

# 《计算机图形学》读书报告 —— 渲染技术中的速度优化调研

11821095 葛林林

2019年2月16日

## 1 引言

场景渲染又称为图像合成技术是一种利用2D或者3D模型自动生成具有真实感或非真实感图片的过程。虚拟场景的要素有几何信息、观测视角、物体纹理信息、光照信息、阴影信息等，在渲染技术中最核心的技术是光线跟踪技术。然而由于每根光线都需要对其他所有物体进行检测和进行求交运算，因此光线跟踪技术是非常耗时的。常用的几种速度优化方案有：（1）使用更快的机器；（2）使用并行处理的方法；（3）使用更为有效的算法（例如分割物体或者分割空间）；（4）减少光束个数。其中应用最为广泛的是构建加速结构对光线的求交进行优化，评价加速结构角度有结构创建的速度、使用的内存大小、遍历结构的速度等。本读书报告对不同的渲染技术优化方案进行了调研。

## 2 基于GPU的并行加速方案

### 2.1 Lauterbach等人的GPU并行加速方案

#### 2.1.1 基本原理

2009年C. Lauterbach和M. Garland等人在线性层次包围盒的基础上针对GPU并行功能提出了基于启发式表面区域（SAH）的方案<sup>[1]</sup>。在该方法中，使用多个内核来并行运行多个节点拆分尽可能使用数据并行计算单元在处理器中加速SAH评估以及实际的拆分操作。

#### （1）构建工作队列

在自上而下的构造过程中，由于每一个拆分操作都是完全独立的，将所有的拆分操作在多个并行核中执行是一种最简单的方式来加快速度。在这个过程中需要引入一个队列来管理拆分操作。然而GPU当前的架构并不支持实现全局工作队列。因此在该文中，提出了使用迭代的方法来并行处理所有当前工作状态的拆分并记录新的拆分操作，之后在其间对队列进行维护，从而给下一步提供一个可用的队列。如下图所示是工作队列的构建步骤示意图：

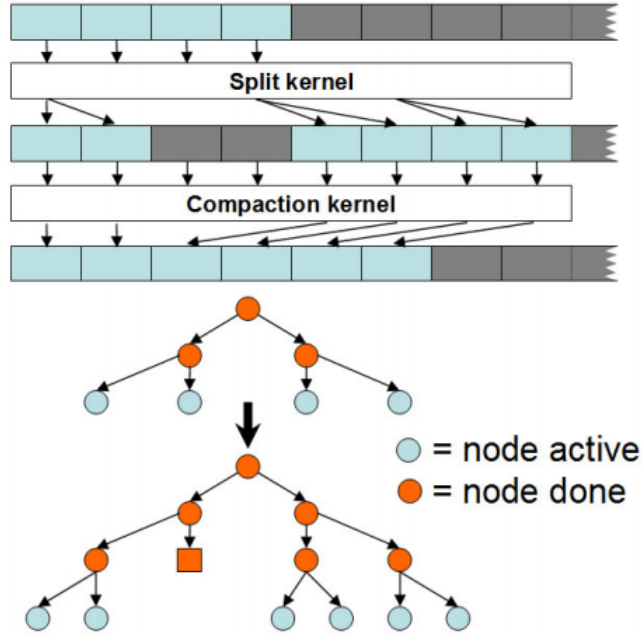


Figure 1: 工作队列的构建，通过使用两个工作队列，可以并行运行所有激活的拆分操作。由于拆分内核可能不会输出新的拆分，因此每次拆分步骤之后会进行压缩操作来删除队列中的空白空间。

## (2) 数据并行处理

为对象层次结构执行SAH拆分操作包括两个步骤：

- 通过评估SAH确定最佳分割位置；
- 基元重排序，与Kd-tree等空间分区方法不同，重排序不需要额外的内存。

上述两个步骤都可以通过利用数据并行性和使用常见的并行操作来执行，并且每个拆分操作都在一个核上执行。

## (3) 细小拆分操作的优化

针对由于层次结构的顶层初始拆分阶段缺乏并行性以及到最后存在很多的细小拆分导致的算法速度瓶颈问题。本文中使用每个处理器的本地内存来维护所有的本地工作队列。由于cache非常快，从而较少了内存访问等待时间。除此之外，内核中使用尽可能少的线程来最大化利用向量运算。分割为“小”的基元的阈值取决于可以将多少几何数据存储到本地存储器中。

