

基于空盒自适应生成的动态场景光线跟踪计算

李 静¹⁾ 王文成¹⁾ 吴恩华^{1),2)}

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⁾(澳门大学科技学院计算机与信息科学系 澳门)

摘 要 提出了一项光线跟踪新技术,能有效提高光线在空白区域的行进速度.该技术首先用一种新方法创建均匀空间网格,然后用较少的空盒自适应聚集空的空间网格,以加快光线跟踪的计算.新加速结构的创建时间复杂度和空间复杂度均是 $O(n)$,而相应的光线跟踪计算的时间复杂度为 $O(\log n)$,与 kd 树结构相当.当该结构与已有的一些加速结构结合后,能很好地处理大规模动态场景.比如,光线逐根跟踪且计算二次衍生光线时,新技术可在普通 PC 机上高真实感地交互绘制包含 6G 三角面片的多 Buddha 动态场景.

关键词 加速;光线跟踪;空间网格;二次衍生光线;动态场景

中图法分类号 TP391

DOI号: 10.3724/SP.J.1016.2009.01172

Ray Tracing of Dynamic Scenes by Managing Empty Regions in Adaptive Boxes

LI Jing¹⁾ WANG Wen-Cheng¹⁾ WU En-Hua^{1),2)}

¹⁾(State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190)

²⁾(Department of Computer and Information Science, University of Macau, Macao)

Abstract This paper presents a new ray-tracing technique to efficiently speed up ray traversal in empty regions. It first produces grid cells in a new method, and then aggregates empty cells adaptively into fewer empty boxes, so that much acceleration can be obtained by reducing computation on ray going through empty regions. The new acceleration structure can be constructed in $O(n)$ time, and its storage complexity is also $O(n)$. By such a structure, rays can be traced in $O(\log n)$ time, similar to ray tracing via a kd-tree. When this structure is integrated with existing acceleration structures, large dynamic scenes can be efficiently rendered. For instance, a dynamic scene with multiple buddha models in up to 6G triangles can be interactively rendered in high photorealism on a common PC, where rays are traced individually and second rays are also computed.

Keywords acceleration; ray tracing; grid cell; secondary ray; dynamic scene

1 引 言

光线跟踪的加速计算一直是计算机图形学中的研究热点.尽管人们提出了诸如 kd 树和层次包围

盒等多种加速结构,但由于重建它们的代价昂贵,因此它们主要被用于静态场景而不适用于动态场景.最近提出的一些光线跟踪动态场景的技术能降低加速结构的重建开销以加速,但这些技术不能有效处理大规模场景,并且它们通常只跟踪从视点发出的

收稿日期:2007-10-09;最终修改稿收到日期:2008-01-12. 本课题得到国家“八六三”高技术研究发展计划项目基金(2006AA01Z306)、国家自然科学基金(60773026,60873182,60833007)和 Research Grant of University of Macau 基金资助. 李 静,女,1972 年生,博士,助理研究员,主要研究方向为计算机图形学、计算几何等. E-mail: lij@ios.ac.cn. 王文成,男,1967 年生,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、计算机图形学等. 吴恩华,男,1947 年生,博士,教授,博士生导师,主要研究领域为真实感绘制、虚拟现实、基于物理的动画等.

主光线和从光源发出的阴影探测光线,不考虑反射、折射等二次衍生光线.虽然一些方法借助于 GPU (Graphic Processing Unit)可以快速跟踪计算二次衍生光线,但由于 GPU 存储空间的限制而不能处理大规模场景. OpenRT 系统可以在 24 个双 CPU 计算机集群上实现大规模动态场景的交互光线跟踪绘制^[1].但在普通 PC 机上,以交互速度对大规模动态场景进行包含二次衍生光线的光线跟踪计算依然是个挑战.

在本文,我们提出了能在普通 PC 机上对大规模动态场景进行包含二次衍生光线交互绘制的方法.该方法可逐根跟踪光线以处理包含大量三角面片(如多达 6G)的动态场景.图 1 展示了一些绘制结果.在此,每帧重建场景加速结构. Kitchen 场景使用 6 个光源, Museum 场景使用 2 个光源,其它场景均为单光源.在配置为 Xeon 3.0GHz (2 CPU)、3.0G RAM 的普通 PC 机上,绘制上面 3 个场景(800×600 分辨率)和下面 3 个场景(1024×1024 分辨率)的效率如下:当绘制包括 shading、纹理、硬阴影、二次衍生光线时,分别达到 1.0, 1.3, 1.5, 2.5, (3.8_远+1.2_近)=2.5, (2.6_远+1.0_近)=1.8fps;当包括 shading、纹理而不包含阴影和二次衍生光线时,分别达到了 2.9, 2.6, 3.8, 4.0, (4.2_远+1.6_近)/2=2.9, (3.5_远+1.8_近)/2=2.65fps;当不包括 shading、纹理、硬阴影、二次衍生光线时,分别达到了 5.7, 3.1, 5.5, 5.4,

(6.0_远+2.0_近)/2=4.0, (4.5_远+2.0_近)/2=3.25fps. 该技术对光线跟踪计算有如下两个主要贡献:(1)通过生成高质量的空间网格结构、用空盒聚集空网格单元、将空盒与空间网格相结合等措施,大大加快了光线穿越空区域的速度并能迅速捕捉到运动物体的位置改变.这种结构被称为空盒网格混和结构 (Association of Empty Boxes and Grids, AEBG). (2)将 AEBG 与两种层次结构相结合以对大规模动态场景进行包含二次衍生光线的交互绘制.一种为文献[1-2]中所采用的用于绘制动态场景的双层结构,即为运动物体在其各自的局部坐标系下创建局部加速结构,同时为整个场景创建上层加速结构.另一种则是层次化的空间网格.为了既保持层次化空间网格组织的优越性,又降低更新空间网格所含内容的开销,我们将层次深度设为 2. 简而言之,新方法取得了如下进展:

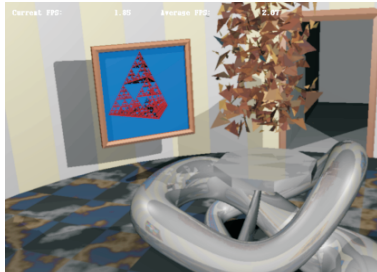
- (1)与现有技术相比,光线的行进速度更快,因为空盒很好地减少了光线在空区域内的行进步数.
- (2)可方便支持二次衍生光线的计算,因为新结构是针对单根光线加速计算的.
- (3)重建加速结构速度快,可快速绘制各种动态场景.
- (4)空间开销小,仅为 $O(n)$,可处理大规模场景.
- (5)易于与各种现有加速技术结合,如层次运动物体的局部管理^[1-2](详细讨论见第 4 节)和集束光线跟踪(packet-ray tracing)(结论明显,故本文略过).



