基于密度估计与聚类的混合空间加速结构

郭昀辉,何晓曦,胡 梁,任 和,陈锦伟

(成都信息工程大学 软件工程学院,四川 成都 610225)

摘 要: 为解决在密度分布不均的复杂体素场景中构建的空间加速结构时间长、质量较差等问题,提出一种基于密度估计与聚类的混合空间加速结构。首先,在传统边界包围盒层次结构的基础上,通过改进的核密度估计方法为离散的体素数据构造连续的密度环境;其次,根据核密度估计函数的一、二阶偏导方程和聚类结果,以确定体素空间中的高密度区域;最后,提出一种自适应跳跃洪泛算法在这些区域中构建相应的有向距离场,作为加速结构树的叶子节点,计算过程均由GPU实现以满足复杂场景的大规模并行计算需求。结果表明,所提算法在体素密度分布不均的复杂场景中,构建的空间加速结构树时间较短、搜索效率更优,可有效提升光线追踪的渲染速度。

关键词: 空间加速结构;核密度估计;基于密度的聚类;有向距离场;光线追踪

DOI: 10. 11907/rjdk. 241773 **开放科学(资源服务)标识码(OSID):**

中图分类号: TP18 **文献标识码:** A **文章编号:** 1672-7800(2025)002-0121-08

Hybrid Spatial Acceleration Structure Based on Density Estimation and Clustering

GUO Yunhui, HE Xiaoxi, HU Liang, REN He, CHEN Jinwei

(*College of Software Engineering, Chengdu University of Information Technology, Chengdu* 610225, *China*)

Abstract: To solve the problem of long construction time and poor quality of spatially accelerated structures constructed in complex voxel scenarios with uneven density distribution, a hybrid spatially accelerated structure based on density estimation and clustering is proposed. Based on the traditional bounding volume hierarchy, a continuous density environment is constructed for discrete voxel data through an improved kernel density estimation method. The high-density regions in the voxel space are determined based on the first- and second-order partial derivative equations of the kernel density estimation function and the clustering results. An adaptive jump flooding algorithm is proposed in this paper to construct the corresponding signed distance fields in these regions as leaf nodes of the accelerated structured tree. All computational processes are handed over to GPUs for implementation, thus meeting the demand for massively parallel computation in complex scenarios. The results show that the construction time of the spatially accelerated structure tree supported by this algorithm is significantly shortened and the search efficiency is improved in complex scenes with uneven voxel density distribution, which implies that this method can effectively improve the rendering speed of ray tracing.

Key Words: spatial acceleration structure; kernel density estimation; density-based clustering; signed distance fields; ray tracing

0 引言

光线追踪是一种基于物理光学原理的图形渲染技术,由于准确模拟了光线在真实世界中的传播过程(折射、反射、阴影等),因此生成的图像更逼真,更接近人们所追求的高质量目标。光线追踪的局限性在于需要追踪屏幕中每一个像素点发出的射线,这些射线经过各种可能的物理过程直至达到中止条件,因此计算成本十分巨大,关键点还是要找到对应光线与场景之间的交集。在实际计算过程中,可能需要对数十亿条光线与场景中数百万个图形基元(三角形或体素点)进行相交测试,因此必须使用一些加速结构来组织和管理三维场景,以提升计算效率。

目前,常见的空间加速结构包括八叉树、KD树、包围盒层次结构(Bounding Volume Hierarchy, BVH)等。其中,BVH使用最为广泛,相较于其他空间加速结构具有以下优

收稿日期: 2024-09-09

扫描二维码阅读全文:

作者简介: 郭昀辉(1999-),男,成都信息工程大学软件工程学院硕士研究生,研究方向为图形图像与可视化、混合现实与虚拟仿真;何晓曦(1978-),男,博士,成都信息工程大学软件工程学院副教授,研究方向为图形图像与可视化、混合现实与虚拟仿真、数字孪生与数据可视化。本文通讯作者:何晓曦。

缺点：①构建和遍历相对简单直观，易于实现和理解；②适用性广，适用场景包括动态、静态场景及各种形状大小的物体，在体素场景中也有出色的表现；③节点只需存储边界信息和子节点指针，相较于其他结构内存的占用较少，但在某些情况下可能会导致构建的树不平衡，使得某些节点的深度过深，从而影响渲染性能；④对于大型场景而言，BVH 构建可能需要较多的时间和计算资源，特别在动态场景中需要频繁更新结构；⑤对于图形基元（三角形或体素点）密度较高的场景，BVH 可能会导致树的层级较深，从而影响搜索效率。

为了解决上述问题，很多学者对 BVH 的构造算法进行了改进和优化。目前，BVH 构建方法大致可分为自顶向下、自下而上和增量构造 3 类。其中，结合表面启发式（Surface Area Heuristic，SAH）的自顶向下的构建方法可生成高质量的 BVH，但要计算所有可能的子节点面积，因此计算代价较大[1]。在此基础上，Lauterbach 等[2]通过空间莫顿编码的线性排序加快构建速度。

此外，BVH 的构造就是空间分割三维场景中的图形基元，实际上是一个层次聚类的问题。为此，Meister 等[3]将 K-Means 的聚类方法引入 BVH 的构建过程中，并将整个过程在 GPU 上进行实现，结果表明该模型的构建速度与质量水平相当高。本文针对体素密度分布不均的三维场景，使用核密度估计（Kernel Density Estimation，KDE）和基于密度的聚类（Density-Based Spatial Clustering of Applications with Noise，DBSCAN）划分高密度区域和低密度区域，提出了一种基于密度估计和聚类的混合空间加速结构。在高密度区域中，由于 BVH 可能会导致树的层次过深，因此使用局部的有向距离场（Signed Distance Field，SDF）代替 BVH，作为空间加速结构树的子节点；在低密度区域依然使用结合 SAH 的 BVH 结构。同时，本文提出一种自适应跳跃洪泛算法（Adaptive Jump Flood Algorithm，AJFA），既加快了 SDF 构建的收敛速度，又提升了模型准确度。

