文章编号：2096-1618(2021)03-0300-05

大规模三维模型加速渲染技术的研究与应用

刘一明， 何晓曦， 黄世贤
(成都信息工程大学软件工程学院，四川 成都 610225)

摘要：随着在游戏和仿真领域中对大规模场景的要求日益提高，为了保证一般计算机在大规模三维模型的场景中仍能够较高帧率运行，基于Morton码提出了一种场景遍历算法。该算法根据满四叉树节点编码思想对线性四叉树的Morton码进行了改进，提出了一种线性多段结构，并且结合了自底向上合并的BVH来减少遍历节点，最后利用视锥体对场景内的物体进行快速裁剪。该算法结合场景中的多层均匀网格，利用CPU的多核并行计算能力，根据上层遍历结果，快速得到下层输入，能够快速对场景中的物体进行可见性裁剪，提高计算机在大规模模型场景的渲染帧率。实验结果表明了该算法的正确性和可行性，与传统的加速结构相比，帧率有了明显的提升。

关 键 词：大规模场景；加速结构；Morton码；四叉树；BVH
中图分类号：TP312 **文献标志码**：A
doi：10.16836/j.cnki.jcuit.2021.03.010

0 引言

实时计算机图形学在AAA级游戏、智能城市仿真等多个领域的应用，正在追求更广阔的世界、更真实的渲染和更丰富的细节。虽然硬件的不断升级在很大程度上支持了渲染能力，但实时渲染大型复杂场景仍然是一个很大的挑战。大量的研究工作致力于减少对渲染流水线[1]的输入数据，根据视觉感知减少模型的细节表现[2-3]，并利用CPU或GPU的多核并行机制在更短的时间内完成更多的计算[4-5]。基于前人的研究提出了MBVH(morton bounding volume hierarchy)，旨在减少同时绘制的模型对象数量，降低渲染压力，提高场景的整体帧率。MBVH借鉴了地理信息系统的瓦片技术，利用地图分割和分级的思想，对三维场景进行类似的处理。如何组织场景结构，如何得到“瓦片”的可视信息，是文中主要研究内容。

计算机图形加速绘制算法的核心基于空间数据结构。目前，场景管理算法有两大发展方向[6]，一个是面向高性能，另一个是面向交互。面向高性能的常见的结构有二叉树、四叉树、八叉树[7]、KD树[8]等。其中，KD树的性能相对较好，但是在处理大规模场景时，初始化过程非常耗时。二叉树不适用于室外场景，而适用于复杂的室内场景，其优点体现在快速碰撞检测[9]、可见性确定等方面。在大规模场景中，八叉树由于其特性，容易产生聚集上层场景树结构的小几何对象。面向交互的场景管理大多采用场景图[10]的方式，使用这种方式能较好地修改和重用对象，但是不能对加速渲染起到很好的作用。

Daniel等[11]不使用传统的KD树、四叉树等来构建BVH，而是用一种“邻近-规则”簇。该文使用均匀网格包围所有图元，给均匀网格分配Morton编码，然后利用Morton码的Z型曲线进行一维的范围搜索来识别最近的图元。Yi Lei等[12]提出了(global multi-scale grid integer coding model，GMGICM)，该方法相比Geohash的Morton数据排布更具聚合性，理论上有利于高效的遍历查询。Zonghui Li等[13]利用基于Morton码的均匀网格分割场景后，在压缩路径时使用并行算法，提高了树的构建和遍历速度。ArsènePérard-Gayot等[14]通过将八叉树嵌套到网格中，从而构建出一种平面的、非层次的空间结构，然后对这些网格进行合并处理，并且在射线遍历过程中分离网格边界。该方法可以跳过空白区域以避免重复的检测，进而提高射线的遍历性能。

与此同时，一些基于GPU的构建算法也被提出。Lauterbach等[15]第一次提出了LBVH(linear bounding volume hierarchy)的概念，首先对每个物体的AABB包围盒提取中心坐标，然后对这些物体中心坐标所在的Morton网格进行线性排序。Pantaleoni等[16]对LBVH进行改进，提出了HLBVH(hierarchical linear bounding volume hierarchies)算法，使用了SAH构建BVH顶点的Morton簇。杨鑫等[17]则是把BVH的构建分为不同阶段并设计了不同的策略。尽管这些算法在一定程度上提高了BVH的构建和遍历效率，但是对于大规模场景的处理还存在不足。

