文章编号:1001-5078(2023)01-0137-09　　·图像及信号处理·

基于点云空间分布特征的多级索引结构

杨丽娟[1,2],崔钰琳[1,2],杨紫骞[1,2],翟光杰[1,2],王　超[1,2]
(1. 中国科学院国家空间科学中心,北京 100190;2. 中国科学院大学,北京 100049)

摘　要:为解决点云数据分布不规则、非均匀产生的查询效率低下的问题,提出了一种基于三维点云数据空间分布特征的多级索引结构。将点云空间信息引入传统八叉树,形成一种新的数据结构——方向八叉树,用于点云空间的全局划分。在每次划分空间之前,先对点云数据进行主成分分析,形成节点的方向包围盒,再进一步将空间划分为八个子空间。为了实现数据的快速调度与查询,在局部,使用 KD 树对方向八叉树的叶子节点进行二次组织构建。实验结果表明,方向八叉树能有效减少节点总数和冗余节点数量;方向八叉树和 KD 树的组合嵌套结构可以有效划分海量点云数据,实现点云数据的高效检索,对点云数据进行有效管理。

关键词:点云数据;方向八叉树;KD 树;索引结构
中图分类号:TP391　　**文献标识码:**A　　**DOI:**10.3969/j.issn.1001-5078.2023.01.021

Multi-level index structure based on spatial distribution characteristics of point cloud

YANG Li-juan[1,2], CUI Yu-lin[1,2], YANG Zi-qian[1,2], ZHAI Guang-jie[1,2], WANG Chao[1,2]
(1. National Space Science Center, Chinese Academy of Sciences, Beijing 100190, China;
2. University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: In order to solve the problem of inefficient querying arising from irregular and uneven distribution of point cloud data, a multi-level index structure based on the spatial distribution characteristics of 3D point cloud data is proposed. The point cloud spatial information is introduced into the traditional octree to form a new data structure-the oriented octree, which is used for the global division of point cloud space. Before each division of the space, a principal component analysis is performed on the point cloud data to form an oriented bounding box of the nodes, and then the space is further divided into eight subspaces. To achieve fast scheduling and querying of data, KD tree is used for the secondary organization and construction of the data in the leaf nodes locally. The experimental results show that the oriented octree can effectively reduce the total number of nodes and the number of redundant nodes, the combined nested structure of oriented octree and KD tree can effectively divide the huge amount of point cloud data, realize efficient retrieval of point cloud data, and effectively manage point cloud data.

Keywords: point cloud; oriented octree; KD tree; index structure

---

基金项目:国家重点研发计划项目(No. 2016YFE0131500);中国科学院青年创新促进会优秀会员项目(No. 2013105;No. Y201728);发改委国家重大科技基础设施项目(No. 2018YFA0404201;No. 2018YFA0404202)资助。
作者简介:杨丽娟(1997 -),女,硕士研究生,主要从事三维点云数据处理与可视化等方面的研究。E-mail:yanglijuan19@mails.ucas.ac.cn
通讯作者:王　超(1981 -),男,副研究员,主要从事图像处理,嵌入式应用等方面的研究。E-mail:wangchao@nssc.ac.cn
收稿日期:2022-02-16;**修订日期:**2022-04-05

1 引 言

点云数据是三维空间中离散分布的一组点,广泛应用于地形测量、文物保护、三维重建、城市规划、智能驾驶、虚拟现实等领域[1-4]。点云空间分布离散无序,数据量巨大,为后续的数据处理带来了挑战。建立合理高效的空间索引机制,是实现数据高效检索和快速调度的关键,是后续数据处理的前提[5]。传统的单一索引模型难以对海量点云数据进行高效组织管理,混合索引模型通常结合了两种及以上不同索引的优势,是当前研究的重点[6]。

文献[7]提出了一种八叉树和三维 R 树集成的空间索引方法,显著提升了空间利用率和空间查询效率。文献[8]提出了一种多级格网和 KD 树相结合的混合空间索引,既提升了查询效率,又解决了单一分辨率数据冗余的问题。文献[9]将点云的方向信息引入传统的模糊 c-均值,并使用 BSP 树对点云进行逐点划分,使索引能够沿着点云的空间结构扩展,避免产生不必要的分区。文献[10]提出了一种全局 KD 树和局部八叉树相结合的两级混合索引结构,实现了树结构的均衡,并能以块为单位对海量点云进行快速检索。文献[11]提出了一种结合 KD 树空间切分思想的类八叉树索引结构,降低了内存空间占用和邻域搜索耗时。文献[12]将快速划分空间的八叉树和高效查询空间的三维 R^* 树相结合,构建了一种名为 3DOR*-tree 的混合空间索引结构,实现了三维地质四面体模型的有效访问和高效查询。文献[13]针对地铁隧道的点云数据特点,提出了一种格网和多分辨率八叉树结合的索引模型,提升了空间分布不平衡但集中的线状点云的索引构建效率和质量。