1 相关工作

1.1 BVH 的构建

BVH 作为一种三维场景中组织和管理图形基元的有效方式，在碰撞检测与光线追踪中应用广泛，主要以二叉树为基本的数据结构分层组织场景，如图 1 所示。其中，根节点包含整个场景的边界；内部节点包含其所有子节点的边界；叶子节点包含图形基元；A、B、C、D、E 代表非图形基元节点。

对于 BVH 的构建方法而言，最经典的是结合 SAH 的自顶向下的构造方法，旨在最小化每一步的分割成本。Macdonald 等[1]认为包围盒的表面积与射线相交的概率具有紧密的联系。另一类众所周知的方法是从叶子节点（即图形基元）开始，通过迭代合并节点的方式自下而上进行

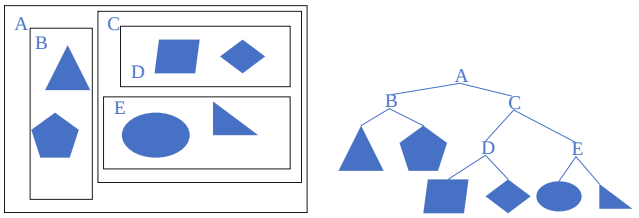


Fig. 1 An example of BVH in a 2D scene

图 1 一个在二维场景中 BVH 的示例

构造，尽管这种方式能以非常低的 SAH 成本生成树，但在实际的光线追踪测试中性能一般不及自顶向下的方法[4]。

目前，BVH 构建方法的优化和改进很多，包括分箱的 SAH 与线性 BVH（LBVH）。分箱的 SAH 将每个节点可能包含的分割平面限制为固定数量，而非考虑所有可能的情况；进一步的优化方法是使用固定大小的网格在整个场景中对图形基元进行分组操作，并统一管理给定网格单元中重叠的节点[5,6]。线性 BVH 首先沿着一个空间填充曲线（空间莫顿编码等）对图形基元进行排序，然后将他们递归组合，使每个节点都代表一个线性图形基元的范围[2,7,8]。Karras 在测试中证明，线性 BVH 的构建在整个空间结构树上完全可以实现并行化操作，虽然 LBVH 的构造速度很快，但其光线追踪性仍然不够强[9]。为此，部分学者又提出了混合算法，在空间重要区域使用高质量的构建算法，而在构造代价较大的区域使用快速构建算法。

自底向上的构建方法也是构造 BVH 的常用方法之一，其基本思想是从图形基元出发，依据判定条件寻找相邻的节点进行合并从而构建新的 BVH 节点。在进行上述操作时一般需要引入聚类过程，例如 Walter 等[4]基于聚集聚类的方法自底向上构建 BVH，使用的距离函数是紧密包围两个相邻图形基元的包围盒表面积，虽然 BVH 总成本较低但构建时间长。Meister 等[3]在 BVH 的构建中融合 K-Means 方法，并将 GPU 作为计算平台以有效缩短计算时间。为了进一步缩短构建时间，使计算过程更好地适应现代 GPU 框架，Walter 等[10]提出一种并行局部有序聚类（PLOC）的方法，虽然提升了构建效率但存储信息大，内存占用代价高。Carsten 等[11]在此基础上提出并行局部有序聚类的进阶版（PLOC++），侧重于通过简化和融合不同阶段来解决 PLOC 存在的一些问题。此外，Chitalu 等[12]开发了新的 BVH 在内存中的表示方法，基于简单位移的隐式索引表示紧凑内存位置的映射，旨在快速构建用于宽相位碰撞检测的 BVH。

BVH 的构建过程中，包围盒结构一般采用轴对齐包围盒（Axis-Aligned Bounding Box，AABB），但在毛发、数量较大的丝状物等特殊场景中，图形基元呈长、薄、弯曲等特质时，相邻图形基元 AABB 之间的重叠可能较大，渲染成本很高。为此，Wald 等[13]使用支持 rtx 硬件、仅在图形基元级别执行的轴向边界包围盒（Oriented Bounding Box，OBB），加速效果相较于 AABB 提升较大。

上述是对BVH构建方法的一些改进，随后不少学者对构造完成后的BVH也进行了优化。例如，大多数优化算法都是针对简化的成本模型，即假定叶节点固定，如此整个BVH树的代价评定标准就变为了内部节点包围盒表面积的总和。Kensler等[14]提出使用树的旋转来提升树的质量，其灵感来自于传统平衡二叉搜索树中的操作。Bittner等[15]建议直接删除子树并将他们插入到新的位置，以提供更大的搜索空间，但选择处理的子树顺序不当会导致更大的开销。Bittner等[16]通过Metropolis-Hastings采样操作对文献[15]的方法进行了改进，提出一个优先队列和分支定界的算法来寻找新位置。

本文方法采用的是混合结构，相较于其他混合结构的不同之处在于：①分类标准是场景中体素点的密度，通过核密度估计与基于密度聚类方法划分体素点的高密度区域与低密度区域；②不仅仅使用了包围盒结构作为节点载体，还在体素高密度区域计算SDF来连接叶子节点。

1.2　SDF构建方法

有向距离场是一种描述物体边界的数据结构，记录了空间中每个位置到距离该位置最近的有效样本单元（三角形面或体素点）。若该位置在有效样本单元之外则距离为正值，若该位置在有效样本单元上则距离为0，若该位置在有效样本单元之内则距离为负值。如此，SDF能有效表示复杂的几何形状，并快速进行体积碰撞检测和物体表面重建等操作。