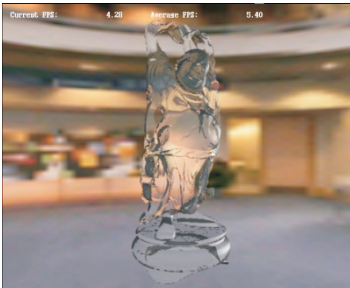
(a) Kitchen场景, 包含110K面片



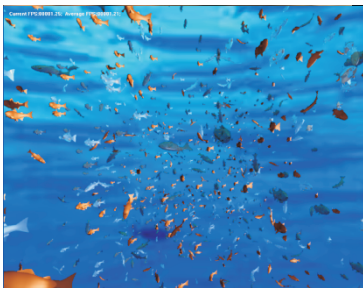
(b) Robot场景, 包含71K面片



(c) Museum场景, 包含11K面片



(d) Single Buddha场景, 包含6M面片



(e) Multiple Fish场景, 包含13.3M面片



(f) Multiple Buddha场景, 包含6G面片

图 1 用本文方法光线跟踪绘制的一些动态场景

本文将在第 2 节介绍相关工作,然后在第 3 节讨论新加速结构及相关加速技术;第 4 节介绍用新方法绘制大规模动态场景的加速技术;第 5 节给出实验结果,第 6 节进行总结并探讨进一步的工作。

2 相关工作

在光线跟踪中,光线穿越空区域会消耗大量的时间。为此,人们提出多种结构和技术来减少光线在空区域的计算。与这些方法相同,新方法也是基于一种数据结构来加快光线在空区域内的行进。下面,我们介绍相关的工作并与本文方法进行比较。

层次树(hierarchical trees). 包括八叉树^[3]和 kd 树^[1]在内的层次树结构被广泛用于光线跟踪的加速计算。它们是基于空间剖分的加速结构,即以某种方法剖分整个场景空间并将得到的空间单元组织起来。当光线在场景中行进时,也同时穿越空间单元。借助于空间单元之间的组织关系,光线跨越空单元占据的空白区域,找到包含物体的单元以进行光线-物体求交测试。因此,对这些结构来说减少光线所跨越的空白单元具有重要意义。层次树结构自适应地细分包含物体的空间,使空白区域被尽可能大的单元占据,可由此使光线在空白区域内的行进步数减少。一些研究表明,绘制静态场景时, kd 树优于其它结构^[4-6],因此最近的交互光线跟踪系统通常采用 kd 树进行加速^[1,7-8]。但是,层次树结构的创建开销很大。就我们所知,最快的创建方法的时间复杂度为 $O(n \log n)$ ^[9]。因此,层次树并不十分适合于绘制动态场景。相较而言,新方法能方便地处理动态场景,甚至比基于 kd 树的技术能更快地绘制静态场景,这将在 3.3 小节中讨论。

空间网格(grid). 空间网格结构通过将场景包围盒剖分为一些大小相等的长方体单元来进行加速^[10]。该结构简单、易于创建,并且光线可以通过增量加法依次快速访问空间网格单元^[11]。虽然该方法在处理几何元素分布均匀的场景时效率很高,但当几何元素分布不均匀时其性能将大幅下降。为此,人们提出了层次空间网格结构,即用较大的空间网格单元表示空白空间而将几何元素包裹在较小的空间网格单元内^[12-16]。但是,这些方法过高的空间需求使得它们不便于处理大规模场景^[4]。而新方法空间需求小,便于处理大规模场景。文献^[17]中,一种被称为大区域(macroregion)的六面体结构被用来表示多个相邻的空白空间网格单元,并由此使光线快速

穿过空白区域。该思想与我们的工作比较近似。但大区域是通过覆盖尽可能多的空网格单元得到的,因此多个大区域之间存在重叠,这样,为光线选择合适的大区域需要花费不少的计算。此外,生成大区域非常耗时且需要大量存储空间。而新方法中,空盒之间没有重叠,因此光线能够快速找到相应的空盒,并且,空盒可以被快速创建且空间开销小。

空间网格行进算法. 现已有多种用于加快光线在空间网格内行进的算法。如基于模板的行进算法^[18],就重复使用光线与连续空间网格单元相交的周期性模板,使得光线可以一次访问多个空间网格单元。但该算法仍然需要检测每个相交的网格单元是否为空。据我们所知,基于距离转换(distance transformation)的算法非常高效,这是因为每个空网格单元都记录了它到最近非空网格单元的最短距离,根据这些距离,光线可以以更大的步长一步跨越多个空网格单元^[19-23]。但是距离转换计算很昂贵,因此不适于处理动态场景。此外,位于空白区域边缘的网格单元记录的距离相对较短,这导致光线在到达这些网格单元时行进速度变慢。大体上,新方法有些类似于基于距离转换的算法,但它可以快速计算空盒,使光线一步跨越整个空盒,因此能方便地处理动态场景。

光线集束. 同时跟踪一束相邻光线能够显著降低每根光线的平均开销。该技术已被用于多种交互光线跟踪系统中,并与 kd 树^[1,7]、空间网格^[24]等多种数据结构结合,已取得了很好的效果。该技术非常适合于处理相关性强的光线,如主光线和阴影探测光线;但跟踪相关性差的二次衍生光线效率较低。新方法能够支持集束光线跟踪,但它主要是为了加速单根光线的跟踪计算,因此能很方便地跟踪二次衍生光线。第 5 节中的实验结果表明,新方法能够对非常大的动态场景进行包含二次衍生光线的交互绘制,而这是目前已有的集束光线跟踪技术难以达到的。

动态场景. 绘制动态场景的困难在于运动的几何元素可能会使先前创建的加速结构失效。最近的研究主要集中在降低重建开销上^[2,25]。OpenRT 系统可在计算机集群上交交互绘制大规模动态场景^[1]。在普通 PC 机上,中等规模动态场景可被交互绘制,小规模场景可被实时绘制^[26-27]。但是这些方法都局限于处理某种类型的动态场景。文献^[24]用相关网格行进算法跟踪集束光线,可处理各种动态场景;但它在场景规模和跟踪二次衍生光线方面具有局限

性.除了这些基于 CPU 的技术,还有一些基于 GPU 的加速技术^[28-29],可对动态场景进行包括二次衍生光线的交互绘制.但由于 GPU 存储空间的限制,这些方法不能处理大场景,并且它们还有其它的一些不足,如需要过多的预处理时间、光线与面片的求交是近似计算等.相较而言,新方法既便于处理大场景又能精确计算光线与面片的交点,能生成高质量的图像.

