基于空间聚类增强 lightcuts 的光照计算*

王光伟[1),2)] 谢国富[1),2)] 王文成[1)]

[1)](中国科学院软件研究所计算机科学国家重点实验室 北京 100190)
[2)](中国科学院研究生院 北京 100049)

摘 要 Lightcuts 是面向多光源场景的高效绘制方法. 它渐进地对光源进行聚类组织, 并以二叉树进行管理, 由此在绘制时可用一些光源聚类的代表光源(统称为‘光源割’)进行光照计算, 以减少需要计算的光源数量. 但是, 当光源很多并分布复杂时, 在二叉树结构中寻找光源割也要花费不少的计算. 为此, 有些方法提出利用绘制中的图像连贯性来减少光源割的搜寻计算, 并取得了较好的效果. 本文提出利用空间聚类来减少光源割的搜寻计算, 即先根据位置和法向对场景中的几何位置进行聚类, 为各个类的代表点寻找其光源割, 以作为该类其它点搜索各自对应光源割的初始情况. 实验表明, 本文方法能很好地减少光源树中搜索光源割的计算, 提高 lightcuts 的光照计算效率, 优于利用图像连贯性的加速方法, 加速性能稳定, 并且光源越多分布越复杂, 加速效果越好.

关键词 光照计算;lightcuts;空间聚类
中图法分类号 TP301

Using Spatial Clusters to Improve Lightcuts for Illumination Computation

WANG Guang-Wei[1),2)] XIE Guo-Fu[1),2)] WANG Wen-Cheng[1)]

[1)](State Key Laboratory of Computer Science, Chinese Academy of Sciences, Beijing 100190)
[2)](Graduate University of Chinese Academy of Sciences, Beijing 100049)

Abstract Lightcuts is an efficient rendering algorithm for scenes with many complex lights. In order to approximate lights by representative lights of some clusters (defined as a cut), lightcuts builds a binary light tree to progressively cluster the lights. As a result, the number of lights to be processed is reduced during rendering. However, when dealing with plenty of complex lights, finding cuts in a light tree is still expensive. Therefore some algorithms exploit image coherence to accelerate cuts computation and achieve good performance. This paper proposes an algorithm that exploits spatial coherence to reduce the search cost of cuts. The proposed method clusters geometry positions by their material and normal. For each cluster, the proposed method first calculates the cut of the representative point of the cluster, and then uses the cut as the initial to search for the cuts of other points in this cluster. Experimental results show that the new method can significantly reduce the cost of finding cuts in light tree, enhance illumination computation and outperform other methods using image coherence, with stable effect on acceleration. In general, with more lights in a more complex distribution, the new method can achieve more acceleration.

*本课题得到国家自然科学基金(60773026,60873182,60833007)资助. **王光伟**, 男,1987 生,博士研究生,主要研究方向为实时真实感绘制等.E-mail: wanggw@ios.ac.cn.**谢国富**, 男,1985 生,博士研究生,主要研究方向为计算机图形学及可视化等.E-mail: guofu@ios.ac.cn.**王文成**, 男,1967 生,博士,研究员,博士生导师,主要研究领域为科学计算可视化、虚拟现实及计算机图形学等.E-mail: whn@ios.ac.cn.

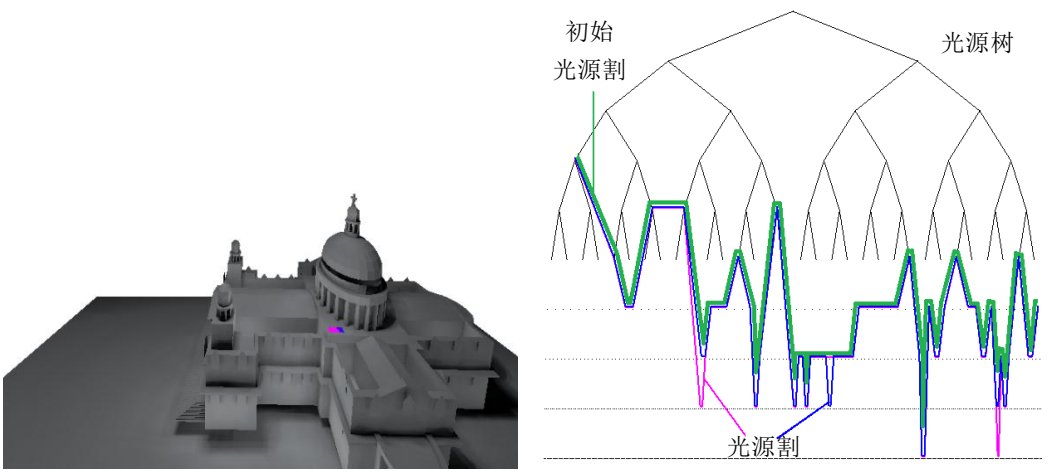
Keywords illumination computation; lightcuts; spatial coherence

