics. Aire-la-Ville, Switzerland: Eurographics Association, 2016: 99-108

[28] Bus N, Mustafa N H, Biri V. Global illumination using well-separated pair decomposition [J]. Computer Graphics Forum, 2015, 34(8): 88-103

[29] Callahan P B, Kosaraju S R. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields [J]. Journal of the ACM, 1995, 42(1): 67-90

[30] Maria M, Mustafa N, Bardoux T, et al. Visibility based WSPD for global illumination [C/OL] //Proc of the 13th Int Joint Conf on Computer Vision, Imaging and Computer Graphics Theory and Applications. 2018 [2019-11-01]. https://hal.archives-ouvertes.fr/hal-01698656

[31] Ou Jiawei, Pellacini F. LightSlice: Matrix slice sampling for the many-lights problem [C] //Proc of the 2011 SIGGRAPH Asia Conf. New York: ACM, 2011: No.179

[32] Hašan M, Pellacini F, Bala K. Matrix row-column sampling for the many-light problem [J]. ACM Transactions on Graphics, 2007, 26(3): No.26

[33] Huo Yuchi, Wang Rui, Jin Shihao, et al. A matrix sampling-and-recovery approach for many-lights rendering [J]. ACM Transactions on Graphics, 2015, 34(6): No.210

[34] Shen Yuan, Wen Zaiwen, Zhang Yin. Augmented Lagrangian alternating direction method for matrix separation based on low-rank factorization [J]. Optimization Methods and Software, 2014, 29(2): 239-263

[35] Novák J, Nowrouzezahrai D, Dachsbacher C, et al. Virtual ray lights for rendering scenes with participating media [J]. ACM Transactions on Graphics, 2012, 31(4): No.60

[36] Novák J, Nowrouzezahrai D, Dachsbacher C, et al. Progressive virtual beam lights [J]. Computer Graphics Forum, 2012, 31(4): 1407-1413

[37] Bartels P, Dutré P, Frederickx R. Adaptive lightslice for virtual ray lights [C] //Proc of Eurographics 2015. Aire-la-Ville, Switzerland: Eurographics Association, 2015: 61-64

[38] Huo Yuchi, Wang Rui, Hu Tianlei, et al. Adaptive matrix column sampling and completion for rendering participating media [J]. ACM Transactions on Graphics, 2016, 35(6): No.167

[39] Veach E. Robust Monte Carlo methods for light transport simulation [D]. Palo Alto, CA: Stanford University, 1997

[40] Akerlund O, Unger M, Wang Rui. Precomputed visibility cuts for interactive relighting with dynamic BRDFs [C] //Proc of the 15th Pacific Conf on Computer Graphics and Applications. Piscataway, NJ: IEEE, 2007: 161-170

[41] Wang Rui, Åkerlund O. Bidirectional importance sampling for unstructured direct illumination [J]. Computer Graphics Forum, 2009, 28(2): 269-278

[42] Wu Yuting, Chuang Yungyu. VisibilityCluster: Average directional visibility for many-light rendering [J]. IEEE Transactions on Visualization and Computer Graphics, 2013, 19(9): 1566-1578

[43] Donikian M, Walter B, Bala K, et al. Accurate direct illumination using iterative adaptive sampling [J]. IEEE Transactions on Visualization and Computer Graphics, 2006, 12(3): 353-364

[44] Vévoda P, Křivánek J. Adaptive direct illumination sampling [C] //Proc of SIGGRAPH ASIA 2016 Posters. New York: ACM, 2016: No.43

[45] Vévoda P, Kondapaneni I, Křivánek J. Bayesian online regression for adaptive direct illumination sampling [J]. ACM Transactions on Graphics, 2018, 37(4): No.125

[46] Estevez A C, Kulla C. Importance sampling of many lights with adaptive tree splitting [J]. Proceedings of the ACM on Computer Graphics and Interactive Techniques, 2018, 1(2): 25

[47] Moreau P, Clarberg P. Importance sampling of many lights on the GPU [G] //Ray Tracing Gems: High-Quality and Real-Time Rendering with DXR and Other APIs. New York: Apress, 2019: 255-283

