Science Discovery
2016; 4(5): 303-309
http://www.sciencepublishinggroup.com/j/sd
doi: 10.11648/j.sd.20160405.17
ISSN: 2331-0642 (Print); ISSN: 2331-0650 (Online)

Accelerating Many-Lights Rendering with Multi-directional Scene Voxelization

Meng Chunlei*, Su Tao

College of Computer Science, Beihang University, Beijing, China

Email address:
1240957820@qq.com (Meng Chunlei), sthp888@163.com (Su Tao)
*Corresponding author

To cite this article:
Meng Chunlei, Su Tao. Accelerating Many-Lights Rendering with Multi-directional Scene Voxelization. *Science Discovery.* Vol. 4, No. 5, 2016, pp. 303-309. doi: 10.11648/j.sd.20160405.17

Received: September 24, 2016; **Accepted**: October 10, 2016; **Published**: October 13, 2016

Abstract: Many scenes of interest to computer graphics applications contain a large number of dynamic light sources. Lighting is computationally expensive because it implies solving a visibility problem for every point light source. We present a method for voxel based approximation of the geometry of a scene. The ray is intersected with the approximation to accelerate the visibility determination. The scene geometry is approximated with a 2D array of voxelizations, with one voxelization for each direction from a dense sampling of the 2D space of all possible directions. The ray/scene intersection is approximated using the voxelization whose rows are most closely aligned with the ray. We support dynamic scenes with rigidly moving objects and complex dynamic scenes. The results of our experiments show our method can render scenes containing thousands lights.

Keywords: Realistic Rendering, Visibility Determination, Many-lights, Voxelizations

基于场景多方向体素化的多光源场景加速绘制方法

孟春雷*, 苏涛

计算机学院，北京航空航天大学，北京，中国

邮箱
1240957820@qq.com（孟春雷），sthp888@163.com（苏涛）

摘要：计算机图形应用中的很多场景包含大量的动态光源，对这类场景进行光照计算是一个耗时的过程，因为包含了光源与场景的可见性判断。本文提出一个基于场景多方向体素化的可见性计算方法，通过光线与近似场景求交来加速可见性判断。首先对场景进行预处理，根据对光线方向的离散进行三维场景多方向的体素化；然后在每一帧中当需要进行光线与场景求交时，通过使用与光线方向最接近的场景体素化结果来获取对应的体素行，根据其内是否包含几何面片来进行可见性判断。本文方法支持刚体运动以及其他复杂的动态场景。实验的结果表明本文方法可以快速渲染含有上千光源的复杂场景。

关键词：真实感图形绘制，可见性计算，多光源，体素化

1．引言

在计算机图形应用的很多场景中都会使用大量的动态光源，绘制含有多光源的复杂场景，对于增强图形的真实感沉浸感是一个非常重要的途径。在现有的图形硬件条件下，虽然已经可以实现对一些包含复杂几何模型的场景进行交互式的操作，但是对含有大量动态光源的场景进行绘制，速度还是比较慢，有待于提高。

光照计算是一个耗时的工作，因为它涉及到了光线和场景的可见性计算。尽管图形处理器可以快速地绘制复杂的场景，但是如果对于上千个光源，使用暴力方法上千遍地绘制复杂的几何场景还是极其耗时的。

假定输出图像的分辨率为w×h，绘制的几何场景包含n个光源，那么为了计算这n个光源对输出图像每个像素的光照情况，需要进行w×h×n根光线与几何场景的相交，这些光线是输出图像的像素到各个点光源的连线。例如输出图像的分辨率为1,024×1,024，并且场景中有10,000个点光源，那么将会产生100亿次左右的求交运算。

为了加速上述求交计算，本文提出了一种对于含有大量光源场景进行实时绘制的方法，该方法能够在固定时间内完成光线和场景几何的求交计算，确定光源可见性。在预处理阶段首先对光线在三维空间中的方向进行离散化，然后根据这些方向对几何场景进行体素化，生成多方向体素化的场景近似表示。多方向体素化的每个体素中如果包含了几何面片，将其设置为1，否则设置为0。在每一帧的绘制时，按照光线方向，找到对应方向的场景体素化结果，通过光线的起点位置找出光线穿过的该方向体素化结果的行，并且通过位移运算可判断光线穿过的一段体素是否含有1，从而确定该光线是否被场景的几何面片所遮挡。由于本算法采用了多方向的体素化对几何场景进行近似表示，所以避免了单一方向体素化所带来的需要沿着光线方向对体素按照一定步长进行搜索和0/1判断，所以能够在固定时间内确定光线的可见性。