1 引 言

复杂光源下的真实感绘制一直是计算机图形学中的重要研究内容.一般地,复杂光源被离散成一些点光源,然后使用点光源的累积计算来进行绘制[1],但这种累积计算往往很费时.为提高绘制效率,Walter 等提出对点光源进行聚类的处理方法[2],称为 lightcuts 方法,即:一个聚类中所有光源对某个几何位置的实际光能贡献,与使用该聚类的代表光源近似计算出的光能贡献相差不大时,就根据代表光源来计算该聚类对该位置的光能贡献.由此,可减少大量的光照计算.具体地,该方法是以二叉树的方式对光源进行层次化的组织,简称光源树(light tree),以便能自适应地选择合适的光源聚类,提高绘制效率.对于一个几何位置而言,其对应的各个光源聚类在光源树中就形成了一条分割界线,统称为‘光源割’(cut),如图 1 所示.绘制时,每个几何位置都需要从光源树的根结点开始搜索对应的光源割,因此,会在光源树中产生大量的搜索计算.

由于场景中邻近的成像点(成像面发出的绘制光线与三维场景的交点)一般在光源树中有相似的光源割,如图 1 中紫色及蓝色光源割所示,许多方法提出利用这种连贯性来进一步提高绘制效率.目前,这方面的工作主要是基于图像空间的连贯性来处理的,即:如果图像中某区域里邻近采样点的光照连贯性较强,则该区域中的许多像素点可以通过对邻近已进行了光照计算的像素点进行插值计算来获得其光照亮度[2];或者将这样的一个像素区域作为一个整体来搜索光源割,然后据此为其中的每个像素进行光照计算[3].但是,当场景比较复杂时,图像空间的连贯性会降低,因而会影响绘制效率的提高.

本文提出一种利用场景三维空间连贯性的加速方法,以增强 lightcuts 的光照计算效率,即:根据几何位置和法向,对场景空间中与绘制光线相交的各成像点进行在线的逐步聚类,并为每个类动态地更新初始光源割,以作为初始位置来为该类中各成像点搜索其相应的光源割.这样,在处理复杂的光照计算时,可节省很多在光源树中的搜索计算.如图 1 所示,对于紫色区域中的点,如果其搜索不是从光源树的根结点开始,而是从绿色的初始光源割开始,则可节省从根结点到绿色光源割的搜索开销.实验表明:与 lightcuts 算法相比,新算法搜索步数的节省效率可达 48.29%~73.57%,绘制计算的加速比可达 32.70%~49.15%;而与基于图像连贯性加速的类似方法[2-3]相比,新方法的加速比可达 6.38%~50.00%,并且光源越复杂,加速效果越好.在处理各种复杂的场景和光源情况时,新算法的加速性能比较稳定.



(a) 场景 (b) 光源树

图 1 场景几何中相近成像点的光源割以及其初始光源割.图(a)为寺庙场景,图(b)为针对该场景的光源树,其中紫色及蓝色的线分别对应左图中紫色区域及蓝色区域成像点相应的光源割;绿色线为该区域的初始光源割

本文将在第二节对相关工作进行简要介绍和讨论;在第三节介绍本文新算法及其实现;然后在第四节讨论新方法的实验结果,及其与类似工作的对比情况,并在第五节对新算法进行总结.

2 相关工作

多光源复杂场景的高效绘制一直是图形学中一个很具挑战性的问题.如果对各光源的光照结果进行累积运算,则绘制时间通常是随着光源的数目呈线性增长的.为了加快绘制速度,一些算法[2,4-5]采用重要性采样或聚类的方式来减少需要处理的光源数目,以提高绘制速度.其中,lightcuts 算法[2]渐进地对光源进行层次化聚类组织,可为每个成像点自适应地生成光源割,以快速高质量地进行绘制,是当前流行的多光源绘制加速技术.