## 2.2 Karras和Apetrei的并行优化

### 2.2.1 基本原理

2012年Tero Karras在基数树的基础上提出了一种LBVHs的改进算法，从而减少了LBVHs在构建时的耗时<sup>[2]</sup>。

Karras在其文中指出传统的构建包围盒和八叉树是基于在空间填充曲面内对原始数据进行排序，该方法的缺点无法并行的进行。该文中方法整体流程如下所示：

(1) 给每个原始数据根据其质心进行莫顿码赋值；

(2) 对莫顿码进行排序；

(3) 构建二叉基数树；

传统的构建基数树的方式是一种串行的方式。为了使得构建的过程并行化，Karras为内部节点分配了索引，并且利用一个特定的树布局来建立节点索引和关键值之间的关系，使其能够在不依赖于早期结果的情况下查找其子节点。文中提到的用来帮助实现并行化的二叉基数树示意图如下图所示，树中每个内部节点都被分配了一个介于0-6之间的索引，并与相同索引的叶节点水平对齐。由每个节点覆盖的键的范围由水平条指示，并且根据键值的第一个比特位的不同，将位置进行区分开来由红色圆圈表示。

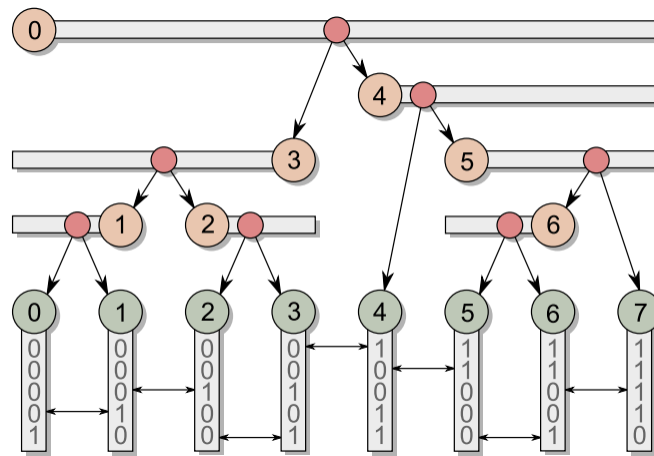


Figure 2: 并行构建二叉基数树示意图

(4) 给二叉基数树的内部节点赋予包围盒；

2014年Apetrei有在Karras方法上做了进一步改进<sup>[3]</sup>。相比于原始方法在排序阶段有了将近50%的速度提升。Apetrei从两个方面改进了Karras的方法。第一方面是将层次结构和包围盒产生放在同一个cuda的kernel中来完成。与Karras不同的是Apetrei采用的是自下而上的构建顺序，既从每一个键出发，把这些键一层一层的归并到某个簇中，直到顶层节点包含了所有的键。第二方面是将内部索引关系和涵盖的键范围建立步骤相合并，通过跟踪键的范围作为自下而上减少的一部分，并使用它们推导每个步骤的父节点的索引。每个叶节点都覆盖一个键的范围，每个内部节点合并其子节点的范围。

## 3 基于深度控制的优化方案

### 3.1 基本原理

1983年Hall, Roy A.和Donald P. Greenberg<sup>[4]</sup>等人提出了一种自适应深度的方法来减少光束的数量。他们将光线跟踪的分为四个步骤：（1）给物理环境建模；（2）对光线在环境中的传播进行仿真；（3）确定成像平面的密度函数；（4）将离散成像平面密度信息转化为可显示的形式。

传统上，任何采样点的交叉树被构造成预先指定的任意深度以确保能够捕获足够的与产生的图片相关的反射和折射信息。然而对于大多数典型环境，由反射或者透明表面组成的场景占比很少。因此有很大的空间去优化树的深度。

该文中将交点树中任何节点的贡献值的上界近似为最终采样点的颜色值，通过考虑树种每个节点的强度来使得强度达到最大。第一个节点在树上将贡献100%的最终颜色样本点。子节点的最大贡献值可以通过将代表反射孩子节点设置统一为 $I_r$ 和 $F_r$ ，将传输的孩子节点设置统一为 $I_t$ 和 $F_t$ ，最终算出相关的贡献率。可以对树中任何父节点的子节点重复此过程。通过乘以近似最大贡献，可以计算子节点对样本点的最大累积贡献。因此，通过建立截止贡献阈值，从而可以在光线跟踪过程中自适应地控制树的深度。

## 4 基于层次包围盒（BVH）的改进方法

我们将对象组包含在分层边界体积集中，并首先测试与边界体积的交集，然后仅在交叉点的情况下对着体积所包围的对象进行测试。

边界体积应易于测试交叉点，例如球体或盒子（平板）。最佳边界体积将由底层对象的形状决定。例如，如果对象是长而薄的，那么球体将主要包围空白空间并且盒子要好得多。对于分层边界卷，框也更容易。使用这样的分层系统（假设它已经仔细完成）会将交叉点计算时间从对象数量的线性依赖性改变为线性和逻辑关系之间的某种程度。这是因为，对于一个完美的情况，每个求交测试会将可能性划分为两个，并且我们将具有二叉树类型结构。