二次衍生光线.二次衍生光线能够大幅增强真实感效果,但遗憾的是跟踪二次衍生光线代价较高.因此现有的交互光线跟踪系统大多不计算二次衍生光线^[7,24].而那些支持二次衍生光线的方法又由于具有太多的局限性而难于实际应用.但是,新方法则可以很自然地支持二次衍生光线的计算.

3 用 AEBG 加速光线跟踪

在新方法中,最重要的任务是更新空间网格和快速计算空盒.在此,空间网格分辨率是一个决定性因素.为此,我们提出一种创建高质量空间网格的方法,以获得优化的空间网格分辨率.在此基础上,我们给出相应的空盒创建方法以及基于新结构的光线跟踪算法.

3.1 创建空间网格

空间网格创建算法已被人们广泛研究.创建时,如果空间网格单元过小,存储需求将增加,绘制效率则会降低;而如果空间网格单元过大,则单元内可能包含过多的几何元素,使得光线搜索相交的面片要花费更多的时间,同样会降低绘制效率.解决这一问题的简单做法是如文献^[4,12-13,15]中所讨论的,设定空间网格分辨率为 $r_x=r_y=r_z=\lfloor \sqrt[3]{dn}+0.5 \rfloor$,在此, r_x, r_y, r_z 分别为 x, y, z 方向上的空间网格单元个数, n 为场景中的面片数, d 为调整场景包围盒内空间网格单元密度的参数(通常 $d=1.0$).在该方法中,各坐标轴方向上的空间网格分辨率相等且产生的空间网格单元数量与 n 成线性关系.另一种方法是根据包围盒的长宽高分别确定 r_x, r_y, r_z 的值.这种创建方法能提供更多的加速,因为它考虑了面片的空间分布情况^[14].但这些方法都只能保证生成 $O(n)$ 数量的空间网格单元,而不能保证存储空间复杂度也为 $O(n)$,因为每个空间网格单元都需要存储指向其内部面片的指针,而一个大面片可能占据多个单元,从而需要存储大量指针.

在此,我们给出一个计算空间网格分辨率的新

算法,它综合考虑了面片分布、空间网格单元数量、面片占据的空间网格单元数和创建时间.该方法的创建时间为 $O(n)$,空间开销也为 $O(n)$,从而能确保在处理动态物体时能快速更新空间网格.具体步骤如下:

设 A, B, C 分别为场景包围盒的长、宽、高, a, b, c 分别为生成的空间网格单元的长、宽、高.再设每个面片的长、宽、高分别为 x_i, y_i, z_i ,在此 $i=1, 2, \dots$.显然,空间网格分辨率为 $r_x=A/a, r_y=B/b, r_z=C/c$.为了保证生成的空间网格单元数量为 $O(n)$,并且保证指向面片的指针数量也为 $O(n)$,下面两个不等式应得到满足,在此,不等式(1)的左侧为产生的空间网格单元数,不等式(2)的左侧为空间网格单元内存储的指向相关面片的指针总数.

$$\frac{A}{a} \cdot \frac{B}{b} \cdot \frac{C}{c} \leq O(n) \quad (1)$$

$$\sum_{i=1}^n \left(\frac{x_i}{a} + 1 \right) \left(\frac{y_i}{b} + 1 \right) \left(\frac{z_i}{c} + 1 \right) \leq O(n) \quad (2)$$

通过分析不等式(2)的左侧可知,如果满足下列7个不等式,则不等式(2)一定被满足,由此可得到 a, b, c 的值.

$$\frac{1}{a} \sum_{i=1}^n x_i \leq O(n) \quad (3)$$

$$\frac{1}{b} \sum_{i=1}^n y_i \leq O(n) \quad (4)$$

$$\frac{1}{c} \sum_{i=1}^n z_i \leq O(n) \quad (5)$$

$$\frac{1}{ab} \sum_{i=1}^n x_i y_i \leq O(n) \quad (6)$$

$$\frac{1}{ac} \sum_{i=1}^n x_i z_i \leq O(n) \quad (7)$$

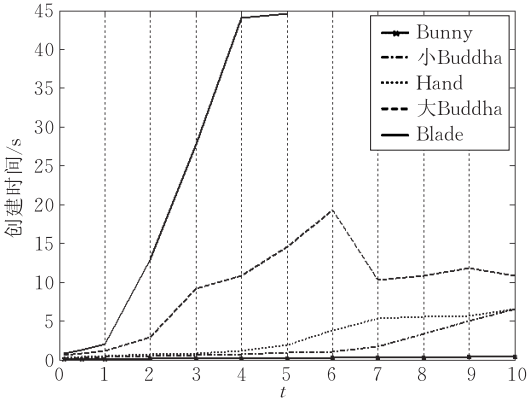
$$\frac{1}{bc} \sum_{i=1}^n y_i z_i \leq O(n) \quad (8)$$

$$\frac{1}{abc} \sum_{i=1}^n x_i y_i z_i \leq O(n) \quad (9)$$

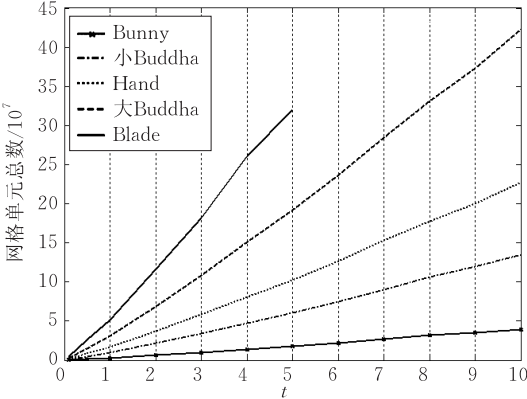
为了计算方便,我们设置参数 t ,并令 $a = \frac{1}{t} \cdot \frac{1}{n} \sum_{i=1}^n x_i, b = \frac{1}{t} \cdot \frac{1}{n} \sum_{i=1}^n y_i, c = \frac{1}{t} \cdot \frac{1}{n} \sum_{i=1}^n z_i$,因此上述不等式中的 $O(n)$ 可以被表示为 $t \times n$.设 t 的初始值为 1.0,然后用 Newton-Rapason 方法迭代调整 t 的值,直到上述不等式均被满足.尽管该方法有些复杂,但它很有效.图 2 展示了该算法开销随 t 值变化的情况.实验的被测模型从斯坦福大学的模型网站下载,它们分别为 Bunny(91K),小 Buddha(293K),Hand(654K),大 Buddha(1087K)和 Blade(1765K)

(括号中为模型包含的三角面片数). 为了尽量排除程序实现对实验的影响,除空间网格创建时间外,算法开销用 4 个与计算机硬件配置无关的参数来衡量,它们是:空间网格单元总数、空间网格包含的总指针数、每根光线穿越的平均空间网格单元数和每根光线与三角形面片的相交的平均次数. 一般来说, t 值越大,空间网格单元越密,所需的创建时间和占用空间越多,并且光线穿越的单元数也越多. 在图 2