固定时间内光线-场景的求交使得含有大量动态光源场景的实时绘制成为可能。本文方法预计算场景几何近似表示来加速计算每一帧中光线-场景的求交，因此可以实现多光源在帧间的自由运动，无需额外的时间开销。

由于上述方法需要对场景进行预处理，所以当场景中的几何对象运动或变形时，预先计算好场景的多方向体素化结果不能继续使用。因此，本文又给出了两种方法来实现对动态场景的支持。对于包含少量对象进行刚体运动的场景，在预处理时，除了对静态场景进行多方向体素化外，本文还单独对运动对象进行多方向体素化。在对每一帧的绘制中，除了同静态场景一样的进行光线与静态场景的可见性判定外，本文还将光线转换到运动对象的局部坐标系，判断光线是否被动态对象所遮挡。对于更加复杂的包含可变形对象的场景，本文在每帧均计算一个方向上的场景体素化数据，然后通过进行体素的旋转形成多方向的场景近似数据，以加速光线求交计算。

本文提出了一个在固定时间内确定光线与近似几何场景是否相交的方法，这种方法之所以能够加速确定可见性，是因为以下两个原因：

1) 本文对几何场景进行了传统的体素化。光线与几何场景的求交问题转化为光线与包含多边形面片的体素进行是否相交的判断；
2) 但是跟踪体素化的一条对角光线仍然需要很大时间开销。因此，本文将三维空间中光线的方向进行了离散化，然后对几何场景进行了多方向的体素化。这样可以为任何给定光线找到对应方向的场景体素化结果数据，直接获取体素行，避免了遍历对角光线的额外开销。

因此本文方法可以对多光源场景进行实时绘制。与光线跟踪算法相比，在可接受的误差范围内，本文方法相比光线跟踪方法更为高效。与之前的多光源实时绘制技术相比，本文方法在同等性能下的绘制质量会有显著提升。

2．准备工作

在直接光照和全局光照绘制中，本文都需要确定输出图像像素采样点与多光源的可见性。阴影图和光线跟踪是确定光源可见性的经典方法。然而，对于包含多光源的场景，这两种算法效率低下。目前大多数的研究主要采用以下两种策略对上述方法进行加速：一是对几何场景进行近似，减少光线求交的时间开销；二是光源聚合，减少参与求交计算的实际光源的数量。

2.1．基于阴影图的方法

阴影图方法是进行包含少量光源场景实时绘制的常用方法。阴影图是以光源位置为视点对几何场景进行绘制，从而得到场景的几何近似。它可以被GPU进行并行处理。然而，如果场景中的光源数较多，要为每个光源绘制一遍场景，生成对应的阴影图，时间开销过大。

Ritschel et al.的论文[1]中介绍了连贯阴影图，它由一些被压缩的深度图构成，这些深度图是预先从n个方向对场景进行绘制生成的，这里的n比光源的数量要小很多。在计算每条光线的可见度时，首先需要根据光线方向选取从最相近方向绘制的阴影图，然后进行深度比较来确定光线是否遮挡。在此方法的基础上，连贯表面阴影图方法(CSSM)[2]被提出。该方法支持场景间接光照计算。虚拟面光源是由一组虚拟点光源组成[3]，虚拟面光源的可见性可以利用扩展抛物线投影的CSSMs来计算，这避免了对每个虚拟点光源的单独计算。

不完美的阴影图(ISM)[4]是为针对多点光源场景绘制的阴影图的技术。为了提高绘制效率，不完美的阴影图的分辨率较低，并且对几何场景的采样点进行分批绘制。ISM是多光源实时绘制中常用的方法，所以本文在结论部分给出了本文的方法和ISM方法的比较。

层次化场景几何近似方法一直被用于加速阴影图的绘制中。例如，隐式可见性方法[5]使用了一种基于片状结构的四叉树表面近似方法，ManyLoDs[6]使用了场景几何层次划分方法。虚拟阴影图方法[7]将阴影投射场景几何划分成簇，为其绘制适当分辨率的立方体贴图，从而提高多光源复杂场景的绘制效率。