### 4.1 基本概念

#### 4.1.1 BVH

BVH英文全称Bounding Volume Hierarchies，边界体积层次结构（BVH）已广泛存在用于光线跟踪和碰撞检测。在交互式光线跟踪领域中，基于对象层次结构的方法由于Kd-Tree技术在CPU和GPU上得到了发展而重新获得关注。早期的方法使用简单的更新技术来维护动态数据集并且使用快速构建技术来重建BVH<sup>[1]</sup>。

#### 4.1.2 LBVH

2009年Lauterbach等人引入了BVH构造算法，该算法基于在场景边界框内运行的填充莫顿曲线对基元进行排序。空间填充曲线长期以来一直用于改进空间算法。可以计算沿着 $n$ 阶莫尔顿曲线的三维点的坐标，也称为其莫顿码，将其坐标离散化为 $n$ 位并交织它们的二进制数，从而获得 $3n$ 位索引。由于算法将构建层

次结构的问题转化为沿曲线排序一组点的问题，Lauterbach等人将其称为线性边界体积层次（LBVH）算法。该方法的构建步骤如下所示：

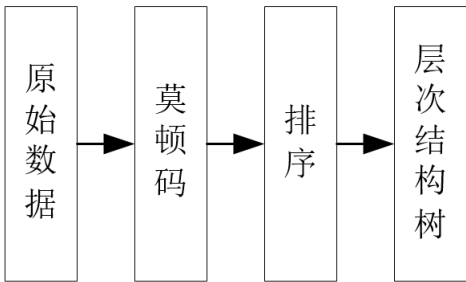


Figure 3: LBVH构建步骤示意图

4.1.3 表面积启发式算法（SAH）

SAH英文全称Surface Area Heuristic，中文称为表面积启发式算法，表面积启发式算法法基于的理论是复杂度成本分析和概率论，主要用于解决怎样划分轴的问题，可以被应用到适用于多种类型的分层加速结构。

4.1.4 莫顿码（Morton Code）

莫顿码可以将多维数据转化为一维数据编码，根据一维编码位数可确定多维数据保留精度，是一种比较常用的压缩编码方法，尤其是作为哈希表的映射算法等，加速了树结构数据的存储和访问速度。莫顿码引入空间填充曲线，它提供了一种稳定的量化矢量的稳定排序方法，这意味着具有相同子序列莫顿码的矢量在空间上是彼此接近。虽然有其他形式的空间填充曲线，但是莫顿码计算的简单性使得它得到了广泛的运用。莫顿码可以通过几次简单的比特交叉操作得到。如下图所示是二维莫顿码的简单示例：

	00	01	10	11
00	0000	0001	0100	0101
01	0010	0011	0110	0111
10	1000	1001	1100	1101
11	1010	1011	1110	1111

Figure 4: 二维4比特莫顿码示意图

	x:	0	1	2	3	4	5	6	7
		000	001	010	011	100	101	110	111
y: 0	000	000000	000001	000100	000101	010000	010001	010100	010101
1	001	000010	000011	000110	000111	010010	010011	010110	010111
2	010	001000	001001	001100	001101	011000	011001	011100	011101
3	011	001010	001011	001110	001111	011010	011011	011110	011111
4	100	100000	100001	100100	100101	110000	110001	110100	110101
5	101	100010	100011	100110	100111	110010	110011	110110	110111
6	110	101000	101001	101100	101101	111000	111001	111100	111101
7	111	101010	101011	101110	101111	111010	111011	111110	111111

Figure 5: 二维6bit莫顿码示意图

莫顿码的一个主要参数是该码的位数。莫顿码的产生步骤如下：

- (1) **确认量化坐标:**例如对应三维的莫顿码其量化坐标为 $\mathbf{v}^* = \{v_x^*, v_y^*, v_z^*\}$
- (2) **交叉操作:**例如对应的三维莫顿码，假设量化坐标 $v_x^* = x_7x_6x_5x_4x_3x_2x_1x_0$ ， $v_y^* = y_7y_6y_5y_4y_3y_2y_1y_0$ ， $v_z^* = z_7z_6z_5z_4z_3z_2z_1z_0$ ，利用交叉操作得到的莫顿码为

$$m(v^*) = x_7y_7z_7x_6y_6z_6x_5y_5z_5x_4y_4z_4x_3y_3z_3x_2y_2z_2x_1y_1z_1x_0y_0z_0$$