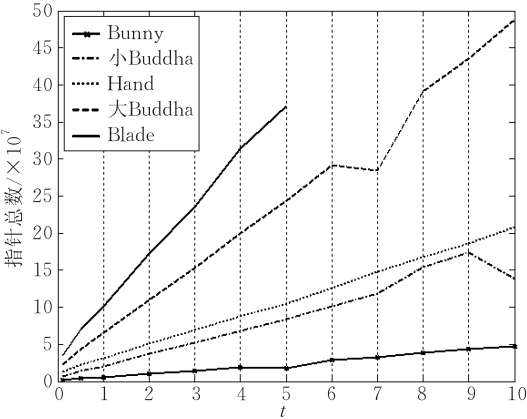
(d)中,曲线并未随 t 值的增大而上升,这是由于我们的新加速结构(详细讨论见下一小节)能有效抑制遍历的单元数. 从统计数据可知,当 t 超过 2.0 时,存储开销上升但访问的面片数并未显著下降. 相反,当 t 小于 0.5 时,被访问的面片数则过多. 因此,我们一般可选择 1.0 作为 t 的优化初始值,并将该值用于下面的实验中.



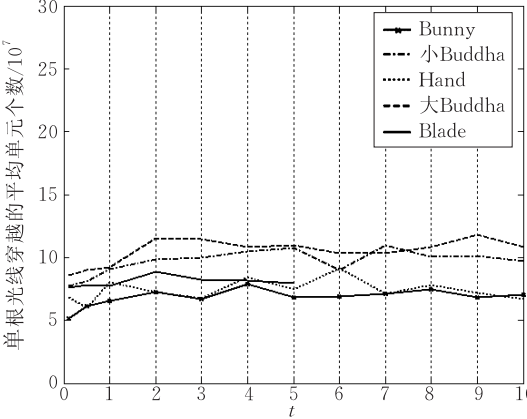
(a) 创建时间



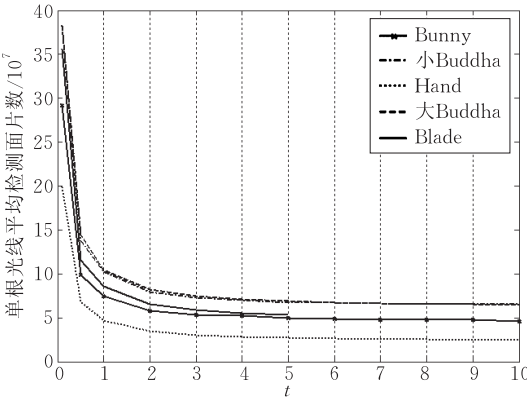
(b) 网格单元总数



(c) 指针总数



(d) 单根光线穿越的平均单元个数



(e) 单根光线平均检测面片数随 t 值变化的曲线

图 2 空间网格创建和光线跟踪开销随 t 值变化的数据曲线

3.2 创建空盒

空盒是根据不包含任何几何元素的空网格单元

创建的. 每个空单元都存有指向覆盖它的空盒的指针. 由于我们用空盒加快光线在空白区域内的行进,

因此其数量越少越好. 此外, 由于在处理动态场景时空间网格会被频繁更新, 因此空盒应该能够迅速地在线创建. 这里, 我们采用一种简单创建方法. 尽管它不能保证产生最少的空盒, 但它确实能产生足够的空盒, 以高效绘制动态场景. 具体步骤是: 首先任意选取一个空单元作为种子, 然后通过迭代加入相邻的且未被其它空盒覆盖的空单元来逐步扩张空盒. 在扩张过程中, 对应空盒 6 个面的 6 个方向被依次反复处理以免产生狭长的空盒. 重复上述过程直到所有的空网格单元均被空盒覆盖为止. 图 3 用一个 2 维例子说明了空盒的生成过程.

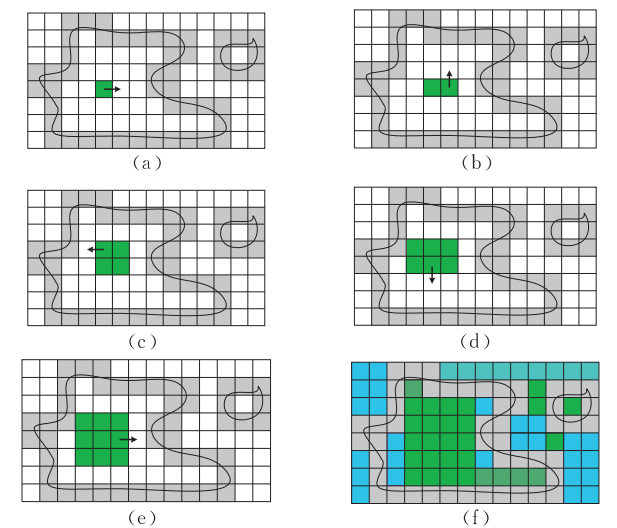


图 3 创建空盒的步骤, 白色和浅灰色空间网格单元分别表示空网格单元和含有几何元素的网格单元((a)中, 深灰色单元为被选中作为种子的网格单元. 在(b)、(c)、(d)和(e)中, 空盒被依次沿不同方向扩张, 在(f)中最终完成空盒的创建. (f)中, 不同的空盒以不同的颜色表示, 它们覆盖了所有的空网格单元)

在生成空盒过程中, 因为空网格单元有 6 个面, 所以每个空网格单元至多被访问 6 次. 又因为空间网格单元数量为 $O(n)$, 所以场景中的所有空盒可以在 $O(n)$ 时间内被创建. 此外, 由于空盒之间没有重叠, 空盒个数一定小于网格单元数, 因此空盒的空间复杂度也为 $O(n)$.

3.3 快速光线跟踪

基于前两小节讨论的 AEBG 结构, 我们可以同时利用空盒与空间网格的优点来加速光线跟踪: 使用空盒可以减少光线在空白区域的行进步数; 使用空间网格则可以以简单的计算跟踪光线的行进. 当光线到达与某空盒相关联的空单元时, 一步跨越整个空盒覆盖的空白区域, 然后到达与空盒射出点对应的新空间网格单元. 这样, 光线在空区域内的行进步数将大为减少而加速跟踪计算. 图 4 给

出了一个光线在 AEBG 结构中穿越空区域的二维例子.