矩阵行列抽样(MRCS)[8]通过输出样本/光源对，来为输出样本和光源组计算典型阴影图。个别样本/光源对的可见性会被相关的典型阴影图代替。随后光源聚合方法会减少需要生成的典型阴影图的数量。MRCS算法可以在GPU中实现[9]，它减少了需要进行实际确定的可见性数量，因此提高了计算效率。另外，通过减少光源簇和输出图像的可见性估计可进一步提高该算法的效率[10]。

对于存在大量光源的场景，从这些光源绘制的阴影图之间存在冗余，随着场景复杂性和光源数量的增加，阴影贴图之间的冗余也会增加。本文方法对场景进行多方向的体素化，从而形成场景的三维几何近似数据，实现固定时间下光线-场景的求交。在本文方法中，虽然多方向的体素化数据也存在着冗余，但冗余与光线方向的离散化密切相关，在多光源或者复杂的场景中的冗余不会增加。本文所用的方法没有聚合光源，本文计算了场景中每条光线的可见性，提供了高质量的阴影效果。

2.2. 基于光线跟踪的方法

目前，许多基于光线跟踪的算法对几何场景进行近似来加速可见性的计算。微面绘制方法[11]使用点的层次结构进行场景的近似计算，这可以加速动态场景中光线的遍历和几何场景的更新。光线跟踪方法还可以通过对几何场景的体素化来进行加速[12]。本文所用的方法也借鉴了几何场景的体素化的思想，使用多方向体素化数据进一步减少了光线-场景求交的开销。

还有一些基于光线跟踪的方法采用对光源的简化来加速可见性判定。八叉树的层次结构用于实现基于位置和影响范围的光源聚合[13]。光源被分在许多松散的集合中，各个顶点的光源向量通过PCA进行压缩，在低频光照环境下实现了高质量和高帧频的绘制效果。光源切割[14]是基于二叉树中多光源阴影绘制的常用方法。对于输出的每个像素样本，在二叉树找到一个合适的切割，利用切割中的光源进行该像素的多光源光照计算，该方法假设所有光线均为可见。预计算光源切割方法对此方法进行了扩展，预计算了光源的可见性，因此它适合于光源不发生变化的静态场景。随后提出的双向光源分割方法[15]可以处理动态场景。

另外一些光线跟踪方法对光源和几何场景都进行了近似。例如可见度聚合[16]利用稀疏矩阵将场景几何与光源进行分组，非零子矩阵相当于场景几何与光源簇的可见度交集。这类方法提高了近似计算的效率。

本文提出的方法的实质是加速的光线跟踪算法，与上面讨论的基于光源聚合的光线跟踪加速算法不同，本文方法没有使用减少光源的方式来减少光线数量。这将会带来显著的质量优势，因为该方法计算出了所有光源的可见性，也确保了在动态光源下的时间相关性，避免了由于光源突然改变而产生的光亮突变，实现了每个光源自由移动。与基于层次化场景几何的方法相比较，该方法在光线-场景求交的时间开销明显较小。

3. 固定时间光线-场景的求交

光线-场景的求交计算可以通过对几何场景近似来进行加速。几何场景的多方向体素化通过算法1进行计算。

算法1 几何场景多方向体素化算法

输入：三角形表示的三维模型S

输出：多方向体素化的结果，采用二维数组V存储。

算法步骤：

1) 设定每个方向的角度的增量k；
2) 算法遍历两个方向的角度θ和Φ，并且每次增加k度；
3) 初始化设V[θ/k][Φ/k]=Ø；
4) 通过深度剥离方法(depth peeling)计算V[θ/k][Φ/k]，对于包含几何场景多边形面片的体素置1，空体素置0。然后当前层通过反复的深度剥离进行计算，直到深度的最后一层为止。

深度剥离是利用前一次的深度缓存值来计算当前值的方法。将当前层加入到体素化信息中，并将上一层更改为当前层，为下一次深度剥离做好准备。深度剥离会在正交投影下执行三次，分别在x轴，y轴，z轴方向上各执行一次。此三重投影使得表面定向的取样更具鲁棒性。例如，如果只沿z轴方向进行深度剥离，将会漏掉平行于z轴方向的表面。