在 lightcuts 方法的基础上,有许多工作进行了进一步的发展.其中有一些工作对 lightcuts 算法进行了扩展,使其能支持多种高级绘制效果,如景深、透明等[6-7].而另外的一些工作主要针对绘制效率的提高.这些加速方法可分为以下三类.

1)基于可见性预计算的方法.Akerlund 等首先提出在预计算光辐射传输技术(Precomputed Radiance Transfer)中使用 lightcuts 技术[8],可根据可见性预计算光源割,能以交互的速度绘制静态场景.文献[9]提出先分别根据可见性、几何材质和光源的位置来计算各自相关的光源割,然后再进行综合以获得各成像点对应的光源割,可以使用 GPU 进行并行计算以加速,并能得到高质量的光照结果.但这些方法需要存储大量预计算的可见性数据,不适合大规模场景的绘制.并且它们的绘制结果受限于预计算的准确性以及采样点的密度,难以进行自适应的动态处理.

2)增强光源割搜索质量的方法.文献[10]提出把 BRDF 重要性采样与 lightcuts 方法结合在一起,根据采样的重要性在光源聚类中创建多个代表光源,以提高光源割搜索的质量,并加快绘制速度.文献[11]提出使用物体截面图来提高成像点到光源结点之间可见性的计算效率,以改进光源割的计算质量.但这两种算法都没有利用场景中成像点之间关于光源割搜索的共享性,可以和本文算法相结合,以进一步提高绘制效率.

3)利用图像空间的连贯性的方法.Lightcuts 一文[2]中也提出了利用图像空间连贯性加速的算法:reconstruction lightcuts.该算法将成像面分成许多块,使得各个块内的像素具有很好的光照连贯性,于是,在各个像素块内可对大部分成像点使用简单的插值计算来获取它们的光照量.但该算法需要计算各像素块四个角上的光源割,并且需要为每个像素块建立一个局部光源树.当图像连贯性减少时,算法需要使用较密的块划分,会增加很多的计算,降低加速效果.并且,该算法使用插值来计算光源对成像点的亮度贡献,容易丢失细节.Coherent lightcuts 算法[3]提出对成像面进行逐步细分的划分,直至各个划分的子块中的像素相应的光源割具有很大的相似性,则以一个子块为一个整体在光源树中寻找其光源割,并根据该光源割为此子块中的各个像素进行光照计算.但该算法在成像面逐步细分的过程中要不断地在光源树中进行搜索计算,因此,当图像的连贯性不高时,该算法需要将成像面划分得很细,会降低加速效果.第四节的对比实验表明,当图像连贯性下降时,这些基于图像连贯性的方法的加速效率下降,甚至还会慢于原来的 lightcuts 方法;而本文的新算法加速性能稳定,能有效处理各种绘制情况.

3 算法思路

3.1 Lightcuts算法计算光源割的步骤

Lightcuts 算法在计算光源割时,自上而下逐步地遍历光源树中子节点.如果一个结点的代表光源对成像点的亮度贡献的误差上限超过所设定的误差阈值,就继续分别考察它的两个子结点,直至在所有搜索路径上找到了满足要求的结点,这些结点就形成了一个光源割.

根据韦伯定律[12],人可以感知到的亮度的最小变化大概是当前亮度的 1%,而文献[2]的实验中,使用亮度的 2%作为误差阈值就能得到较好的绘制结果,从而该算法使用如下公式计算成像点 x 的误差阈值 E_{thre}:

$$E_{thre}(x)=\sum_{C_k\in cut} L_{C_k}(x)*2\% \tag{1}$$

其中,cut 为成像点 x 对应的光源割;C_k 为光源割中的结点;$L_C(x)$表示结点 C 对成像点 x 的亮度贡献,该算法

使用 C 中代表光源 j 近似计算这一亮度贡献:

$$L_C(x)=\sum_{i\in C}M_i(x)G_i(x)V_i(x)I_i \approx M_j(x)G_j(x)V_j(x)\sum_{i\in C}I_i \tag{2}$$

其中,M 为材质项;G 为几何项;V 为可见性项;I 为光源亮度;$\sum_{i\in C}I_i$ 为结点 C 所覆盖的各光源的亮度累积,可预先计算以备重复使用.