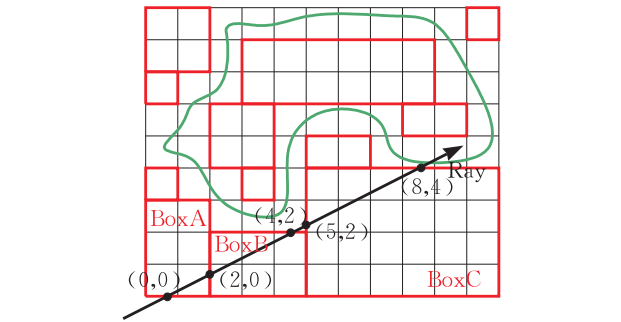


图 4 曲线部分表示场景中的几何元素, 粗实线矩形则表示空盒. 当光线进入包围盒后, 首先到达单元 $(0, 0)$, 在此得到空盒 BoxA. 根据光线离开 BoxA 的点, 计算出光线到达的新单元 $(2, 0)$, 并由此得到空盒 BoxB. 在离开 BoxB 后, 光线到达单元 $(4, 2)$, 没有找到相交面片, 因此继续行进, 到达单元 $(5, 2)$, 得到空盒 BoxC. 在离开 BoxC 后光线到达单元 $(8, 4)$, 并在这里找到相交面片. 图中的黑色圆点代表这根光线穿越空白区域时所经历的步幅.

为检测新结构的加速性能, 我们将其与另两种常用的加速结构进行了对比实验, 这两种结构是: 无空盒的均匀空间网格和 kd 树. 比较的性能参数分别为创建时间、占用空间(用空间网格单元总数和总指针数表示)、单根光线访问的平均空间网格单元数、与单根光线进行求交测试的平均三角面片数. 我们对 20 个模型进行了测试, 它们是关于 Stanford 的 Buddha 和 Hand 模型的不同简化版本, 其统计数据见表 1.

表 1 被测模型的统计数据

模型编号	Hand 三角面片数	模型编号	Buddha 三角面片数
1	804	11	1672
2	2080	12	3408
3	4420	13	7260
4	9508	14	15264
5	19623	15	31092
6	40936	16	64808
7	80180	17	128216
8	163776	18	263108
9	321652	19	516404
10	656528	20	1056004

在我们的实验中, 空间网格分辨率按照第 3.1 小节给出的方法计算, 且创建 kd 树时选择空间单元的中分面作为分割面. 实验结果在图 5 中给出. 在图(a), (b)和(c)中, 新结构的曲线与纯空间网格法极相近; 而在图(d)和图(e)中, 新结构的曲线与 kd 树的曲线形状极接近. 因此可知, 在创建时间和所需空间方面, AEBG 结构与纯空间网格法相近, 尽管要

稍微多一些;而在光线跟踪时间上,AEBG 与 kd 树相近,均为 $O(\log n)$,并且要比 kd 树稍快. 因此可以

说,我们的方法能更好地加速动态场景光线跟踪计算.

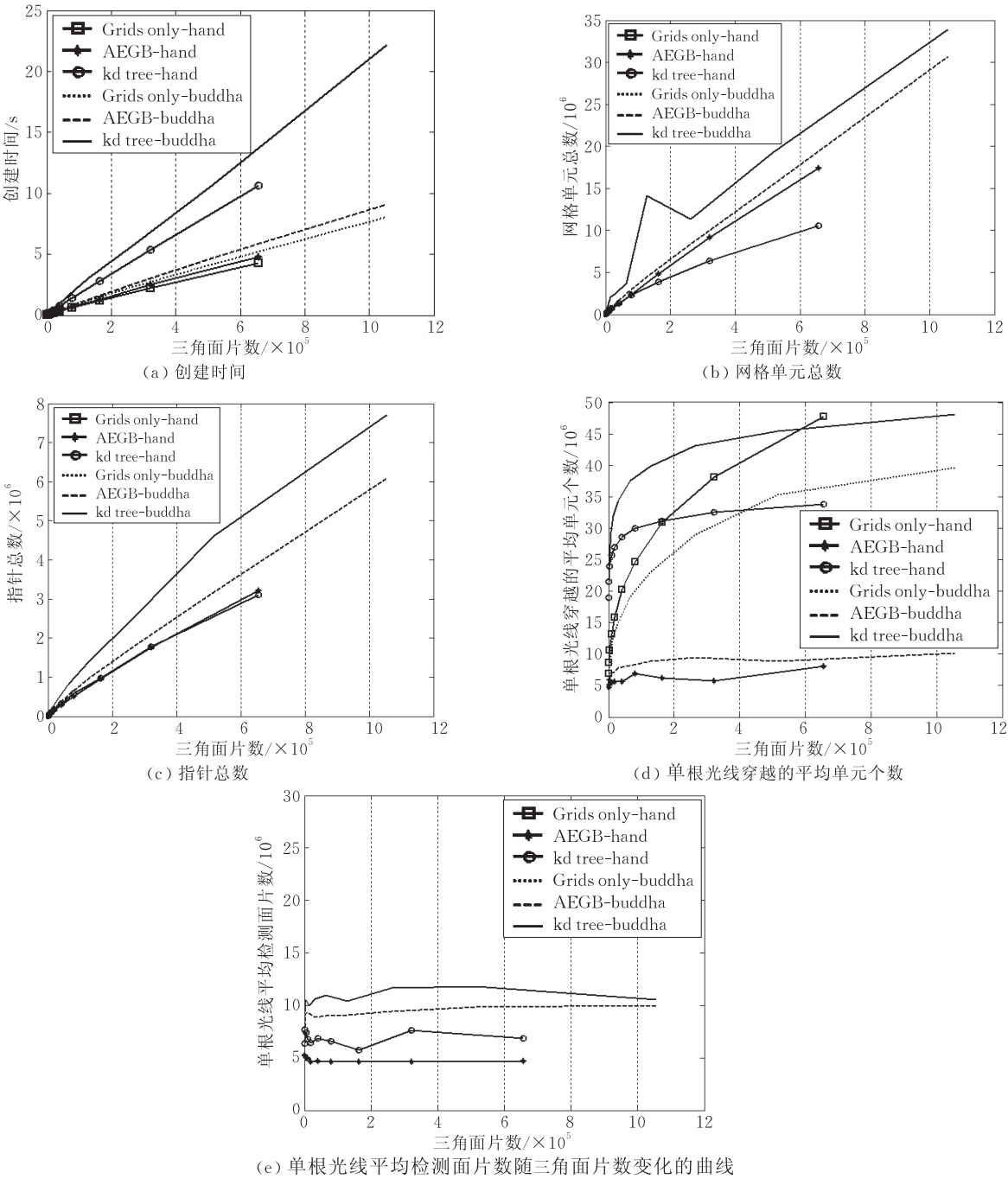


图 5 用 3 种加速结构,即无空盒空间网格、AEBG 和 kd 树,对两个系列模型进行性能测试的结果(图中显示 AEBG 与空间网格结构具有相似的创建时间和空间需求,而与 kd 树在光线跟踪计算上有相仿的性能)