事实上如果设定角度增量k = 2度，将会产生大小为90×90的体素化数组。如果单方向体素化的分辨率为128×128×128，体素化信息的每行会有128位，等于4个32位的整数型大小，每个体素化信息需要256KB，存体素化信息的二维数组大小将近2GB。

光线r和体素化二维数组V的求交采用了算法2进行计算。

算法2 光线的可见性判断算法

输入：二维的体素化数组V，角度增量k以及光线r；

输出：一个布尔值，即光线是否和场景有交点；

算法步骤：

1) 由光线r的方向确定角度θ和Φ；
2) 由θ和Φ找到光线r在V[θ/k][Φ/k]中穿过的一行row；
3) 由光线r的端点确定row的一个子区间，如果子区间不为0，则相交，否则不相交。

首先根据r的方向计算对应的离散化方向(θ, Φ)，然后根据生成几何场景多方向体素化结果中用到的k来获取对应方向的场景体素化结果V[θ/k][Φ/k]，从而进行该光线是否被近似场景所遮挡的结果。

如图1所示，在θ=30，Φ=42时的离散方向值与光线方向最为接近，k=2度，因此光线-场景求交需要使用体素化数据为V[30/2][42/2]中存储的体素化数组。

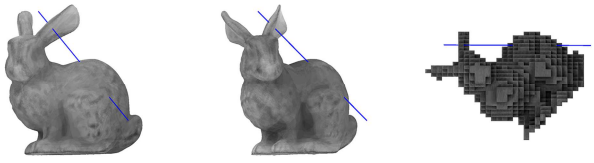

图1 计算θ，Φ示意图。

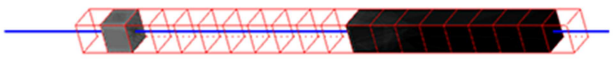

图2 光线和对应方向体素化结果相交列示意图。

光线-场景的交点可以通过V[θ/k][Φ/k]计算得到。每个体素中如果存在几何信息，标记为1，否则为0。光线的两个端点会投影在这行体素上。通过移位操作，得到这行体素删减后的长度。其中光线与场景相交的数据包含至少一位非零数据则说明相交。在图2中，该行体素有32位。整行信息为0000 0010 0000 0000 1111 1111 0000 0000。从i=4到i=27是光线段被裁减的范围，即0010 0000 0000 1111 1111 0000，可以看出数据不为0，所以此光线与场景相交（在兔子模型的耳朵和身体上）。

4. 动态多光源的实时绘制

固定时间内光线-场景加速求交方法使得动态多光源场景能够进行实时绘制，绘制采用了算法3。

算法3 静态场景动态多光源实时绘制算法

输入：场景S，相机位置C，n个光源的集合L，二维体素化数组V，角度增量；

输出：从相机C看到的场景S的光照情况；

算法步骤：

1) 在无照明条件下绘制输出图像I。
2) 枚举I的每一个像素p，计算其光照强度。对于一个确定的p，枚举每一个光源Li，由p和Li会确定一条光线r，借助于算法2可得到Li是否会照射到p。这样就得到了p的亮度。

图3展示了实验的3个静态场景。

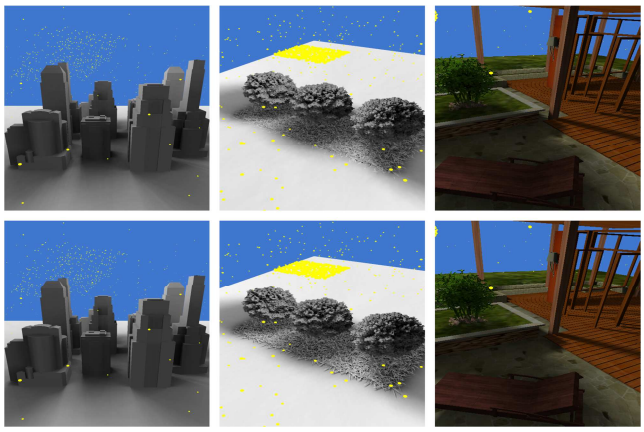

图3 静态实验场景，其中第一行为我们的方法对于不同场景的渲染结果，第二行为光线跟踪方法结果。

5. 动态场景的多方向体素化方法

对于静态几何场景，一个预先计算好的多方向体素化结果就可以支持大量的动态光源可见性的实时计算。然而，当场景中几何结构发生改变时，预计算结果不能正确表示变化后的几何场景，如果每一帧都使用算法1计算场景的多方向体素化数据，时间开销过大。因此，本文提出了以下两种方法实现对动态场景的支持。