在众多研究者已经提出并实现的层次包围盒构建和遍历算法的基础上，结合Morton编码的特性和视锥

收稿日期：2020-11-26

体裁剪,实现了基于 CPU 的大规模场景的层次包围盒的构建与遍历。

1　相关理论知识

1.1　Morton 编码

Morton 码是对均匀网格进行编码的一种算法,Morton 码编码结果展现为一种 Z 形的填充曲线,通常用于四叉树的四进制或十进制编码。给定一个 n 维的整数坐标值,将其二进制下的每一位交错排布组合成一个新的二进制数,这个数即为该 n 维整数坐标对应的 Morton 码。以二维坐标(x,y)为例,设 x 和 y 二进制上的每一位为 x_i 和 y_i,即

$$x=\{x_{n-1}\cdots x_1x_0\}$$
$$y=\{y_{n-1}\cdots y_1y_0\}$$

则对应的 Morton 码为:

$$\{y_{n-1}x_{n-1}\cdots y_1x_1y_0x_0\}$$

1.2　BVH

BVH(bounding volume hierarchy)是一种基于“物体”的场景管理技术,广泛应用于碰撞检测、射线相交测试之类的应用场景中。比较常见的用法是把不规则物体用一个 AABB(axis-aligned minimum bounding box)包围起来,然后就可以用这个包围盒的相交测试结果来代替原来顶点复杂的物体的相交测试结果。

1.3　视锥体裁剪

视锥体(frustum),是指场景中摄像机可见的一个锥体范围。它由上、下、左、右、近、远共 6 个面组成。在视锥体内的景物可见,反之则不可见。在较大的三维场景中,为提高性能,不需要渲染整个场景中的物体,而是通过计算其中与视锥体有交集的对象进行绘制。视锥体裁剪算法要实现的功能就是视锥体和点或者线段求交,将视锥体内的部分显示出来,外部的视为不可见。通过视锥体裁剪算法能判定出大量不可见的部分,大大减少了送入绘制管道的数据量,从而极大地提高了图形的显示性能。计算出视锥体 6 个面的空间平面方程,将点坐标分别代入 6 个面的平面方程做比较,则可以判断点是否在视锥体内。空间平面方程可表示为:

$$Ax+By+Cz=0 \tag{1}$$

对于点(x_1,y_1,z_1),有:

若 $Ax_1+By_1+Cz_1=0$,则点在平面上;

若 $Ax_1+By_1+Cz_1<0$,则点在平面的一侧;

若 $Ax_1+By_1+Cz_1>0$,则点在平面的另一侧。

2　Morton 层次包围盒(MBVH)

算法首先给三维场景划分多层均匀网格包围盒,并利用位运算计算出每个网格包围盒的 Morton 编码,然后把场景中的所有物体分配到各自的包围盒中,最后利用视锥裁剪算法,自上而下遍历包围盒,得到包围盒的可视情况,利用可视结果来渲染场景物体。算法的总体流程如下:

(1)给三维场景划分多层均匀网格包围盒;

(2)计算出每个网格的 Morton 编码当作网格包围盒数组的下标;

(3)遍历所有模型,得到模型所在包围盒下标;

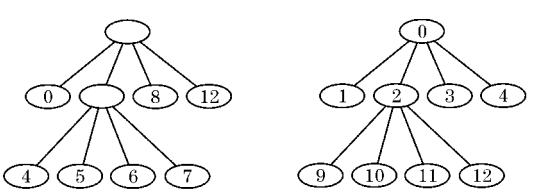
(4)对包围盒自底向上进行合并;

(5)利用视锥裁剪计算包围盒可视情况。

2.1　线性四叉树节点编码的改进

线性四叉树有两种建立方法,一个是以自上而下分裂的方式在建立四叉树的过程中逐步产生 Morton 码,另一种则是先计算每个网格的 Morton 码,然后再按照一定的扫描方式采用自下而上的合并方法建立四叉树。这里采用自上而下的方法。Morton 码有十进制和四进制两种方式。十进制的 Morton 码是 0 ~ M 的自然数,在建立的过程中可以不用开辟地址码和深度的内存数组,而直接用网格数组下标来代替。采用十进制的 Morton 码,不仅可以提高运算速度,而且可以节省内外存空间。所以,选取十进制的 Morton 码来实现算法。