4 绘制大规模动态场景

处理大规模动态场景时,除了利用加速结构大幅提高光线行进速度外,快速重建加速结构也是十分重要的.为此,我们将 AEBG 与其它两种层次结

构相结合,以综合利用它们的优点.一种是层次空间网格,另一种是在文献[1-2]中使用的针对动态场景的双层结构.对于层次空间网格,我们仅创建两层,并分别为每层空间网格创建 AEBG 结构.这样,不但空间需求不会增加太多,而且可利用层次空间网格的优点.对于双层结构,需要为每个具有相同仿射

变换的几何元素集合在其局部坐标系中创建一个局部加速结构,同时还需要为整个场景创建一个上层加速结构.这里,我们将每个加速结构都构造两层的层次空间网格.由此,就形成了针对大规模动态场景绘制的混合加速结构,其创建算法伪代码如图 6 所示.

```

CreateHHG(object, facets)
/* object 指具有相同仿射变换的面片集合, facets 指不包含在
   任何 object 内的面片 */
{
    根据 object 数和 facet 数计算场景空间网格的分辨率;
    生成空间网格;
    根据空间网格创建空盒;
    for(每个所含 object 或 facet 数量超过给定阈值的空间网格
        单元)
    {
        视其为包围盒并创建自己的细分空间网格,空间网格分辨率
        根据该单元内包含的 object 和 facet 数计算;
        根据细分空间网格创建自己的空盒;
    }
    for(每个 object)
    {
        建立该 object 的局部坐标系;
        在局部坐标系下为该 object 创建局部空间网格,空间网格
        分辨率根据 object 中包含的 facet 数计算;
        创建局部空盒;
        for(每个所含 facet 数超过给定阈值的单元)
        {
            视其为局部包围盒并创建自己的细分空间网格,空间网
            格分辨率根据该单元内包含的 facet 数计算;
            根据细分空间网格创建自己的空盒;
        }
    }
}

```

图 6 混合结构的创建算法

不失一般性,图 7 展示了一个混合结构的二维示例.在该例中, W_2 、 W_3 和 W_4 具有相同的仿射变换,在创建整个场景的空间网格结构时被看作为一个物体(称为 W_5).整个场景的空间网格用虚线表示,其相关空盒用实线矩形表示,如图 7(a)所示.由于与 W_1 相关的某些空间网格单元包含了过多的面片,因此它们被再次细分.对于物体 W_5 ,我们为其创建包围盒.在 W_5 的局部坐标系中,以相同的方法为其创建空间网格与空盒结构,如图 7(b)所示.

为便于描述,我们将图 7(a)中所示的顶层场景加速结构称为 HHGS(Hybrid Hierarchical Grid for Scenes),而将图 7(b)中所示的局部加速结构称为 HHGO(Hybrid Hierarchical Grids of Objects).例如,在一个包含 1000 个 Buddha 模型、每个 Buddha 包含 6M 面片的场景中,可为 Buddha 模型建立一个 HHGO,而为整个场景建立 HHGS.这样

HHGS 中将每个 Buddha 看作一个物体,故只需处理 1000 个,而非 $1000 \times 6M$ 个面片.

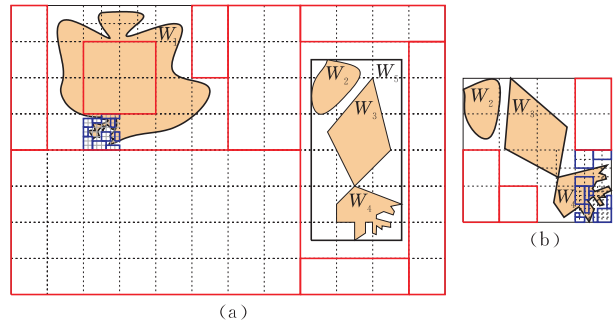


图 7 (图(a)为用于光线跟踪大规模动态场景的混合结构的一个二维示例,其中实线矩形和表示层次结构中位于不同层次的空盒.由于 W_2 、 W_3 和 W_4 具有相同的仿射变换,因此它们被视为一个整体,并为其构造局部混合结构,如图(b)所示)

这种混合结构对于处理层次运动类型的动态场景非常有效.对于无结构运动类型的场景,混合结构中不考虑针对物体的局部加速结构.第 5 节的实验表明这种混合结构对于各种类型的运动场景均十分有效,并且可以在普通 PC 机上交互绘制大规模动态场景.

4.1 用混合结构加速光线跟踪

基于混合结构,可以方便地处理大规模场景,其计算方式如下:当一条光线穿越混合结构时,首先计算它射入场景包围盒时 HHGS 中第一个与之相交的空间网格单元.然后根据该单元的属性采取相应的措施进入下一网格单元.以这种方式依次访问空间网格单元,直到光线与一个面片相交或者射出场景包围盒为止.根据网格属性采取的措施如下:

(1)若空间网格单元为空单元,则利用对应的空盒跨越一段空白区域.

(2)若空间网格单元包含面片,则检测光线是否与这些面片有交.如有,则找到首个与光线相交的面片;否则光线进入下一空间网格单元.

(3)若空间网格单元包含更细的空间网格,则取得该细分空间网格,重复上面的操作直到在细分空间网格中找到相交面片或者光线射出该细分空间网格.

(4)若空间网格单元包含 HHGO,则依次处理这些 HHGO.如果在这些 HHGO 中可以找到与光线相交的面片,则比较这些面片,从中找出最近的面片.在处理 HHGO 时,光线被转换到物体局部坐标系中,并用与上面相同的方法在 HHGO 中行进并

寻找相交面片,直到找到相交面片或射出 HHGO 为止.

对于动态场景,只需在每帧绘制前对混合结构进行更新,就可采取上面的方式进行光线跟踪计算.

5 实验结果

我们在一台配置有一个 3.0GHz 双核 Xeon CPU、3.0GB 内存和 Windows XP 操作系统的 PC 机上进行了实验. 由于未使用 GPU,因此大模型可以被方便地处理. 在测试中,我们采用 Phong 模型进行光照计算,所有的光线均是单独跟踪计算,且未对模型进行层次细节 (level-of-detail) 的组织 and 运用.

为了检测新方法对多种动态场景的处理效率,我们对动画光线跟踪的 7 个基准测试场景进行了测试^[30] (由 <http://www.ce.chalmers.se/research/group/graphics/BART> 获得). 此外,我们还用 “instance” 技术通过拷贝模型生成了一些大规模场景.