Rosenfeld等[17]对图像进行倒角距离变换，并提出了距离场的概念。此后，为了提升倒角距离变换的精度，许多学者提出矢量距离变换、快速行进方法等新方法。在三维领域，由于计算三维离散网格上一组对象最短距离的蛮力算法非常简单，仅计算所有对象到该体素的距离并存储最小的一段，但这种方法不切实际，因为计算时间成本难以想象。一种常见的优化算法是通过空间划分结构来加速计算，例如Strain[18]使用四叉树加速二维距离计算。Guezlec[19]基于网格的分层表示提出Meshsweeper算法，在每一层中通过一个包围盒与每一个图形基元相互关联。Koschier等[20]提出一种从多样化多边形输入网格，在轴向六面体网格上生成离散高阶多项式SDF的自适应算法，并将其作为碰撞检测器来模拟基于物理的高度复杂几何体。Wu等[21]提出一种由三角形网格构造SDF的新算法，首先计算三角形网格的内部与外部距离场，其次由这两个距离场直接生成SDF，即只使用点来生成距离域以避免计算点到三角形的复杂计算。除了使用分层结构加速计算之外，还引入了特征方法。此类方法都基于Mauch[22,23]提出的特征概念，将网格中的每个图形基元转换成一个多面体特征，包括了最接近该特征的点。Sigg等[24]在此基础上使用图形硬件来扫描和转换特征。

数学方法也是计算SDF的有效方法，一开始将简单的几何曲面函数作为计算SDF的依据，但无法满足复杂模型的需求。为此，Pujol等[25,26]除了使用复杂的数学条件来满足SDF隐式曲面表示外，提出分段连续插值法自适应逼近有向距离场，使用一种自适应结构，通过值和导数的三线性或三次插值来表示SDF，从而允许快速查询域。Jiang等[27]使用由SDF表示的三维形状的可微渲染方法，用于多视图的三维重建。此外，DeepSDF被提出用于学习一类形状的连续SDF表示[28]。深度水平集作为端到端的可训练模型，可预测任意拓扑的隐式曲面[29]。

由于大规模并行计算的需求，本文在跳跃洪泛算法（Jump Flooding Algorithm，JFA）的基础上对其收敛步长进行自适应判断，提出自适应跳跃洪泛算法（Adaptive Jump Flooding Algorithm，AJFA），在既满足每个体素点计算独立性的同时，又提升了SDF的计算精度[30]。

2　本文方法

2.1　核密度估计

由于需要在体素分布不均的复杂场景中估计出体素密度较高的区域，本文引入了数据分析与概率统计学中的核密度估计。其中，核密度估计（Kernel Density Estimation，KDE）是一种重要的非参数统计方法，用于估计未知区域的概率密度，将每个样本点视为一个中心，使用一个核函数（通常是对称的概率密度函数）度量该样本点对其他样本点的贡献，通过离散样本点数据的线性累加来构建一个连续的概率密度函数，以获得整个样本空间的平滑样本分布，并最终得到所有样本点的密度峰值。

在实现核密度估计的过程中，最重要的是选择核函数与带宽。核函数的要求为非负、积分为1、符合概率密度性质及均值为0，决定了每个样本点对给定样本点贡献的形状；带宽决定了贡献范围。一般情况下，核函数包括高斯核、Epanechnikov核、矩形核、三角核及多项式核函数。在体素场景下，本文选择一个以Epanechnikov核（见式（1））为基础的变体——四次核函数（又称四次多项式核或三次B样条核函数）作为本文核密度估计的核函数，如式（2）所示。

$$K(t)=\frac{3}{4}\left(1-t^2\right)\ \text{for}|t|\leqslant 1 \tag{1}$$

$$K(t)=\frac{15}{16}\left(1-t^2\right)^2\ \text{for}|t|\leqslant 1 \tag{2}$$

式中：t为评估点与数据点间距离与带宽的比值。

式（2）是Epanechnikov核的一种变体，将平方项增加到核函数中使其既能保持有限的支持范围（在$|t|<1$的区域核函数值大于0），由能在t接近±1的时候函数值能更平滑地接近0。

带宽计算公式如式（3）所示。

$$h=0.9*min\left(\sigma,\ \sqrt{\frac{1}{\ln 2}}*D\right)*n^{-0.2} \tag{3}$$

式中：h为带宽长度；σ为样本距离标准差（见式（4））；

D 为所有有效体素点到平均中心距离的中值(平均中心即所有有效体素点对其坐标取均值);n 为有效体素点的数量。

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}\left(\mathrm{dist}\left((x_i, y_i, z_i), (\bar{x}, \bar{y}, \bar{z})\right) - \overline{dist}\right)^2}{n}} \tag{4}$$

式中:n 为有效体素点的数目;dist 为欧氏距离;(x_i, y_i, z_i)为第 i 个有效体素点的坐标;$(\bar{x}, \bar{y}, \bar{z})$为所有有效体素点坐标的均值;$\overline{dist}$ 为所有有效体素点到平均中心距离的平均值。

基于核函数和带宽,某一体素点(x, y, z)的核密度估计如式(5)所示。

$$\begin{aligned}\mu(x, y, z) &= \frac{1}{nh^3}\sum_{i=1}^{n} K(t) \\ &= \frac{1}{nh^3}\sum_{i=1}^{n} K\left(\frac{dist\left((x, y, z), (x_i, y_i, z_i)\right)}{h}\right)\end{aligned} \tag{5}$$