查表法可用于预先计算某些位范围的代码，然后在运行时通过简单的添加和组合进行组合。

## 4.2 Hank Weghorst等人改进方案

1987年Hank Weghorst等人<sup>[5]</sup>指出传统的光线跟踪算法是从视角出发反向回到环境中，其具体的步骤如下：

- (1) 首先求解光线与最近表面的交点，之后会得到反射和折射光线；
- (2) 接下来进一步递归的产生反射折射光线，从而得到所有的相交面；
- (3) 在递归求解交点的过程中，一个与采样点对应的由交点构成的树形结构会被建立起来；
- (4) 最终光照密度通过累加树中每个节点的光密度与贡献值的乘积求得。

该文中从三个角度对光线跟踪算法的进行改进。第一个角度是选择合理的包围盒；第二个角度是对层次结构的建立过程进行改进；第三个角度是利用对象存在的连续性提出了一种可见表面预处理方法。

### 4.2.1 包围盒的选取

该文中指出在包围盒的选取过程中需要考虑的因素有面和基平行、空白投影区域、包围物体的复杂性。空白区域是指包围盒与物体投影区域的差值，如下图所示：

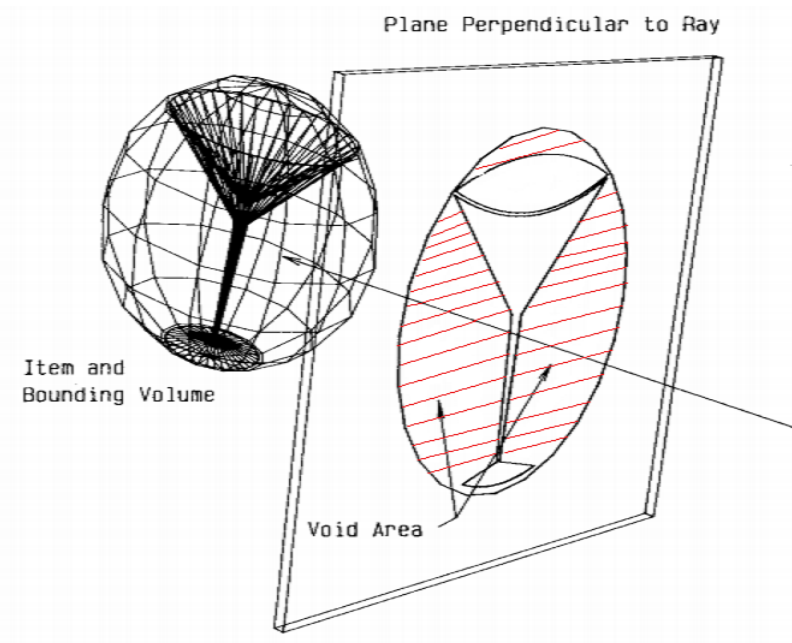


Figure 6: 投影空白区域示意图（红色阴影线部分）

则选取包围盒时，应该使得空白面积最小化。综合上述因素，则总的耗时函数可以表示如下：

$$T = bB + iI$$

其中,  $T$ 为总的耗时,  $b$ 是交点检测中包围盒被检测的次数,  $B$ 单个包围盒检测交点的耗时,  $i$ 物体交点检测的次数,  $I$ 物体交点检测的耗时。当物体指定是 $b$ 和 $I$ 是一个常值, 通过改变包围盒的大小和形状,  $B$ 和 $i$ 可以被改变从而减少整体的耗时。

综合考虑上述因素之后, 选取物体的包围盒首先需要确定包围该物体的一个点集, 该点集组成的多面体是该物体的凸包, 其中最优的凸包顶点集合应该满足的条件是是让多面体与物体的体积之差最小。因此该文中提出了一个包围盒选取的方案, 其选取步骤如下所示:

- (1) 给定一组包围盒的候选集, 例如球体、长方体、圆柱体;
- (2) 给包围盒候选集中的所有元素指定一个因子, 该因子与交点检测的复杂度相关, 例如球的该因子最小, 而圆柱体的该因子最大;
- (3) 包围盒选取的结果为使得选定的元素对应的因子与物体对应的潜在边界形状体积的乘积。

#### 4.2.2 层次结构的建立改进

传统方法中, 层次结构中引入了组的概念。距离比较相近的物体可以被归并到同一个组, 而同一组又可以用一个包围盒来表示。而这个包围组的包围盒相对于物体的包围盒更高一层。