使用公式(2)的近似计算会引入误差(即此近似亮度贡献和实际亮度贡献的差值).为此,lightcuts 算法提出以 M、G 和 V 在该结点处的最大可能变化范围来计算这一误差的上限.由于可见性的最大变化范围是 1,因此,该算法就将误差上限 $E_{C\text{-}upper}$ 的计算简化为:

$$E_{C\text{-}upper}(x)=M_{upper}(x)G_{upper}(x)\sum_{i\in C}I_i \tag{3}$$

其中,M_{upper} 为材质项的误差上限;G_{upper} 为几何项的误差上限.

3.2 基于空间聚类的lightcuts光照计算

在场景空间中,如果位置邻近的成像点的法向、材质和可见性信息较为相似,则它们相应的光源割也会比较类似.由此,它们在光源树中的许多搜索计算就可以共享来加速.为此,本文对场景中的成像点进行聚类,并为每个类中的代表成像点计算其光源割作为初始光源割,并动态地对其进行更新,然后该类中的其它成像点在搜索各自的光源割时,就以初始光源割作为起始位置,可减少在光源树中的大量搜索计算.这对于大规模复杂的光照计算是很有利的,因为此时的光源树一般会很大,计算光源割时的搜索也会比较多.

为减少对成像点聚类的计算,本算法先用均匀网格剖分场景,然后对同网格单元内的成像点进行聚类.对于一个成像点而言,它先与所在网格单元内已有的聚类的代表成像点进行比较;如果它与一个代表点在材质、几何方面比较相似,它就归入该类;否则,它就自成一个新类,并作为此类的代表成像点.当然,如果一个成像点是所在网格单元的第一个成像点,它就是一个类的代表成像点.在我们实现的聚类计算中,如果一个成像点与一个代表成像点具有相同的材质,并且它们的法向的夹角小于 10 度,我们就将它们归于同一个类.图 2 中显示了一个场景中的成像点的聚类情况.

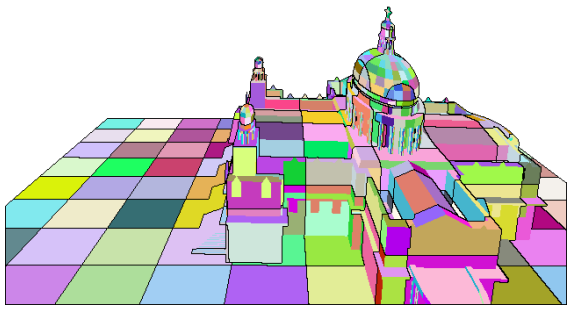


图 2　寺庙场景中成像点的聚类分布图.这里,用黑线分割各个网格,同一网格中的同色部分属于同一个类

在计算一个类的代表成像点的光源割时,我们用公式(1)计算该成像点的误差阈值,用公式(3)计算所考察结点的误差上限,当一个结点相关的误差上限小于误差阈值时,该结点就归入光源割.之后,该类中的其它成像点就将以该光源割作为初始光源割来寻找它们各自的光源割,具体的搜索计算方法还是采用 lightcuts 中的相应工作.此时,初始光源割中的结点,对于一个成像点而言,有 3 种情况.

1)正好合适,即:该结点的父结点对该成像点的光照计算有太大的误差,而该结点的光照计算却能满足要求.

2)欠分割,即:该结点对该成像点的光照计算有太大的误差,需要继续考察它的子光源结点.

3)过分割,即:该结点的父结点、甚至更高层的结点对该成像点的光照计算,能满足要求.此时,对于绘制质量来说没有影响,因为采用了更细致的光照计算;但计算的光源数量会增加,因而影响绘制速度.