式中:(x, y, z)为需评估的三维点坐标;(x_i, y_i, z_i)为样本数据中第 i 个点的三维坐标;n 为有效体素点总数;h 为带宽;dist 为两点之间的距离(选用欧氏距离);$K(t)$如式(2)所示。

对于一个 256×128×256 大小,以 RGBA16 SFloat 为存储类型的三维体素数据,其核密度估计可视化结果如图 2 所示(彩图扫 OSID 码可见,下同)。由此可见,图中颜色越接近红色说明周围体素密度越高,蓝色点周围体素密度越低,证明了使用核密度估计能筛选出当前体素场景下周围密度最高的有效体素点,为体素空间划分和降维密度聚类奠定了基础。

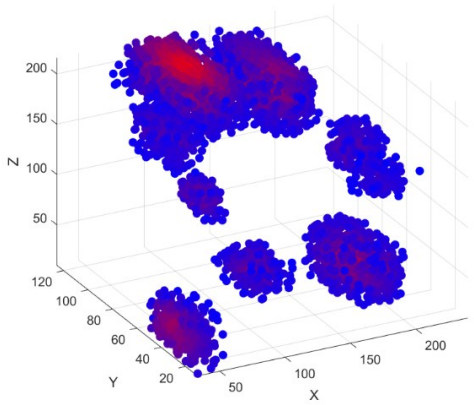


Fig. 2 Visualization of the results of KDE calculations

图 2 KDE 计算结果可视化

2.2 基于密度的聚类

聚类方法旨在将具有相似特征的元素归为一类。Rodriguez 等[31]提出密度峰值聚类(Density Peak Clustering, DPC)的方法,使用局部密度和给定数据点与另一个密度更高的点之间的最小距离构建决策图。同时,在决策图中手动选择密度高且距离其他高密度区域较远的位置,即在体素点分布不均的场景中每次都要找出一个划分位置和对应的划分值,仅需将全局的密度峰值区域与其他区域进行划分,但该算法不擅长识别不同密度的簇。此外,基于密度的聚类方法还包括 DBSCAN 算法、OPTICS 算法、DENCLUE 算法等。

由于在密度分布不均的体素场景中无法预先指定聚类簇的数量,不少算法存在识别任意形状的簇及处理数据维度不高等情况。为此,本文选择基于密度的聚类算法,其优点在于适用于任意形状的簇,能自动检测噪声点且无需预先指定簇的数量,虽然在处理高纬数据时的效率较低,但可通过数据降维来解决这个问题。DBSCAN 通过一个核心点 eps(ε)的邻域半径与一个数据点 minPts 成为核心点所需的最少邻居数(包括自己)来定义簇。该算法通过访问所有数据点进行核心点判断、扩展簇、递归扩展、标记噪声来实现整个样本点空间的聚类。实验表明,在数据降维且参数选择合适的情况下,该算法可聚类处理所有有效体素点,为体素空间划分标准提供依据。

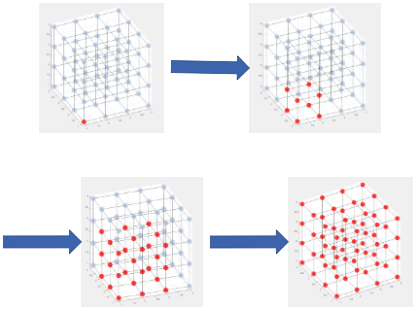
2.3 基于 AJFA 的 SDF 构建

在体素密度分布较大的区域,由于有效体素点之间的距离较小甚至相邻,在光线与有效体素相交计算时,光线在 SDF 中的安全步进距离较短,能快速跳过空体素点,提升反馈效率。

普通的洪泛算法(二维情况下)是在一次计算中,固定地向相邻的一个像素点传播信息。目前,Rong[30]在此基础上提了一种计算 Voronoi 图和距离变换的算法 JFA。为了满足大规模计算的需求,本文将 GPU 作为计算单元,在跳跃洪泛算法的基础上自适应调整步长,提出了自适应跳跃洪泛算法,即按照 2 的幂次递增或递减来传递信息,使构建过程既能满足收敛速度,又能够保证一定的精确程度。这种思想在三维领域同样适用,算法会按照所选跳跃步长进行搜索,递归过程中步长逐渐收敛直至 1(即相邻位置)代表完成了 SDF 的计算,如图 3 所示(红色点为有效信息点)。

由此可见,跳跃洪泛算法解决体素点 SDF 构建的过程为:首先在确定区域中对每个非有效体素点,向 26 个方向根据设定的步长行进,如果该位置就是某一个有效体素点的位置,则将该位置的坐标和两个体素点之间的距离(欧氏距离)记录到该非有效体素点的信息中;若该位置也是个非有效体素点,则访问该点信息,并比较其存储的有效体素点位置到需要计算的非体素点距离(更新步长)与需要计算的非有效体素点存储的距离信息,选择较小的一段距离记为需要计算的非体素点信息,如图 4 所示(红色圆代表的是有效体素点。)。

一般而言,步长的初始值选择扩散空间最长为边长的一半,这样能保证在第一步计算中就可以覆盖一半以上的有效样本,步长的衰减过程通常每次选取当前步长的一半作为衰减值,直至收敛至 1。然而,这样的收敛过程并非对



（a）　Flooding with double step jumps

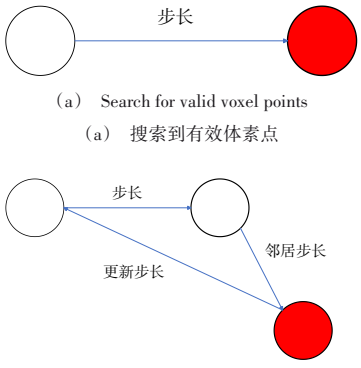
（a）　以两倍步长跳跃方式进行洪泛

（b）　Flooding with half the jumps in side lengths

（b）　以边长一半跳跃方式进行洪泛

Fig. 3　Two methods for selecting the propagation step size of information points in 3D voxels