在传统的十进制 Morton 编码中,叶节点对应区域的地址码从 0 开始,在合并子区域时,用子区域中的最小地址码表示合并后区域的地址码。虽然地址码中隐含了叶节点的位置,但是它不能很好地表示区域之间的空间关系,且编号对应的区域大小随树型结构的改变而改变。为了弥补这些缺点,本文对四叉树 Morton 编码进行改进,利用满四叉树的编码思想为每个叶节点赋值,如图 1 所示。



(a)传统四叉树节点编码　　(b)改进后的四叉树节点编码

图 1　传统与改进后的四叉树节点编码比较

十进制 Morton 码实际上是由行列号中的二进制

数交叉结合的结果,若行列号为(5,3);则行列号的二进制表示为 $i=0101$,$j=0011$,交叉结合后得到 Morton 码的二进制表示为 00011011,将其转为十进制为 27。交叉过程如图 2 所示。

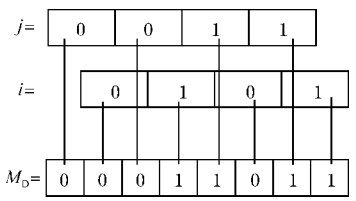


图 2 二进制行列号计算 Morton 码

所以,可以使用位运算快速得到相应的 Morton 码 M_D。首先以此取出行号 II,列号 JJ 的二进制数中的各位数值 P_{it} 和 P_{ij}:

$$P_{it}=(\mathrm{II}\&2^{t-1}) \tag{2}$$

在取出行列号的二进制各位值后,依次交叉放入 M_D 码的变量中,利用按位操作的或运算,对于行而言:

$$M_D=M_D\mid(P_{it}\ll t) \tag{3}$$

对于列而言:

$$M_D=M_D\mid(P_{it}\ll t-1) \tag{4}$$

根据改进的四叉树思路得到对应 Morton 码 M_D:

$$M_D=M'_D+\sum_{i=0}^{i<n}4^i \tag{5}$$

其中 $\sum_{i=0}^{i<n}4^i$ 为 $2^n\times2^n$ 网格图层对应满四叉树的内部节点的个数。令 $n=2$,网格 Morton 码的变化如图 3 所示。

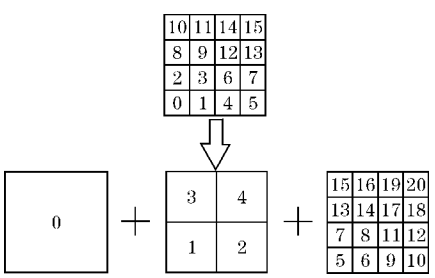


图 3 改进四叉树的 Morton 表示

2.2 自底向上合并包围盒

传统的 Morton 的生成和建立过程有两种不同的方案:一是用自上而下分裂的方式在建立四叉树的过程中逐步产生 Morton 码,另一种方案是先计算每个网格的 Morton 码,然后按一定的扫描方式采用自下而上合并的方法建立四叉树。第一种方法需要大量运算,合并包围盒速度较慢,所以本文采用第二种方式并结合 BVH 进行改进。

根据前面的满四叉树结构生成线性分段包围盒数组,数组大小为 $\sum_{i=0}^{i<n}4^i$,下标从 0 到 $\sum_{i=0}^{i<n}4^i-1$。借鉴传统 Morton 网格自底向上合并网格的办法,从下标 $\sum_{i=0}^{i<n}4^{i-1}-1$ 到 $\sum_{i=0}^{i<n}4^i-1$ 每隔四个包围盒检测其中是否有物体存在。若全部没有物体,则将 4 个包围盒中下标最小的状态属性赋值为 1(默认为 0);如果有一个包围盒里面存在物体,则直接跳到下一组包围盒进行检测,如图 4 所示。

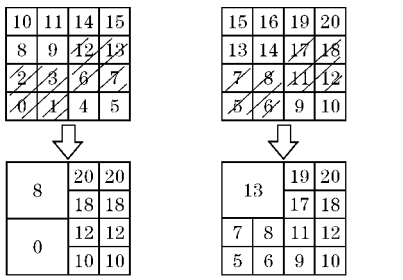


图 4 传统与改进后的 Morton 网格合并结果

虽然改进后的 Morton 网格没有传统的合并程度高,但是在后续的视锥裁剪阶段,高合并度的物体网格会严重影响渲染的效率,在极端的情况下,可视范围只有 4 个底层网格的大小也会渲染场景的全部物体。