在测试这 7 个场景时,采用文献[30]给出的光源与纹理配置,且绘制图像的分辨率均为 800 × 600. 场景与绘制效率的统计数据在表 2 中列出.

表 2 用文献[30]中的场景测试新加速技术的统计数据

场景	Object 数	Facet 数/k	效果 1 /fps	效果 2 /fps	效果 3 /fps
Kitchen	393	110	1.0	2.9	5.7
Robot	1848	71	1.3	2.6	3.1
Museum3	208	10	2.0	5.0	8.0
Museum4	400	10	1.8	4.5	6.7
Museum5	1168	11	1.5	3.8	5.5
Museum6	4240	14	1.2	3.0	3.6
Museum7	6520	26	0.7	1.5	1.7

注: 效果 1: 绘制包括 shading、纹理、硬阴影和二次衍生光线的绘制速度; 效果 2: 绘制包括 shading、纹理、不包括阴影和二次衍生光线的绘制速度; 效果 3: 绘制不包括 shading、纹理、阴影和二次衍生光线的绘制速度.

在测试我们自己方法生成的大规模场景时,我们绘制了不同大小、不同远近的图像 (远景、近景). 相关的统计数据在表 3 和表 4 列出.

表 3 大规模动态场景的统计数据

场景	Object 数	Facet 数	模型种类
Single Buddha	1	6.0M	1
Fish1	1200	13.3M	2
Fish2	12000	133M	1
Multiple Buddha	1000	6.0G	1

表 4 绘制大规模动态场景的统计数据

场景	远/近	图像大小	效果 1 /fps	效果 2 /fps	效果 3 /fps
Single Buddha		VCD	28.0	31.2	39.5
		512 ²	12.0	14.5	20.8
		DVD	5.2	8.4	10.9
		1024 ²	2.5	4.0	5.4
Fish1	远	VCD	27.8	30.5	37.8
		512 ²	13.4	14.6	19.5
		DVD	7.5	8.6	11.2
		1024 ²	3.75	4.2	6.0
	近	VCD	8.7	10.5	13.4
		512 ²	4.1	6.8	8.7
		DVD	2.5	3.2	3.9
		1024 ²	1.2	1.6	2.0
Fish2	远	VCD	2.2	2.2	2.2
		512 ²	2.0	2.0	2.0
		DVD	1.8	1.8	1.8
		1024 ²	1.4	1.4	1.6
	近	VCD	1.8	1.8	1.9
		512 ²	1.4	1.4	1.6
		DVD	1.1	1.1	1.1
		1024 ²	0.7	0.7	0.9
Multiple Buddha	远	VCD	21.1	25.6	30.0
		512 ²	9.8	12.3	15.1
		DVD	5.2	7.0	8.5
		1024 ²	2.6	3.5	4.5
	近	VCD	8.3	13.3	13.9
		512 ²	3.5	5.9	6.2
		DVD	2.3	4.0	4.3
		1024 ²	1.0	1.8	2.0

注: VCD 和 DVD 指用于 VCD 和 DVD 图像分辨率, 分别为 352 × 288 和 720 × 576.

从表 2~4 的数据可知, 新技术可以交互地绘制大规模动态场景. 当运动较复杂时, 特别是有无结构运动时, 绘制速度不高, 如场景 “Museum7”. 其原因是在这种情况下运动计算会占用较多的时间. 新加速结构的创建速度很快, 如处理 “Single Buddha” 场景时, 新方法比文献[30]的绘制速度快很多, 尽管 “Single Buddha” 更大些. 根据表 4 给出的数据, 图像分辨率对新方法的影响比场景大小对其的影响更大, 这是因为图像大小决定了光线的数量. 由于光线跟踪技术天然地排除了不可见几何体, 因此除了需要计算运动物体位置的改变, 花费在不可见物体上的计算很少, 这有利于绘制的加速. 总的来说, 新方法大大减少了重建动态场景加速结构的开销, 以及光线穿越空区域的开销. 因此, 新技术能够在计算二次衍生光线的情况下交互绘制大规模动态场景.

本文附带的 3 段视频展示了对动态场景的绘制结果, 这里每帧图像的绘制都计算了二次衍生光线 (下载地址分别为 <http://lcs.ios.ac.cn/~lij/download/bart.wmv>, <http://lcs.ios.ac.cn/~lij/>

download/largescene_dvd.wmv, http://lcs.ios.ac.cn/~lij/download/realtime_vcd.wmv). 从这些视频可见,新方法可对复杂的或大规模的动态场景进行高效的绘制,使得物体运动的感知很流畅。

6 结论和未来工作

本文提出了一种加速光线跟踪计算的新技术。通过创建高质量的空间网格、用空盒聚集空网格单元、并将空间网格与空盒相结合,该技术能显著提高光线跟踪计算的速度。新结构的创建时间复杂度和空间复杂度与空间网格法相同,均为 $O(n)$,而相应的光线跟踪计算复杂度则与 kd 树相当,为 $O(\log n)$,并且实验结果显示它比 kd 树算法的效率更高。通过将新结构与现有的层次结构相结合,即便是包含二次衍生光线计算,我们也可在普通 PC 机上对大规模动态场景进行交互的高真实感绘制。例如,对于包含 1000 个 Buddha 模型、总共 6G 个三角面片的动态场景,我们可以在跟踪二次衍生光线的情况下对它进行每秒若干帧的绘制。就我们所知,即使是对如此大规模的静态场景进行交互绘制也是很困难的。

未来的工作可以在多方面展开,如将光线集束技术用于新的加速结构,用 SIMD 指令加快多种求交计算,如光线-面片、光线-空间网格单元、面片-空间网格单元等。这样将可进一步加速绘制计算。

致 谢 作者感谢如下资源的提供者:斯坦福大学的三维模型 <http://graphics.stanford.edu/data/3Dscanrep>, BART 动画光线跟踪基准测试场景 <http://www.ce.chalmers.se/research/group/graphics/BART> 和模型修补程序 <http://www.cs.wustl.edu/~taoju/code/polymender.htm>!