本文中提出为了简化过程, 可以将组内只有一个元素的组从层次结构中剔除。这样的简化过程得到的结果依然能够满足用户的要求。

#### 4.2.3 基于可见表面的预处理方法

由于可见表面必然与观察者最为接近, 该文中借鉴了Z-buffer算法的方法产生Item Buffer, 从而对物体的前后顺序得到一种索引列表, 随后光线跟踪算法便可以根据该索引列表得到第一个光线交点, 从而改进光线跟踪中求交的速度, Weghorst建议使用修改的Z缓冲算法来确定第一次命中。

### 4.3 基于HLBVH的方法

2010年J. Pantaleoni和D. Luebke提出了HLBVH方法<sup>[9]</sup>。HLBVH英文全称Hierarchical Linear Bounding Volume Hierarchies。涉及到的主要技术有莫顿码构建线性层次包围盒、表面积启发式算法、hierarchy emission algorithm。

- (1) 首先利用莫顿码给场景中的规则化网格进行编码;
- (2) 将LBVH算法分成两层结构, 在第一层中根据 $m$ 位的莫顿曲线将基元排序为 $3m$ 位的网格, 在第二层对剩下的3个比特位对基元进行排序;
- (3) 为每个基元计算和存储30位的莫顿码。
- (4) 使用层次排放算法, 该算法的核会找出其中比较重要的比特位, 其步骤如下所示:



- 首先，原始数组根据每一个叶子节点在当前输出结构中的位置从概念上将初始数组分割为几个组。
- 根据这些组产生一个深度为 $p$ 小树结构，这些小树结构可以在列表中完整描述每个段中的分裂平面。

## 5 总结

本文针对渲染中常用的加速技术进行了调研，主要涉及了三个方面的加速方案并行加速方案、基于BVH的加速方案、减少光线个数的方案。如下表所示是对上述多种方案的结果总结：

Table 1: 各种方法的耗时数量级总结

方法分类	年代	方法名称（或作者）	耗时数量级
基于GPU并行方案	2009年	Lauterbach等人	毫秒级
	2012年和2014年	Karras和Apetrei	毫秒级
深度控制方案	1983年	Hall等人	分钟级
基于BVH的优化方案	1987年	Weghorst等人	小时级
	2010年	HLBVH（同时使用了GPU并行）	毫秒级

通过上表可以发现，并行方案对于耗时的优化效果较好，并且由于BVH的方案能够与并行方案相结合，因此近年来基于BVH的优化方案研究较多。

## References

- [1] Lauterbach, Christian, et al. "Fast BVH construction on GPUs." Computer Graphics Forum. Vol. 28. No. 2. Oxford, UK: Blackwell Publishing Ltd, 2009.
- [2] Karras, Tero. "Maximizing parallelism in the construction of BVHs, octrees, and k-d trees." Proceedings of the Fourth ACM SIGGRAPH/Eurographics conference on High-Performance Graphics. Eurographics Association, 2012.
- [3] Apetrei, Ciprian. "Fast and simple agglomerative LBBVH construction." (2014).
- [4] Hall, Roy A., and Donald P. Greenberg. "A testbed for realistic image synthesis." IEEE Computer Graphics and Applications 3.8 (1983): 10-20.
- [5] Weghorst, Hank, Gary Hooper, and Donald P. Greenberg. "Improved computational methods for ray tracing." ACM Transactions on Graphics (TOG) 3.1 (1984): 52-69.
- [6] Woop, Sven, Attila T. Áfra, and Carsten Benthin. "STBVH: a spatial-temporal BVH for efficient multi-segment motion blur." Proceedings of High Performance Graphics. ACM, 2017.

- [7] Vinkler, Marek, Jiri Bittner, and Vlastimil Havran. "Extended Morton codes for high performance bounding volume hierarchy construction." *Proceedings of High Performance Graphics*. ACM, 2017.
- [8] Havran, Vlastimil, Robert Herzog, and Hans-Peter Seidel. "On the fast construction of spatial hierarchies for ray tracing." *Interactive Ray Tracing 2006, IEEE Symposium on*. IEEE, 2006.
- [9] Pantaleoni, Jacopo, and David Luebke. "HLBVH: hierarchical LBVH construction for real-time ray tracing of dynamic geometry." *Proceedings of the Conference on High Performance Graphics*. Eurographics Association, 2010.
- [10] Woop, Sven, Attila T. Áfra, and Carsten Benthin. "STBVH: a spatial-temporal BVH for efficient multi-segment motion blur." *Proceedings of High Performance Graphics*. ACM, 2017.
- [11] Benthin, Carsten, et al. "Improved two-level BVHs using partial re-braiding." *Proceedings of High Performance Graphics*. ACM, 2017.