3 实验

3.1 实验环境

Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz 2.80 GHz;内存:16 GB;显卡:GTX1060;操作系统:windows 10。

3.2 实验结果

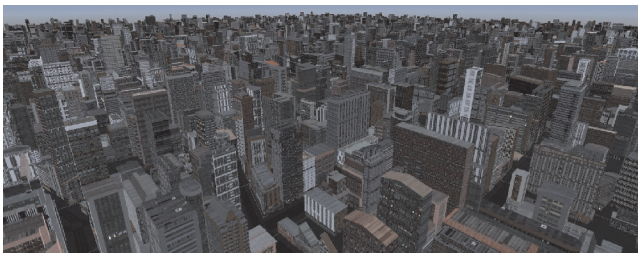
算法使用 C#语言实现。C#提供了高性能的多线程系统,该多线程系统可以使用全部可用的 CPU 核心,所以能很好地利用 CPU 的并行计算能力。实现所用的平台 Unity3D 是一个成熟的游戏引擎,能够较快地开发软件,并且能够实时观测各个方面的实验数据。下面的实验是在不同规模的场景中使用 MBVH 算法与常见加速结构算法进行比较分析。

表 1 实验场景信息

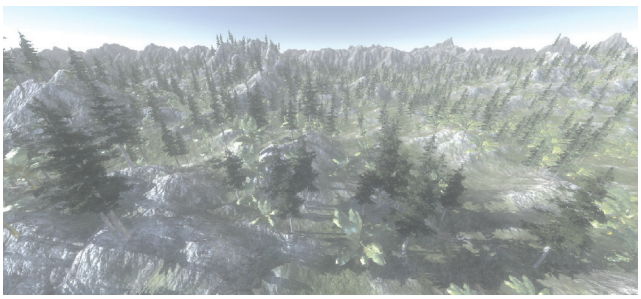
	场景 a	场景 b	场景 c
三角面片/K	1034	3942	8784
大小/MB	575	1721	3713

测试场景共有 3 个,表 1 是各个场景的详细属性信息,最小的测试场景约有1034 K个三角面,最大的测试场景约有8784 K个三角面,场景的内存占用从

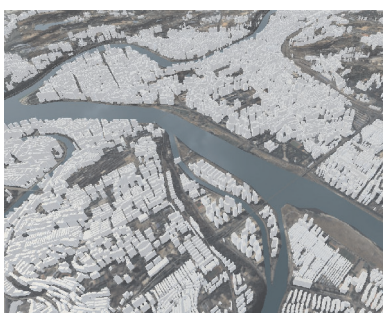
575 MB到3713 MB。图 5 是实验场景的展示效果,实验场景(a)到(c)的规模逐渐扩大。



(a)建筑场景



(b)森林场景



(c)交通仿真场景

图 5　实验场景

表 2 对比了 MBVH 算法与四叉树、八叉树、KD 树在不同规模和复杂程度场景的平均帧率。由表 2 可知,在场景规模较小,场景规模不大的情况下,4 种算法的表现相差不大。但是当场景规模逐渐扩大,物体的分布情况逐渐复杂,传统遍历算法的效率会大大降低,不光是遍历深度,次数的增多,重复的遍历也大大拖低了这些算法的效率。而文中 MBVH 算法,利用改进的线性四叉树 Morton 编码,简化了线性四叉树的邻域查询,同时结合 BVH 与视锥裁剪,提高了查询效率,在规模越大的场景,其优势越明显。

表 2　与常见加速结构对比实验(平均帧率)

	场景 a	场景 b	场景 c
不裁剪	233	67	23
四叉树	278	136	78
八叉树	280	133	69
KD 树	283	135	80
MBVH	307	155	103

4　结束语

通过分析目前大规模模型场景难以高帧率渲染的现状,提出并实现了一种线性多段结构,并且结合自底向上合并的 BVH,最后利用视锥体对场景内的物体进行快速裁剪,提高了整体渲染帧率。MBVH 算法的贡献主要有两点:

(1)使用 Morton 编码和满四叉树特性,构建了分段线性数据结构,再结合 BVH,实现了分段层次包围盒结构。

(2)根据视锥裁剪特点,自底向上合并空节点,减少包围盒数量,提升了遍历效率。

MBVH 算法基于上面两点优化的同时结合 CPU 多核并行运算,实验表明,与传统场景加速结构相比,能够大大提高大规模模型场景的运行帧率。

参考文献:

[1] Mohammad Mirzadeh, Arthur Guittet, CarstenBurstedde, et al. Parallel level-set methods on adaptive tree-based grids[J]. Journal ofutational Physics, 2016, 322:345-364.