图 3　在三维体素中两种信息点传播步长选择的方式



（a）　Search for valid voxel points

（a）　搜索到有效体素点

（b）　Search for non-valid voxel points

（b）　搜索到非有效体素点

Fig. 4　Different search scenarios

图 4　不同搜索情况

所有场景适用，可能会使算法在有效样本点密集的区域收敛速度过快，导致该区域 SDF 计算结果精度较差，或在有效样本点稀疏的区域收敛速度过慢，从而增加递归计算量。值得注意的是，样本点的密集程度是针对当前区域而言，本文空间结构已根据有效体素点的密度进行了划分。

为此，本文提出自适应步长的方法以解决上述问题，即不再以固定的衰减值（如上次步长的一半）进行收敛操作，并在每次洪泛过程后对当前洪泛计算结果进行评价。如果评价结果较好，说明该步长范围内有效样本点密度高，则将下次步长衰减值减小以提升该区域的 SDF 精度；如果评价结果较差，则说明该区域有效样本点密集程度低，可将下次衰减值增大来加快收敛速度。具体参数包括：①在此次洪泛跳跃计算中获得距离更新点数量占总搜索数量的比例（见式（6））；②平均更新比例，即在此次洪泛跳跃计算中获得距离更新点与原距离比值的平均值（见式（7））。

$$E_1 = \frac{N_{(m)}}{k} \tag{6}$$

$$E_2 = \frac{\sum_{i=1}^{k} \frac{d_{mi}}{d_{(m-1)i}}}{N_{(m)}} \tag{7}$$

式中：k 为第 m 次洪泛计算中搜索点的总数；$N_{(m)}$ 为第 m 次洪泛计算中获得距离更新点数量；d_{mi} 为第 m 次洪泛计算中第 i 个体素点获得更新的距离；$d_{(m-1)i}$ 为第 $m-1$ 次洪泛计算中第 i 个体素点信息存储的距离。

本文对同一个体素点数据集使用 JFA 和 AJFA 算法进行收敛，结果如图 5 所示。由此可见，AJFA 算法（红色折线）在第 6 次收敛时达到最终收敛目标，收敛速度更快。同时，在步长为 64~48 和 10~15 的情况时收敛速度放缓，SDF 的计算精度得了显著提升。

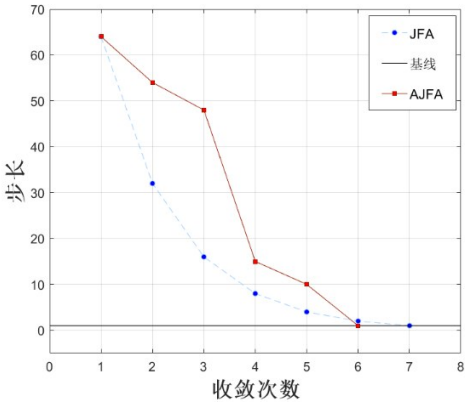


Fig. 5　Convergence results of JFA and AJFA algorithms

图 5　JFA 与 AJFA 算法收敛结果

在 SDF 的构建过程中，本文使用 RGBA 通道的颜色信息作为存储 SDF 信息的存储结构，即对于每个体素点而言，通过 (r, g, b) 存储距离该体素点最近的有效体素点的位置坐标 (x, y, z)，A 通道用于存储该体素点到距离该体素点最近的有效体素点的欧式距离，结果如图 6 所示。其中，左侧为体素源数据，右侧为使用自适应跳跃洪泛算法计算的 SDF 体素数据。由此可见，结合自适应步长的跳跃洪泛算法在保证精确程度的情况下提升了收敛速度，为加速光线与体素场景的相交测试提供了局部优势。

Fig. 6　Construction process of SDF

图 6　SDF 的构建过程

3　混合结构树的构建与搜索

混合结构树的构建分为空间划分和节点构建。其中，

空间划分是前置工作，为节点构建提供了判断依据和参数基础；节点构建是最终结果，对具体的节点内容进行计算和丰富，为体素空间下加速光线与体素点相交测试提供了有利条件。混合结构树的搜索是混合结构树构建的最终目的，为加速光线追踪等操作提供了方法说明和实现途径。

3.1 空间分割

空间划分过程可分为3个具体步骤：首先使用核密度估计方法计算密度峰值为基准的标准差，判断是否需要进行空间分割；其次由核密度估计函数确定划分轴；最后根据基于密度的聚类结果确定划分值。

实际过程中，根据KDE计算密度峰值点对当前节点空间进行一次密度评价，即计算该节点空间内以密度峰值点为参考的体素分布情况，如式(8)所示。如果3个轴向上的标准差之间的差值均小于阈值，则当前区域范围内的体素密度分布均匀，应当根据区域范围构建SDF节点。如果3个轴向上的标准差之间的差值大于阈值，则当前区域范围内虽然包含了体素密度的峰值点但体素密度分布不均，不适合建立SDF节点，而应当作为BVH的中间节点，因此使用者可设定阈值来调整混合结构树的深度。

$$\sigma(x,y,z)=\begin{pmatrix}\sqrt{\dfrac{\sum_{i=1}^{n}(x_i-x_{max})^2}{n}},\\ \sqrt{\dfrac{\sum_{i=1}^{n}(y_i-y_{max})^2}{n}},\\ \sqrt{\dfrac{\sum_{i=1}^{n}(z_i-z_{max})^2}{n}}\end{pmatrix} \tag{8}$$