显然,1)的情况是最理想的,2)的情况可按照原始的 lightcuts 算法继续搜索,也可节省一些搜索计算,而 3)的情况则需要尽可能地避免,因为光照计算一般是比较复杂的,不利于加速计算.对此,我们将在各成像点寻找它的光源割时逐渐地优化初始光源割,将其中的结点逐步地向父结点、甚至更高层的粗化结点进行移动,使得后续成像点在寻找它们的光源割时,能尽量减少‘过分割’现象的出现.我们所采用的措施如下:

1)根据可见性的变化来增大误差阈值,由此来调整初始光源割.根据代表点计算光源割时,我们保存搜索过的每一个光源结点的误差上限.只有当一个结点的误差上限大于误差阈值时,该结点的子结点才需要继续搜索.于是,我们可将误差阈值放大一些,就可将初始光源割的结点往粗化结点移动.为此,我们可根据可见性的变化来更新误差阈值.当初始光源割中的一个结点,对于代表点不可见而对于一个成像点可见时,我们就假设该结点对于代表点可见,并根据公式(1)重新计算误差阈值.

2)优化误差上限的计算.根据公式(3)计算一个光源结点对一个成像点的误差上限时,考虑了材质项的误差上限 M_{upper} 和几何项的误差上限 G_{upper}.由于同一个类的成像点具有相同的材质和类似的法线方向,所以 M_{upper} 可以保持不变.但 G_{upper} 主要决定于成像点到光源结点所覆盖区域的距离及相对位置关系,其值可变小以使得初始光源割向粗化的光源结点移动.为此,根据成像点所在网格单元的 8 个顶点计算它们各自相对于各个光源结点的 G_{upper},然后从中取出最小者作为优化误差上限计算的 G 项.基于以上分析,算法使用如下公式来计算光源结点 C 的误差上限 $E_{C\text{-}upper}$:

$$E_{C\text{-}upper}(P)=M_{upper}\left(x_p\right)\min_{x_i\in GV}\left(G_{upper}\left(x_i\right)\right)\sum_{i\in C}I_i \qquad (4)$$

其中,GV 为成像点所在网格的顶点的集合.

4 实验及讨论

本文所有实验均在一台 Dell T3400 微机上进行,该机装有一个双核 2.33GHz Intel Core E5650CPU,有 2G 内存.因为算法是单线程的,实验中只使用了一个核.我们使用 6 个不同复杂度的场景来验证新算法的性能,及与一些已有工作进行对比.这些场景及相关数据在图 3 及表 1 给出.我们在实验中取误差阈值为 2%,每个成像点的光源割中的结点个数上限为 1000.算法使用正方体三维网格划分场景,以场景包围盒最长边上网格数确定分辨率.通过许多实验,我们发现,当这一值为 40 时能得到较好的加速效果,之后的实验都使用这一值来确定网格分辨率.除特殊说明外,每个场景的图像分辨率为 640*480,每像素投射一根光线.

(a)寺庙 (b)会议室 (c)大厅

图 3 实验所使用的场景

表 1 实验所使用的场景属性

场景	面片数	光源数			
		直接光源	环境光源	间接光源	总计
寺庙	15k	4k	4k	15k	23k
会议室	514k	3k	0	4k	35k
大厅	3007k	16k	4k	495k	515k

4.1 加速效果

绘制图 3 中实验场景的统计数据在表 2 中列出.实验表明,新算法可很好地加速,加速率可达 32.70%~49.15%,并且平均每个像素的过分割结点一般不多于 3 个,相比于它们的光源割中的结点数目是很小

的,这说明,新方法计算出的光源割与 lightcuts 方法的实际光源割很接近,即新算法的初始光源割能很好地反映各像素的实际光源割的情况,节省关于光源割的搜索计算.

表 2　本文方法与原始方法绘制结果对比

场景	Lightcuts			新算法			加速性能		
	光源割大小#	搜索步数*	绘制时间(s)	光源割大小	搜索步数	绘制时间(s)	搜索步数的节省效率¯	过分割结点数¨	绘制加速比¥
寺庙	98.33	195.66	69.8	100.33	131.94	52.6	48.29%	2.00	32.70%
会议室	281.63	562.26	290.7	283.81	323.93	194.9	73.57%	2.18	49.15%
大厅	310.55	620.1	357.6	312.72	373.77	258.3	65.90%	2.17	38.44%

注:光源割大小#:平均每个像素对应的光源割中的光源结点数目.

搜索步数*:平均每个像素对应的在光源数中搜索的结点数目.