5.1. 包含刚体运动对象的场景

一些场景中会有多种对象，每类种对象又有多个实例，各个实例在场景中进行不同的运动。本文采用算法4对这类场景的多光源绘制进行支持。

算法4 对包含刚体运动对象的场景光线可见性判定算法

输入：对于静态场景的二维体素化数组V0,每种动态对象DOTi的多方向体素化数组Vi，当前每个动态对象实例DOj的坐标CSj，光线r；

输出：一个布尔值，如果即光线是否和场景有交点；

算法步骤：

1) 使用算法2判断是否和V0有交点，若相交，则返回遮挡；
2) 否则，算法会枚举每一个动态对象实例DOj，设其类型为DOTi，根据CSj将光线r变换到Vi的坐标系下rj，然后同样使用算法2计算rj是否与Vi有交点。若有则返回遮挡；
3) 若光线与所有动态物体都不相交，返回不遮挡；否则返回遮挡。

当光线首先与场景的静态部分求交时，如果没有交点，光线将会转换到场景中每个对象实例的局部坐标系，之后光线会与该种对象的多方向体素化结果求交。例如，对于图4中左边的飞机场景，预先计算出了两个体素化二维数组：一个是建筑物VB，另一个是飞机VP；然后光线r与场景VB的求交，如果无交点，则需要与VP求交来获取是否被遮挡。

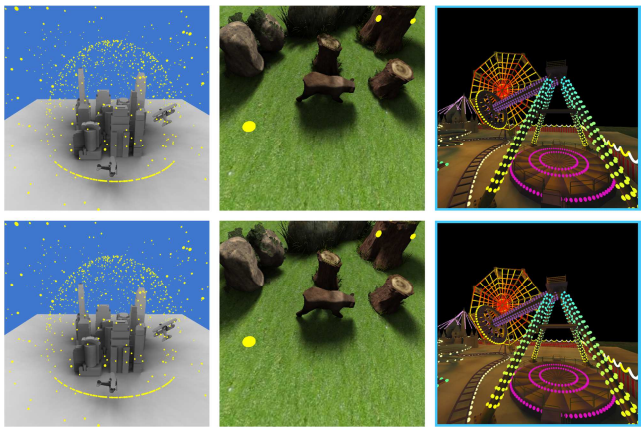

图4 动态实验场景，其中第一行为我们的方法对于不同场景的渲染结果，第二行为光线跟踪方法结果。

5.2. 包含变形对象的场景

对于包含非刚体运动对象或者可变形对象的场景，可以采用算法5在每帧中重新生成场景的多方向体素化结果。

算法5包含变形对象的场景的多方向体素化算法

输入：当前帧的动态场景S，角度增量k；

输出：多方向体素化结果数组V；

算法步骤：

1) 按照算法1中的方法计算V[0][0]。
2) 枚举两个方向的转角（θ，Φ），将V[0][0]旋转θ，Φ的角度得到V[θ/k][Φ/k]。

本文没有采用深度剥离计算场景多方向体素化数据，只是用算法1计算了初始的体素化数据V[0][0]。然后采用旋转初始的体素化数据的方式计算其它方向的体素化数据。一个包含场景多边形面片的体素(i,j,t)，在旋转坐标系之后对应的体素点为(i',j',t')。通过旋转初始的体素化信息来计算其它体素信息虽然在计算精度上稍有损失，但比从原始场景中进行多方向深度剥离的计算方法要快很多。

6. 实验与结果分析

我们对本文提出的方法在以下场景中进行了实验：City（871 k，图3左），Trees（626k，图3中），Garden（416k，图3右），Plane（982k，图4左），Bear（1486k，图4中），Park（499k，图4右）。除了Park场景有7088个光源外，其它场景都包含1024个光源。城市、树林和花园是静态场景，而飞机、熊和公园都是动态场景。本文中所有的性能指标都是在包含如下硬件配置的工作站上测试得到的：3.5GHz Intel(R) Core(TM) i7-4770 CPU，8GB的RAM，以及NVIDIA GeForce GTX 780的显卡。