式中：$\sigma(x,y,z)$为3个轴向上标准差计算结果；(x_i,y_i,z_i)为当前区域范围内第i个体素点的坐标；$(x_{max},y_{max},z_{max})$为体素密度峰值点的坐标。

由于本文使用连续的核密度估计函数(见式(5))在离散的体素空间内构造了连续的概率密度分布，为计算密度分布的边界创造了条件，因此在确定分割轴后便可开始分割空间。具体为，首先通过核密度估计函数分别对x、y、z求二阶偏导数；其次分析二阶偏导数的零点，即通过一阶偏导数的极值点来确定3个轴向上的密度分布边界；最后由核密度估计函数(见式(5))对x、y、z求一阶偏导数，建立一阶偏微分方程。

$$\mu(x,y,z)=\frac{1}{nh^3}\sum_{i=1}^{n}K\left(\frac{dist}{h}\right) \tag{9}$$

$$\nabla\mu(x,y,z)=\frac{-15}{4nh^5}\sum_{i=1}^{n}\begin{bmatrix}(x-x_i)\left(1-\left(\dfrac{dist}{h}\right)^2\right),\\ (y-y_i)\left(1-\left(\dfrac{dist}{h}\right)^2\right),\\ (z-z_i)\left(1-\left(\dfrac{dist}{h}\right)^2\right)\end{bmatrix} \tag{10}$$

式中：$dist$为需要评估的三维点与样本数据中第i个三维点之间的距离。

由核密度估计函数(见式(5))对x、y、z求二阶偏导数，建立的二阶偏微分方程的计算公式如式(11)所示。根据二阶偏微分方程可求解出二阶偏导数的零点，若该零点左侧函数值为负，右侧为正，则说明该零点为一阶偏微分方程的极小值点，即视为在对应轴向上的密度边界点，此处密度下降最快。

$$\nabla^2\mu(x,y,z)=\frac{-15}{4nh^5}\sum_{i=1}^{n}\begin{bmatrix}\left(1-\left(\dfrac{dist}{h}\right)^2-\dfrac{2(x-x_i)^2}{h^2}\right),\\ \left(1-\left(\dfrac{dist}{h}\right)^2-\dfrac{2(y-y_i)^2}{h^2}\right),\\ \left(1-\left(\dfrac{dist}{h}\right)^2-\dfrac{2(z-z_i)^2}{h^2}\right)\end{bmatrix} \tag{11}$$

首先，选择3个轴向上的密度边界点对应一阶偏导数值最小的轴作为此次空间分割的分割轴。其次，确定分割值，由于明确了分割轴，只需对当前范围内体素点在该轴向上的分布进行聚类操作，这样的数据降维处理既降低了计算量又有效解决了基于密度的聚类算法(DBSCAN)对高维数据聚类效果差的问题。最后，判断由KDE计算出的体素密度峰值点在聚类结果的哪个簇中，并将该簇的边界值作为分割值。

3.2 整体构建过程

本文混合空间加速结构树的构建过程从根节点开始，根据KDE结果中的标准差情况来确定是否继续进行空间分割操作。若需要继续分割则再通过核密度函数的一、二阶偏微分方程与DBSCAN计算分割轴、分割值确定分割平面，若密度峰值点侧的子节点空间符合高密度评价标准就构建SDF节点，若不符合就继续划分。在明确分割区域后，对区域中包围盒进行缩小操作来保证每个包围盒最紧密，重复上述操作直至所有点节点无需继续分割，整体建过程如图7所示。

3.3 空间搜索

在光线追踪中，混合空间加速结构的搜索操作过程与其他加速结构类似，即计算一束光线是否与当前场景内的体素相交。首先从混合空间加速结构树的根节点(场景包围盒)开始，判断光线与包围盒是否相交，由于本文使用的是传统的轴对齐包围盒，因此判断速度非常快。若光线与场景包围盒相交，则继续与根节点的两个子节点所代表的包围盒进行相交测试，以此类推。如果当前计算的包围盒已是叶子节点的包围盒了，则需要作出前置判断，即如果该叶子节点为BVH节点，则返回光线与该叶子节点包围盒的交点以便后续采样；如果该叶子节点是为SDF节点，则将搜索方法改为在SDF中搜索。

光线与SDF的相交测试过程为：首先从光线起点(光线与SDF节点包围盒的交点)访问对应SDF体素点中距离

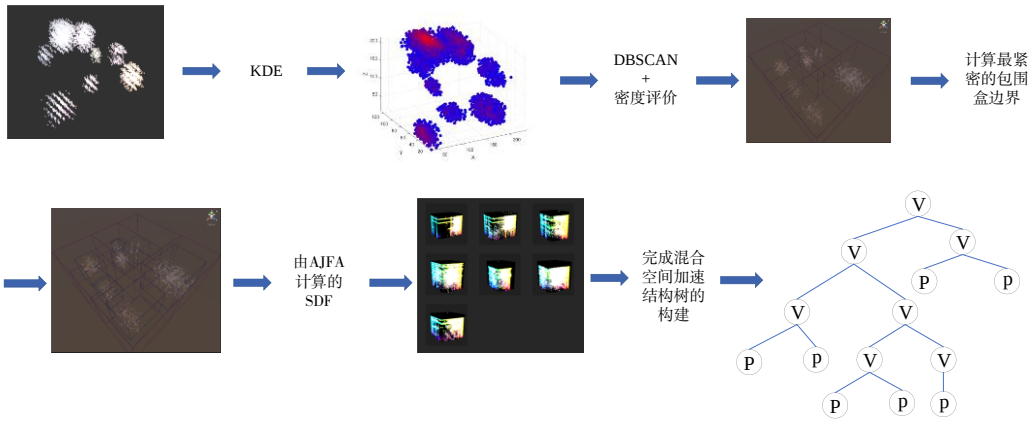


Fig. 7　Overall construction process of hybrid space accelerated structural tree

图 7　混合空间加速结构树的整体构建过程

该体素点最近有效体素点的距离，并将其作为下一次光线行进的安全距离（最小步进距离）；其次重复上述操作直至安全距离小于既定阈值，如图 8 所示。图 8 中红色圆代表有效体素点，蓝色圆代表光线行进过程中沿光线方向以安全距离长度作步进操作的节点，在到达节点 P6 后安全距离小于阈值，说明找到了相交的体素点。