参 考 文 献

- [1] Wald I. Realtime ray tracing and interactive global illumination [Ph. D. dissertation]. Saarland University, Germany, 2004
- [2] Lext J, Moeller T A. Towards rapid reconstruction for animated ray tracing//Proceedings of the Eurographics 2001, Short Presentation. Manchester, 2001: 311-318
- [3] Glassner A S. Space subdivision for fast ray tracing. IEEE Computer Graphics and Applications, 1984, 4(10): 15-22
- [4] Havran V, Prikryl J, Purgathofer W. Statistical comparison of ray-shooting efficiency schemes. Vienna University of Technology, Austria: Technical Report TR-186-2-00-14, 2000
- [5] Kalos L S, Havran V, Balazs B, Szecsi L. On the efficiency of ray-shooting acceleration schemes//Proceedings of the 18th Spring Conference on Computer Graphics, Budmerice, Slovakia, 2002: 97-108
- [6] Stoll G. Interactive ray tracing, Part II: Achieving real time—optimizing techniques//Proceedings of the SIGGRAPH 2005 Courses, Los Angeles, California, 2005
- [7] Reshetov A, Soupikov A, Hurley J. Multi-level ray tracing algorithm//Proceedings of the ACM SIGGRAPH 2005, Los Angeles, California, 2005: 1176-1185
- [8] Stoll G, Mark W R, Djeu P, Wang R, Elhassan I. Razor: An architecture for dynamic multiresolution ray tracing. Department of Computer Sciences, University of Texas at Austin, USA: Technical Report TR-06-21, 2006
- [9] Wald I, Havran V. On building fast kd-trees for ray tracing and on doing that in $O(n \log n)$ //Proceedings of the IEEE Symposium on Interactive Ray Tracing 2006. Salt Lake City, USA, 2006: 61-69
- [10] Fujimoto A, Tanaka T, Iwata K. Arts: Accelerated ray-tracing system. IEEE Computer Graphics and Applications, 1986, 6(4): 16-26
- [11] Amanatides J, Woo A. A fast voxel traversal algorithm for ray tracing//Proceedings of the EUROGRAPHICS 1987. Amsterdam, North-Holland, 1987: 3-10
- [12] Jevans D, Wyvill B. Adaptive voxel subdivision for ray tracing//Proceedings of the Graphic Interface 1989. Toronto, Ontario, 1989: 164-172
- [13] Cazals F, Drettakis G, Puech C. Filtering, clustering and hierarchy construction: A new solution for ray-tracing complex scenes. Computer Graphics Forum, 1995, 14(3): 371-382
- [14] Klimaszewski K S, Sederberg T W. Faster ray tracing using adaptive grids. IEEE Computer Graphics and Applications, 1997, 17(1): 42-51
- [15] Muller G, Fellner D W. Hybrid scene structuring with application to ray tracing//Proceedings of the International Conference on Visual Computing 1999. Goa, India, 1999: 19-26
- [16] Parker S, Parker M, Livnat Y, Sloan P P, Hansen C, Shirley P. Interactive ray tracing for volume visualization. IEEE Transactions on Computer Graphics and Visualization, 1999, 5(3): 238-250
- [17] Devillers O. The macroregions: An efficient space subdivision structure for ray tracing//Proceedings of the EUROGRAPHICS 1989. Hamburg, Germany, 1989: 27-38
- [18] Yagel R, Kaufman A. Template-based volume rendering//Proceedings of the EUROGRAPHICS 1992. Cambridge, England, 1992: 153-167
- [19] Cohen D, Shefer Z. Proximity clouds—An acceleration technique for 3D grid traversal. The Visual Computer, 1994, 11(1): 27-38

- [20] Sramek M. Fast surface rendering from raster data by voxel traversal using chessboard distance//Proceedings of the IEEE Visualization'94. Washington DC, USA, 1994: 188-195
- [21] Freund J. Accelerated volume rendering using homogeneous region encoding//Proceedings of the IEEE Visualization'97. Phoenix, AZ, USA, 1997: 191-196
- [22] Parker S, Parker M, Livnat Y, Sloan P P, Hansen C, Shirley P. Interactive ray tracing for volume visualization. IEEE Transactions on Computer Graphics and Visualization, 1999, 5(3): 238-250
- [23] Sramek M, Kaufman A. Fast ray-tracing of rectilinear volume data using distance transformation. IEEE Transactions on Visualization and Computer Graphics, 2000, 6(3): 236-252
- [24] Wald I, Ize T, Kensler A, Knoll A, Parker S G. Ray tracing animated scenes using coherent grid traversal//Proceedings of the SIGGRAPH 2006. Boston, USA, 2006: 485-493
- [25] Reinhard E, Smits B, Hansen C. Dynamic acceleration structure for interactive ray tracing//Proceedings of the Eurographics Workshop on Rendering Techniques 2000. Brno, Czech Republic, 2000: 299-306
- [26] Gunther J, Fridrich H, Wald I, Seidel H P, Slusallek P. Ray tracing animated scenes using motion decomposition. Computer Graphics Forum, 2006, 25(3): 517-525
- [27] Wald I, Boulos S, Shirley P. Ray tracing deformable scenes using dynamic bounding volume hierarchies. ACM Transactions on Graphics, 2007, 26(1). Article No. 6.
- [28] Carr N A, Hoberock J, Crane K, Hart J C. Fast gpu ray tracing of dynamic meshes using geometry images//Proceedings of the Graphic Interface 2006. Quebec City, Canada, 2006: 203-209
- [29] Huang P J, Wang W C, Yang G, Wu E H. Traversal fields for ray tracing dynamic scenes//Proceedings of the ACM Symposium Virtual Reality Software and Technology 2006. Limassol, Cyprus, 2006: 65-74
- [30] Lext J, Assarsson U, Moeller T. A benchmark for animated ray tracing. IEEE Computer Graphics and Applications, 2001, 21(2): 22-31



LI Jing, born in 1972, Ph. D..

Her research interests include computer graphics and computational geometry.

WANG Wen-Cheng, born in 1967, Ph. D., professor, Ph.D. supervisor. His research interests include computer graphics, scientific visualization, and virtual reality.

WU En-Hua, born in 1947, Ph. D., professor, Ph. D. supervisor. His research interests include realistic image synthesis, virtual reality, and scientific visualization.

Background

Ray tracing is one of the main approaches for photorealistic rendering of 3D scenes. It can achieve high realistic rendering effects, because the rendering process of ray tracing is physically correct. However, high computation cost prevents ray tracing from practical application. Considering this, researchers have been developing spatial acceleration structures and ray traverse algorithms to improve its efficiency. In particular, ray tracing dynamic scenes has attracted much attention from the computer graphics association in recent years, due to its great demand in many applications such as virtual reality and games. As far as we know, the advanced ray tracing techniques can render a dynamic scene in several hundred thousands polygonal facets in several frames per second. But

this is still far below the requirements in real applications. In this paper, we design a novel structure for speeding up ray traversal in empty regions, and it is also very efficient for fast ray tracing dynamic scenes. Its basic idea is to produce big empty boxes to gather neighboring empty grid cells and associate the boxes with grids. Due to the uniform grids, these boxes can be built fast. Meanwhile, these boxes provide an approximation of empty regions in fewer partitions so that rays can traverse fast in empty regions via the boxes. Experimental results have shown the advantages of our new method over existing techniques, to be able to interactively ray trace large dynamic scenes, even in over 1G polygonal facets.