6.1. 质量实验

本文通过两个误差指标来对绘制的质量进行评价。第一个是εv，它是某一像素采样点上的可见性判断错误的光线数量与所有光线数量的百分比。假设场景中有1000个光源，在像素p上有10个光源被错误的标记为可见，有5个光源被错误的标记为不可见，那么εv = (10+5)/1000 = 1.5%。第二个是εs，它是像素点的阴影值相对误差。在上面的例子中，εs = (10-5)/1000 = 0.5%。εv是一个严格的误差指标，而εs相对较为宽松，因为部分错误可以正负相互抵消。然而对于结果图像，εs是一个较好的像素可见性指标。像素点上正确的可见性和阴影值是通过光线跟踪算法得到的（本文使用了NVIDIA的Optix光线跟踪[17]）。此外，还对本文的方法和不完美的阴影贴图(ISM)[4]进行了比较，ISM是目前多光源实时绘制较好的方法。

表1显示了使用本文的方法输出的结果图像像素平均可见性误差εv和像素平均阴影值误差εs。另外我们还进行了本文方法和ISM方法的对比。表1给出的本文方法与ISM方法误差比较是在相同时间性能的情况下进行的。可以看出，本文的方法产生的误差始终比ISM方法产生的误差要小。

表1 与ISM的误差对比。

Scene		City	Trees	Garden	Planes	Bear	Park
εv[%]	Ours	1.2	4.0	5.7	1.8	3.3	21.0
	ISM	5.9	7.5	16.0	5.7	6.1	27.0
εs[%]	Ours	0.5	2.1	2.2	0.8	2.1	7.5
	ISM	3.2	4.7	8.8	4.1	4.9	12.1

表2给出了随着光源数量的改变本文的方法误差的变化情况。误差变化很小，这是因为这些误差度量是相对所有光源产生的光线数量而言的，已经进行过规范化处理。上述实验都用到一个128×128×128分辨率的多方向体素化数据来进行机光线-场景求交的。表3给出了本文方法在低分辨率体素化数据下的质量误差。

表2 不同光源数的误差。

Lights		1,024	2,048	4,096	10,000
City	εv[%]	1.2	1.2	1.2	1.2
	εs[%]	0.5	0.5	0.5	0.4
Garden	εv[%]	5.7	5.7	5.7	5.7
	εs[%]	2.2	2.2	2.2	2.1

表3 不同分辨率下的误差。

Voxelization res.		32 x 32 x 32	64 x 64 x 64	128 x 128 x 128
City	εv[%]	4.7	2.2	1.2
	εs[%]	2.9	1.1	0.5
Garden	εv[%]	19.7	8.5	5.7
	εs[%]	16.0	5.1	2.2

本文做的上述实验还用到了一个90×90的多方向体素化结果数组，这相当于旋转角度增量为2度。表4给出了在较小的体素化数组下的近似误差。要使用60×60的体素化数组，旋转角度增量就是3度，在产生相同效果的情况下，内存使用降低了2.25倍。

表4 不同方向数体素化情况下的质量误差。

Voxelization rotation res.		30 x 30	60 x 60	90 x 90
City	εv[%]	2.5	1.5	1.2
	εs[%]	0.9	0.6	0.5
Garden	εv[%]	9.0	6.4	5.7
	εs[%]	3.2	2.3	2.2

6.2. 速度实验

下面将本文方法与光线跟踪，ISM方法在绘制速度上进行了对比。对于光线跟踪方法，本文使用了NVIDIA Optix的层次包围盒（BVH）[17]对其进行加速。表5给出了本文方法的每帧渲染时间以及对相对于光线跟踪方法的加速比。可以看出本文方法比光线跟踪方法更为高效。飞机场景是用算法4进行绘制的，其中包含两个多方向体素化数据：建筑物和飞机，并且每条光线最多要进行三次求交计算，一次是与建筑物求交，另外两次是与飞机的两个实体求交。对于熊场景绘制的加速比较低，这是因为场景中有变形对象需要用算法5在每一帧内重新来计算多方向体素化数据。

表5 与光线追踪方法的速度对比。

Scene	City	Trees	Garden	Planes	Bear	Park
Ours[ms]	43	46	64	93	586	839
Speedup vs. RT	38.2x	76.4x	37.7x	21.5x	5.6x	33.9x