[48] Liu Yifan, Xu Kun, Yan Lingqi. Adaptive BRDF-oriented multiple importance sampling of many lights [J]. Computer Graphics Forum, 2019, 38(4): 123-133

[49] Heitz E, Dupuy J, Hill S, et al. Real-time polygonal-light shading with linearly transformed cosines [J]. ACM Transactions on Graphics, 2016, 35(4): No.41

[50] Dupuy J, Heitz E, Belcour L. A spherical cap preserving parameterization for spherical distributions [J]. ACM Transactions on Graphics, 2017, 36(4): No.139

[51] Olsson O, Persson E, Billeter M. Real-time many-light management and shadows with clustered shading [C] //Proc of ACM SIGGRAPH 2015 Courses. New York: ACM, 2015: No.12

[52] Saito T, Takahashi T. Comprehensible rendering of 3-D shapes [C] //Proc of the 17th Annual Conf on Computer Graphics and Interactive Techniques. New York: ACM, 1990: 197-206

[53] Hargreaves S, Harris M. Deferred shading [C/OL] //Proc of Game Developers Conf. 2004 [2019-11-07]. https://my.eng.utah.edu/~cs5600/slides/Wk%209%20D3DTutorial_Deferred Shading.pdf

[54] Shishkovtsov O. Deferred shading in S.T.A.L.K.E.R.[G] // GPU Gems 2. Hoboken, NJ: InformIT, 2005: 143-166

[55] Andersson J. Parallel graphics in frostbite-current & future [C/OL] //Proc of SIGGRAPH 2009 Course. 2009 [2019-11-11]. http://s09.idav.ucdavis.edu/talks/04-JAndersson-ParallelFrostbite-Siggraph09.pdf

[56] Swoboda M. Deferred lighting and post processing on Playstation 3 [C/OL] //Proc of Game Developer Conf. 2009 [2019-11-07]. https://www.scribd.com/presentation/16104972/Deferred-Lighting-and-Post-Processing-on-PS3

[57] Lauritzen A. Deferred rendering for current and future rendering pipelines [C/OL] //Proc of SIGGRAPH 2010 Course. 2010 [2019-11-11]. https://software.intel.com/en-us/articles/deferred-rendering-for-current-and-future-rendering-pipelines

[58] Olsson O, Assarsson U. Tiled shading [J]. Journal of Graphics, GPU, and Game Tools, 2011, 15(4): 235-251

[59] Harada T, McKee J, Yang J C. Forward+: Bringing deferred lighting to the next level [C/OL] //Proc of Eurographics. 2012 [2019-11-11]. https://diglib.eg.org/handle/10.2312/conf.EG2012.short.005-008

[60] Harada T. A 2.5D culling for forward+ [C] //Proc of SIGGRAPH Asia 2012 Technical Briefs. New York: ACM, 2012: No.18

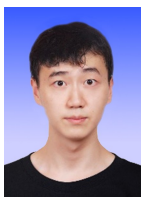
[61] Olsson O, Billeter M, Assarsson U. Clustered deferred and forward shading [C] //Proc of the 4th ACM SIGGRAPH/Eurographics Conf on High-Performance Graphics. Aire-la-Ville, Switzerland: Eurographics Association, 2012: 87-96

[62] Persson E. Practical clustered shading [C/OL] //Proc of SIGGRAPH 2013 Course. 2013 [2019-11-11]. http://www.humus.name/Articles/PracticalClusteredShading.pdf

[63] Drobot M. Improved culling for tiled and clustered rendering in call of duty: Infinite warfare [C/OL] //Proc of SIGGRAPH 2017 Course. 2017 [2019-11-11]. http://advances.realtimerendering.com/s2017/2017_Sig_Improved_Culling_final.pdf

[64] O'Donnell Y, Chajdas M G. Tiled light trees [C] //Proc of the 21st ACM SIGGRAPH Symp on Interactive 3D Graphics and Games. New York: ACM, 2017: No.1

[65] Tokuyoshi Y, Harada T. Stochastic light culling [J]. Journal of Computer Graphics Techniques, 2016, 5(1): 35-60



Liu Yifan, born in 1995. Master candidate in Tsinghua University. His main research interest is realistic rendering.

Xu Kun, born in 1985. PhD and associate professor in Tsinghua University. His main research interests include realistic rendering and image/video editing.