过分割结点数¨:平均每个像素的光源割中的过分割结点个数.

搜索步数节省效率¯:新方法相比于原始 lightcuts 方法在光源树中搜索步数的节省效率,以下面的公式计算:$(c-d)/d$,c 和 d 分别指 lightcuts 方法和新算法的搜索步数.

绘制加速比¥:新方法相比于原始 lightcuts 方法的加速率,以下面的公式计算:$(a-b)/b$,这里 a 和 b 分别指 lightcuts 方法和新方法的绘制时间.

光源的复杂程度对绘制效率的影响很大.为此,我们使用不同数目、不同类型的面光源,对大厅和会议室场景进行绘制测试,每个面光源都离散成一些点光源进行近似.本文使用光源数目与点光源方向的平均差异来度量光源的复杂程度.光源数量越多,点光源方向的平均差异越大,则光照计算越复杂.我们在实验中使用三种不同类型的面光源进行测试,它们的复杂度由低到高排序为:平面光源、柱面光源和球面光源.实验结果如图 4 所示:当光源数目增加时,算法加速效果呈波动上升的趋势(上图);在光源数目相同的情况下,使用较复杂的光源时,算法的加速效果较好(下图).综上可知,光源分布越复杂,新算法的加速效果就越好.这是因为,在复杂光源情况下,遮挡情况和法向分布情况,对光源割的搜索效率有很大的影响;而新算法对此进行了高效的处理.

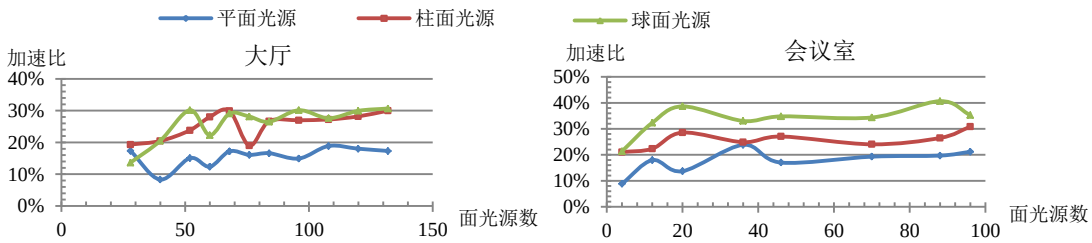


图 4　不同光源复杂度下,新算法的加速效果.横轴表示场景中面光源的个数,纵轴表示新算法相对于 lightcuts 算法的加速比.

4.2　和基于图像连贯性的加速算法的比较

除了与 lightcuts 算法进行比较外,我们还与利用图像空间连贯性进行加速的算法进行比较,包括 reconstruction lightcuts 算法[2]和 coherence lightcuts 算法[3].为了比较不同复杂度场景下三种算法对连贯性的利用程度,实验中绘制了由近及远时的一系列图像,如图 5 所示.

实验中用场景中所有可见的三角形面片的面积之和与它们的投影面积之和的比来度量图象的连贯性,该比值越大,则图象的连贯性越小,即单位面积内场景的复杂度越高.实验结果如图 6 所示,当场景在投影面上投影变小时,三种算法的搜索步骤的节省效率都会有所下降.但本文算法的加速性能比较稳定,下降幅度不大,而其它两种算法都有较大幅度的下降,甚至没有加速效果.与 coherence lightcuts 算法相比,虽然它的搜索步骤的节省效率比本文算法的高,但该算法只能节省搜索计算,而不能节省亮度贡献的计算,所以其加速效果不及新算法.而与 Reconstruction lightcuts 算法相比,它的搜索步骤的节省效率有时会高过新算法,但它需要为每个像素块创建一个光源树,耗时较多.而新算法仅需对代表成像点的光源割进行调整,需时较少,所以新算法的加速比相对

较高.由上可知,新算法的加速效果比较稳定,便于对场景进行各种远近的绘制.