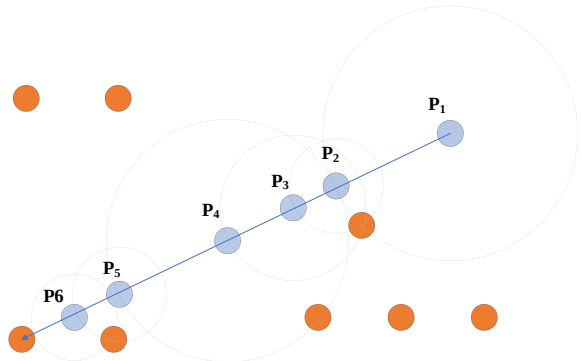


Fig. 8　Process of light traveling in SDF to find intersection points

图 8　光线在 SDF 中行进寻找交点的过程

4　实验结果与比较

在体素点数量与体素点密度分布不均的 4 个场景中，将本文方法与聚集聚类方法（AC）[4]、并行局部有序聚类方法（PLOC）[10]、密度峰值聚类（DPC）的方法[31]、引入 K-means 聚类构建的方法[3]进行比较。在构建速度方面，分别在体素点数量为 17 K，24 K，30 K 和 35 K 的场景中进行测试，结果如表 1 所示。由此可知，大体上空间加速结构的构建时间与体素点数量呈正相关，本文方法与文献[3]的方法在构建速度上具有一定的竞争力，由于使用了 GPU 作为计算单元，因此在计算时间上明显小于其他算法。

此外，为了测试聚类方法在体素点分布不均场景中构建加速结构的效果，本文选取了 0.225，0.367，0.571，0.640 等 4 个体素点的密度标准差的场景进行测试，测试结果如表 2 所示。需要注意的是，所用体素点密度是平均密度，即体素点数量除以空间大小，具体计算公式如式（12）所

Table 1　Time for constructing acceleration structures using 5 methods in 4 scenarios with different voxel point counts

表 1　体素点数量不同的 4 个场景中 5 种方法构建加速结构的时间　（ms）

构造方法	17K 体素	24K 体素	30K 体素	35K 体素
AC	207	219	352	383
PLOC	159	231	314	335
DPC	138	202	252	345
K-means	126	164	**215**	225
ours	**125**	**161**	217	**220**

Table 2　Construction time of 5 clustering methods in 4 scenarios with different voxel densities

表 2　体素密度不同的 4 个场景中 5 种聚类方法构建时间　（ms）

构造方法	0.225SD	0.367SD	0.571SD	0.640SD
AC	87	81	104	113
PLOC	105	114	111	116
DPC	92	103	90	104
K-means	68	71	79	88
ours	**61**	**70**	**74**	**80**

Table 3　Four methods for testing the intersection time of light rays in scenes with different voxel point densities

表 3　4 种方法在体素点密度不同的场景中光线相交测试的时间　（ms）

构造方法	0.225SD	0.367SD	0.571SD	0.640SD
AC	112	184	223	268
PLOC	127	148	202	272
DPC	105	161	187	249
K-means	93	135	**158**	220
ours	**92**	**126**	164	**206**

示。除了构建时间外，本文比较了 4 种方法的光线与体素点相交的时间，具体结果如表 3 所示。

$$\eta(v)=\frac{n\cdot \mathrm{M}}{V} \tag{12}$$

式中：$\eta(v)$为 v 空间下的密度均值；n 为有效体素点数量；M 为体单个素点质量一般取 1；V 为 v 空间的大小，一般为包围盒体积。体素点的密度标准差反映了整个场景中体素点密度分布的情况，若标准差越大则说明体素点密度分布的越不均匀。

通过上述实验结果可知,在体素点数量较大、分布不均匀的情况下,本文方法既能正确划分密度较高的区域,又能够保证较快的构建速度。尽管在个别情况下(表1、表2的第3种情况),本文方法的构建速度相较于文献[3]的方法稍慢,但在大部分情况下均具有较高的质量和效率。在光线相交测试方面,本文方法的检测速度依然最快。

5 结语

本文针对体素密度分布不均的复杂场景中构建的空间加速结构构建时间长、质量较差的问题,提出基于密度聚类的混合空间加速结构。首先,该结构依据核密度估计与基于密度的聚类结果,对空间中体素点进行划分,在内部节点和体素点稀疏区域使用BVH结构,而在体素点密集的区域使用SDF作为节点加入树形结构,以有效解决BVH结构在体素点密集区域生成的树形结构层次较深的问题。其次,本文提出自适应跳跃洪泛算法生成的SDF收敛速度更快,且在有效范围内精度更高。最后,还将GPU作为计算平台,满足了在大规模情况下模型并行计算的需求。

在具体应用方面,本文方法在渲染对象密度分布不均的情况下具有较好的应用效果,例如云的真实感渲染、水体中不同污染物分布等。实验表明,所提方法大幅缩短了加速结构树的构建时间,提升了构建精度、光线与体素点相交测试的效率,为基于三维体素的图像渲染中的光线追踪技术提供了有力支撑。

参考文献:

[1] MACDONALD J D, BOOTH K S. Heuristics for ray tracing using space subdivision[J]. The Visual Computer, 1990, 6: 153-166.