[2] Loubet G, Neyret F. Hybrid mesh-volume LoDs for all-scale pre-filtering of complex 3D assets[J]. Computer Graphics Forum, 2017, 36(2):431-422.

[3] Argudo O, Andujar C, Chica A. Image-Based Tree Variations[J]. Computer Graphics Forum, 2020, 39(1):174-184.

[4] Zonghui Li, Yangdong Deng, Ming Gu. Path compression kd-trees with multi-layer parallel construction a case study on ray tracing[C]. Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, 2017, (16):1-8.

[5] MarekVinkler, Jií Bittner, VlastimilHavran, et al. Massively Parallel Hierarchical Scene Processing with Applications in Rendering[J]. Computer Graphics Forum, 2013, 32(8):13-25.

[6] Xiaoping Liu, YeYu, JieqiongKong, et al. Scene management strategy in collaborative rendering environment[J]. Journal of System Simulation, 2007(1):85-88.

[7] Yu Zhang. A dynamic scene management based on Octree[J]. Computer Knowledge and Technology, 2015(14):54-57.

[8] 张琴,蔡勇,常伟杰.基于空间分割的局部 KD 树动态构建算法[J].机械工程师,2010(12):30-32.
[9] 冯立颖.碰撞检测技术研究综述[J].计算机时代,2014,(8):7-10.
[10] Henry sowizral. Scene Graphs in the New Millennium[J]. Computer Graphics and Applications (S0272-1716),2000,20(1):56-57.
[11] Meister Daniel,Bittner Jiri. Parallel Locally-Ordered Clustering for Bounding Volume Hierarchy Construction[J]. IEEE Transactions on Visualization & Computer Graphics,2018,24(3):1345-1353.
[12] Yi Lei,Xiaochong Tong,Yongsheng Zhang,et al. Global multi-scale grid integer coding and spatial indexing:A novel approach for big earth observation data[J]. ISPRS Journal of Photogrammetry and Remote Sensing,2020,163:202-213.
[13] Zonghui Li,Yang Deng,Ming Gu. Path compression kd-trees with multi-layer parallel construction a case study on ray tracing[J]. Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games,I3D 2017:161-168.
[14] J Hendrich,A Pospíil,D Meister,et al. Ray Classification for Accelerated BVH Traversal[J]. Computer Graphics Forum,2019,38(4):49-56.
[15] Lauterbach C,Garland M,Sengupta S,et al. Fast BVH Construction on GPUs[J]. Computer Graphics Forum,2009,28(2):375-384.
[16] Pantaleoni J,Luebke D. HLBVH: hierarchical LBVH construction for real-time ray tracing of dynamic geometry[C]. Conference on High Performance Graphics,ACM,2010. June 25-27.
[17] 杨鑫,王天明,许端清.基于 GPU 的层次包围盒快速构造方法[J].浙江大学学报(工学版),2012,46(1):84-89.

Research and Application of Large-scale 3D Model Accelerated Rendering Technology

LIU Yiming, HE Xiaoxi, HUANG Shixian
(College of Software Engineering,Chengdu University of Information Technology,Chengdu 610225,China)

Abstract: With the increasing requirements for large-scale scenes in the field of games and simulation and in order to ensure that the general computer can still run at a higher frame rate in the scene of a large-scale three-dimensional model, we propose a scene traversal algorithm based on Morton Code. It is based on the idea of full quadtree node encoding for improving the Morton Code of linear quadtree, and a linear multi-segment structure is proposed. It is also combined with bottom-up BVH to reduce traversal nodes, and finally the frustum is used to quickly crop the objects in the scene. This algorithm combines the multi-layer uniform grid in the scene, uses the CPU's multi-core parallel computing capabilities, and quickly obtains the lower-layer input according to the upper-layer traversal results. It can quickly cut the visibility of the objects in the scene and improve the computer's large-scale model scene rendering frame rate. Experimental results show the correctness and feasibility of the algorithm. Compared with the traditional acceleration structure, its frame rate has been significantly improved.

Keywords: large-scale scene; acceleration structure; Morton code; quadtree; BVH