图 5　实验所使用的场景,按镜头近及远排序

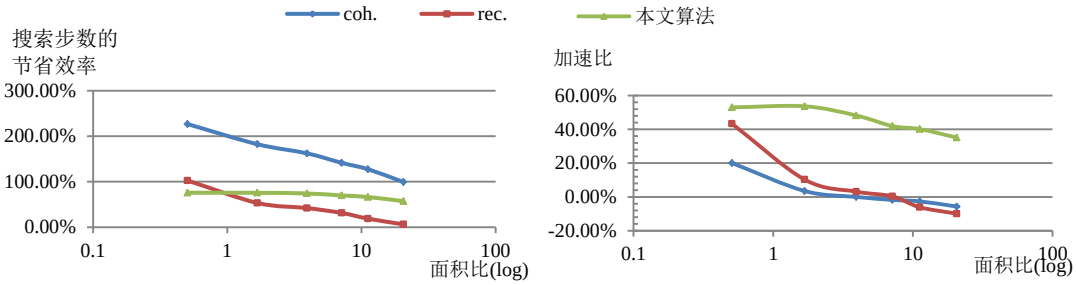


图 6　三种算法相对于 lightcuts 的加速效果的比较.面积比为场景中所有可见的三角形面片的面积之和与它们的投影面积之和的比

5　结语

本文基于空间聚类来利用场景的连贯性,能很好地减少 lightcuts 方法使用时的光源割的搜索计算,提高绘制效率,即:先为成像点的每一个类的代表点计算一个光源割,再以此作为该类中其它成像点搜索其光源割的起始位置,可共享利用很多光源结点的搜索计算.实验表明,本文提出的新算法可将绘制速度提高 32.70%~49.15%,并且光源情况越复杂,场景越大,则加速效果越好.与基于图象连贯性的方法相比,新算法的加速性能更好,并且针对不同远近的绘制的加速情况更稳定,一般不会出现减速的情况.总之,新算法提高了 lightcuts 方法的计算效率,特别适合处理大规模复杂的光照计算.

References:

[1] Keller A. Instant radiosity[C] //Proceedings of ACM SIGGRAPH 1997. New York: ACM Press, 1997. 49–56.

[2] Walter B, Fernandez S, Arbree A, Bala K, Donikian M, Greenberg D. Lightcuts: a scalable approach to illumination[J]. ACM Transactions on Graphics(S0730-0301), 2005, 24(3): 1098.

[3] Bodt T. Advanced global illumination using lightcuts[D]. Master thesis, KatholiekeUniversiteit Leuven, 2008.

[4] Hasan M, Pellacini F, Bala K. Matrix row-column sampling for the many-light problem[J]. ACM Transaction on Graphics(S0730-0301), 2007, 26(3):26-35.

[5] Wald I, Kollig T, Benthin C, Keller A, Slusallek P. Interactive global illumination using fast ray tracing[C] //Proceedings of the 13th Eurographics Workshop on Rendering. Switzerland: Eurographics Association Aire-la-Ville, 2002. 15–24.

[6] Walter B, Arbree A, Bala K, Greenberg D. Multidimensional lightcuts[J]. ACM Transaction Graphics(S0730-0301), 2006, 25(3): 1081.

[7] Arbree A, Walter B, Bala K. Single-pass scalable subsurface rendering with lightcuts[J]. Computer Graphics Forum(S1467-8659), 2008, 27(2): 507.

[8] Akerlund O, Unger M, Wang R. Precomputed visibility cuts for interactive relighting with dynamic BRDFs[C] //Proceedings of the 15th Pacific Conference on Computer Graphics and Applications. Washington, DC: IEEE Computer Society, 2007. 161-170.

[9] Cheslack-Postava E, Wang R, Akerlund O, Pellacini F. Fast, realistic lighting and material design using nonlinear cut approximation[J]. ACM Transactions on Graphics(S0730-0301), 2008, 27(5): 128.

[10] Wang R, Akerlund O. Bidirectional importance sampling for unstructured direct illumination[J]. Computer Graphics Forum(S1467-8659), 2009, 28(2): 269.

[11] Condon T, Walter B, Bala K, Freenberg D. Prefitered cross-section occluders[C] //Proceedings of the 2010 18th Pacific Conference on Computer Graphics and Applications. Washington, DC: IEEE Computer Society, 2010. 117-124.

[12] BLACKWELL H. Luminance difference thresholds[M]. In: Jameson D, Hurvich L M eds. Handbook of Sensory Physiology, vol. VII/4: Visual Psychophysics. New York: Springer-Verlag, 1972. 78–101.