[2] LAUTERBACH C, GARLAND M, SENGUPTA S, et al. Fast BVH construction on GPUs [EB/OL]. https://research. nvidia. com/publication/2009-03_fast-bvh-construction-gpus.

[3] MEISTER D, BITTNER J. Parallel BVH construction using k-means clustering[J]. The Visual Computer, 2016, 32: 977-987.

[4] WALTER B, BALA K, KULKARNI M, et al. Fast agglomerative clustering for rendering[C]// 2008 IEEE Symposium on Interactive Ray Tracing, 2008: 81-86.

[5] WALD I. On fast construction of SAH-based bounding volume hierarchies [C]// 2007 IEEE Symposium on Interactive Ray Tracing, 2007: 33-40.

[6] WALD I. Fast construction of SAH BVHs on the Intel many integrated core (MIC) architecture[J]. IEEE Transactions on Visualization and Computer Graphics, 2010, 18(1): 47-57.

[7] MORTON G M. A computer oriented geodetic data base and a new technique in file sequencing[J]. Advances in Linear Algebra & Matrix Theory, 2014,4(1):1160.

[8] PANTALEONI J, LUEBKE D. HLBVH: hierarchical LBVH construction for real-time ray tracing of dynamic geometry[C]// Proceedings of the Conference on High Performance Graphics, 2010: 87-95.

[9] KARRAS T. Maximizing parallelism in the construction of BVHs, octrees, and k-d trees[C]// Proceedings of the Fourth ACM SIGGRAPH/Eurographics Conference on High-Performance Graphics, 2012: 33-37.

[10] WALTER B, BALA K, KULKARNI M, et al. Fast agglomerative clustering for rendering[C]// 2008 IEEE Symposium on Interactive Ray Tracing, 2008: 81-86.

[11] BENTHIN C, DRABINSKI R, TESSARI L, et al. Ploc++ parallel locally-ordered clustering for bounding volume hierarchy construction revisited[J]. Proceedings of the ACM on Computer Graphics and Interactive Techniques, 2022, 5(3): 1-13.

[12] CHITALU F M, DUBACH C, KOMURA T. Binary ostensibly-implicit trees for fast collision detection[C]// Computer Graphics Forum, 2020: 509-521.

[13] WALD I, MORRICAL N, ZELLMANN S, et al. Using hardware ray transforms to accelerate ray/primitive intersections for long, thin primitive types[J]. Proceedings of the ACM on Computer Graphics and Interactive Techniques, 2020, 3(2): 1-16.

[14] KENSLER A. Tree rotations for improving bounding volume hierarchies [C]// 2008 IEEE Symposium on Interactive Ray Tracing, 2008: 73-76.

[15] BITTNER J, HAPALA M, HAVRAN V. Fast insertion-based optimization of bounding volume hierarchies [C]// Computer Graphics Forum, 2013: 85-100.

[16] BITTNER J, MEISTER D. T-SAH: animation optimized bounding volume hierarchies[C]// Computer Graphics Forum,2015: 527-536.

[17] ROSENFELD A, PFALTZ J L. Sequential operations in digital picture processing[J]. Journal of the ACM, 1966, 13(4): 471-494.

[18] STRAIN J. Fast tree-based redistancing for level set computations[J]. Journal of Computational Physics, 1999, 152(2): 664-686.

[19] GUEZLEC A. Meshsweeper: dynamic point-to-polygonal mesh distance and applications[J]. IEEE Transactions on Visualization and Computer Graphics, 2001, 7(1): 47-61.

[20] KOSCHIER D, DEUL C, BRAND M, et al. An hp-adaptive discretization algorithm for signed distance field generation[J]. IEEE Transactions on Visualization and Computer Graphics, 2017, 23(10): 2208-2221.

[21] WU Y, MAN J, XIE Z. A double layer method for constructing signed distance fields from triangle meshes[J]. Graphical Models, 2014, 76(4): 214-223.

[22] MAUCH S. A fast algorithm for computing the closest point and distance transform[EB/OL]. http://www. acm. caltech. edu/seanm/software/cpt/cpt. pdf.

[23] MAUCH S P. Efficient algorithms for solving static Hamilton-Jacobi equations[M]. California: California Institute of Technology, 2003.

[24] SIGG C, PEIKERT R, GROSS M. Signed distance transform using graphics hardware[C]// IEEE Visualization, 2003: 83-90.

[25] PUJOL E, CHICA A. Adaptive approximation of signed distance fields through piecewise continuous interpolation[J]. Computers & Graphics, 2023, 114: 337-346.

[26] GALIN E, GUÉRIN E, PARIS A, et al. Segment tracing using local Lipschitz bounds[C]// Computer Graphics Forum, 2020: 545-554.

[27] JIANG Y, JI D, HAN Z, et al. Sdfdiff: differentiable rendering of signed distance fields for 3D shape optimization[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2020: 1251-1261.

[28] PARK J J, FLORENCE P, STRAUB J, et al. Deepsdf: learning continuous signed distance functions for shape representation[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019: 165-174.

[29] MICHALKIEWICZ M, PONTES J K, JACK D, et al. Deep level sets: Implicit surface representations for 3D shape inference[DB/OL]. https://arxiv. org/abs/1901. 06802.

[30] RONG G, TAN T S. Jump flooding in GPU with applications to Voronoi diagram and distance transform[C]// Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games, 2006: 109-116.

[31] RODRIGUEZ A, LAIO A. Clustering by fast search and find of density peaks[J]. Science, 2014, 344(6191): 1492-1496.

(责任编辑:刘嘉文)