本文还尝试了在同等质量效果下与ISM方法进行对比。但是即使增大ISM的样本点数量，也达不到本文方法产生的质量效果，如图5所示。增加的样本数量一旦超出其处

理能力，就会使ISM方法比光线跟踪法还要慢。ISM中的样本点仅与场景中的三角形拓扑结构相关，与三角形的位置关系不大，所以可以在变形对象中非常方便地更新采样点的位置，这使得ISM在含有变形对象的几何场景中具有性能上的优势。在熊场景中ISM方法要比本文方法快5倍，但是也存在着两倍的阴影误差 ε s。总之，如表1所示，与ISM方法相比，本文的方法绘制的图像质量更好。

表6给出了本文方法每帧的绘制时间。与预期结果一致，在静态场景和包含刚体运动对象的场景中，当光源数量翻倍，绘制每一帧的时间开销也翻倍，这是由于需要进行光线-体素化结果的求交次数的增加而造成的。在熊场景中，体素化计算要花费很多时间，所以会有一些额外的开销。

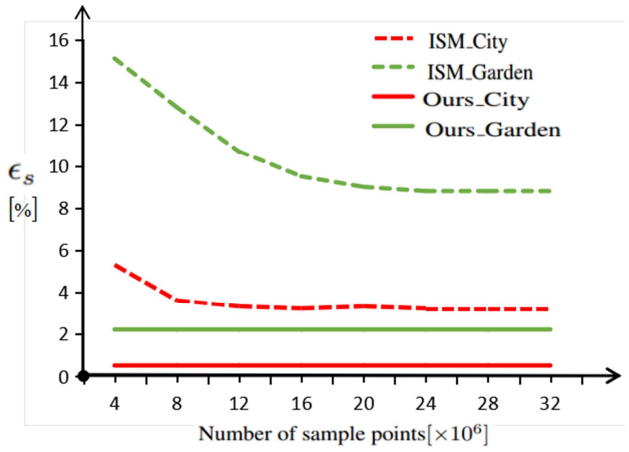

图5 ISM误差随采样点数量的变化。

表6 随光源数量变化的绘制时间。

Lights	512	1,024	2,048	4,096	10,000
City	22	42	84	171	416
Garden	33	64	130	262	639
Trees	23	46	93	181	447
Planes	46	93	185	370	895
Bear	547	586	648	787	1,185

6.3. 局限性

本文方法主要基于几方面的近似处理。首先，场景几何是由多方向体素化结果近似表示的。第二，光线方向使用了基于角增量的离散化处理。这些近似处理产生的质量误差可以被很好地控制，并且利用了GPU存储和计算能力的优势。本文的方法可以减少计算每条光线的复杂度，但需要增加额外的存储开销。与光线跟踪加速方法类似，本文的方法在处理动态场景时需要更新加速数据结构。与分层数据结构不同，场景的多方向体素化数据可以利用GPU的深度剥离进行计算。

7. 结论和未来工作

本文提出了一个基于固定时间内光线与场景求交的多光源实时绘制方法。本文的方法在速度上与光线跟踪方法相比具有明显优势，而且绘制效果良好。与不完美的阴影图方法相比，在相同时间性能下本文的方法可以产生更为准确的结果。本文方法可以计算出每个光源的亮度，没有使用光源聚合的方法。因此，当光源从一个区域移动到另一个区域，可以产生平滑的阴影效果，避免了由于光源组拓扑突变而产生的亮度突变。本文的方法中可见度不是通过插值法计算的，而是通过每条光线和场景的求交运算得到。

光线-场景几何求交法在计算机图形学和加速绘制方法中属于基础操作，其应用包括环境光遮挡，软阴影，高光和漫反射。而对于以下两个问题，本文给出了其区别，第一个问题是某光线与场景几何是否相交了，第二个问题是光线-场景相交的交点在哪里。一些应用包括本文提到的多光源绘制，仅仅需要回答第一个问题就能解决，而其它应用，例如镜面反射，还需要回答第二个问题。第一个问题只需要测试体素化数据的行截取的光线长度不为零即可。第二个问题需要找到第一个非零位在截取的行中的位置，最多进行logw次二分查找就可以找到结果，w是体素行的分辨率。

本文方法主要依赖于场景几何的近似计算，减少了几何场景的复杂性。由于使用多方向体素化对几何场景进行近似表示形式比较统一且简单，所以它的结构、存储很适合用GPU进行处理。本文的方法将光线与场景的求交运算的开销降到了最低。光线-场景求交法通过“内存丢弃”的方法进行加速。在现当前的GPU下本文的方法已经能成功运行，并且有可能成为图形应用中估算场景-光线交点的标准化方法，就像Z缓存算法代替了复杂多边形的可见度排序算法。

本文方法实现了在交互式图形应用中实现复杂的动态光照效果。然而动态光源数量的增加对于光照设计和动画确实是一个挑战。在未来工作中的一个重要的方向就是在复杂光照场景中如何对成千上万的光源及其运动轨迹进行设计与编辑。

致谢

本文为国家八六三重点项目课题(2013AA01A604)；国家自然基金重大项目课(61272349,61190121,61190125)的阶段性成果之一。

参考文献

[1] Ritschel, Grosch, Kautz, Mueller, Interactive illumination with coherent shadow maps[J]. In Proceedings of the 18th Eurographics conference on Rendering Techniques. Eurographics Association, 2007, 61 - 72.

[2] Ritschel, Grosch, Kautz, Seidel. Interactive global illumination based on coherent surface shadow maps[J]. In Proceedings of Graphics Interface 2008. Canadian Information. Processing Society. 2008, 185 - 192.

[3] Dong, Grosch, Ritschel, Kautz, Seidel. Real-time indirect illumination with clustered visibility[J]. In VMV. 2009, 187－196.

[4] Ritschel, Grosch, Kim, Seidel, Dachsbacher, Kautz. Imperfect shadow maps for efficient computation of indirect Illumination[J]. ACM Transactions on Graphics (TOG) 27, 2008, 5, 129.

[5] Dong, Kautz, Theobalt, Seidel. Interactive global illumination using implicit visibility[J]. In Conference on Computer Graphics & Applications. 2007, 77－86.

[6] Hollander, Ritschel, Eisemann, Boubekeur.Manylods: Parallel many-view level-of-detail selection for realtime global Illumination[J]. In Computer Graphics Forum. Vol. 30. Wiley Online Library, 2011, 1233－1240.

[7] Olsson, Sintorn, Kampe, Billeter, Assarsson. Efficient virtual shadow maps for many lights[J]. In Proceedings of the 18th meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. ACM, 2014, 87－96.

[8] Hasan, Pellacini, Bala. Matrix row-column sampling for the many-light problem[J]. In ACM Transactions on Graphics (TOG). Vol. 26. ACM, 2007, 26.

[9] Wang, Huo, Yuan, Zhou, Hua, Bao. Gpu-based out-of-core many-lights rendering[J]. ACM Transactions on Graphics (TOG) 32, 6, 2013, 210.

[10] Wang, Huo, Jin, Bao. A matrix sampling-and-recovery approach for many-lights rendering.Gpu-based out-of-core. ACM Transactions on Graphics (TOG) 34, 6, 2015, 210.

[11] Ritschel, Engelhardt, Grosch, Seidel, Kautz, Dachsbacher. Micro-rendering for scalable, parallel final gathering[J]. ACM Transactions on Graphics (TOG) 28, 2009, 5, 132.

[12] Knichols, Penmatsa, Wyman.Interactive, multiresolution image-space rendering for dynamic area lighting. In Computer Graphics Forum. Vol. 29. Wiley Online Library, 2010, 1279－1288.

[13] Paquette, Poulin, Drettakis. A light hierarchy for fast rendering of scenes with many lights[J]. In Computer Graphics Forum. Vol. 17. Wiley Online Library, 1998, 63－74.

[14] Walter, Fernandez, Arbree, Bala, Donilian, Greenberg. Lightcuts: A scalable approach to illumination[J]. Ac Transactions on Graphics 24, 3, pgs. 2005, 1098－1107.

[15] Walter, Khungurn, Bala. Bidirectional lightcuts [J]. ACM Transactions on Graphics (TOG) 31, 2012, 4, 59.

[16] Wu, Chuang. Visibilitycluster: Average directional visibility for many-light rendering. Visualization and Computer Graphics, IEEE Transactions on 19, 9, 2013, 1566－1578.

[17] NVIDIA. 2016. Nvidia optix ray tracing engine. http://developer.nvidia.com/optix.