这些方法的空间划分都是基于空间规律性和轴对齐包围盒的,无法表达点云本身的空间结构。因此,针对点云分布的不规则性和非均匀性,将方向包围盒引入传统的规则八叉树结构,提出了一种方向八叉树空间划分方法。在全局索引中,采用方向八叉树组织点云数据。通过对点云结构进行主成分分析,自适应地计算包含一组点的每个节点的方向包围盒。为提升索引模型的检索能力,在局部索引中,增加 KD 树来管理方向八叉树的叶子节点。

2 方向八叉树索引结构

2.1 八叉树结构

八叉树结构通过对大小为 $2^n \times 2^n \times 2^n$ 的三维空间实体进行循环递归的体元剖分,其中每个体元的时间和空间复杂度相同,从而构成一个方向图[14]。如果被剖分的体元属性相同,则构成一个八叉树的叶节点;否则将该体元划分为 8 个子立方体,并依次递归划分。八叉树划分及结构示意图如图 1 所示。

传统的八叉树首先为点云数据建立轴向包围盒,之后递归地将空间规则地划分成八个均匀的子立方体,并将空间中的数据分配到相应的子立方体中,可以实现对海量点云数据的高效管理。但是,易产生大量的空白节点,影响树的平衡性[15]。

2.2 方向包围盒

方向包围盒是沿着物体的主成分方向生成的最小立方体包围盒。因此,相较于轴向包围盒,方向包围盒可以根据物体的形状特征尽可能紧密地逼近物体,紧密性更好,能显著降低冗余空间。计算方向包围盒主要是借助顶点坐标的一阶及二阶统计特性来确定最佳方向,并寻找包围盒在该方向上的最小尺寸[16]。

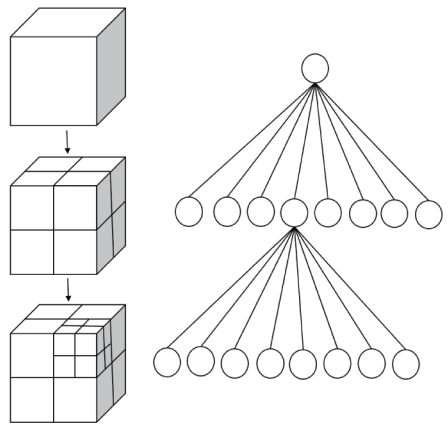


图 1 八叉树结构示意图

Fig. 1 Schematic diagram of octree structure

本文采用了一种利用三角网计算方向包围盒的方法[17],具体步骤如下:

Step1:三角面片的平均向量如式(1)所示:

$$\mu = \frac{1}{3n}\sum_{i=1}^{n}(p^i + q^i + r^i) \tag{1}$$

其中,n 代表三角面片的总数;p^i、q^i 和 r^i 分别表示的是第 i 个三角面片的三个顶点的坐标向量。

Step2：协方差矩阵 C 计算如下：

$$C_{jk}=\frac{1}{3n}\sum_{i=1}^{n}(\bar{p}_j^i\bar{p}_k^i+\bar{q}_j^i\bar{q}_k^i+\bar{r}_j^i\bar{r}_k^i),1\leqslant j,k\leqslant 3 \tag{2}$$

其中，C_{jk} 是指协方差矩阵中第 j 行第 k 列的元素，$\bar{p}^i=p^i-\mu,\bar{q}^i=q^i-\mu,\bar{r}^i=r^i-\mu$。

Step3：计算协方差矩阵 C 的特征向量，确定方向包围盒的方向和尺寸。对特征向量进行单位化，将其作为一个基。由于矩阵 C 是对称矩阵，因此特征向量基是正交的。沿基的每个轴向找到该轴向上的极值顶点，并由极值顶点确定方向包围盒的尺寸[18]。将轴向作为方向包围盒的方向。

2.3　方向八叉树的思想与构建

方向八叉树是一种用来描述三维空间的树状结构模型。方向八叉树的每个非叶子节点代表对应空间数据的方向包围盒。每个节点空间可以均匀分割成 8 个子立方体，8 个子立方体对应的点集组成当前节点的 8 个子节点。如果子节点满足分割条件，则对其进行递归划分，直至满足分割停止条件。

方向八叉树的输入是原始点云集 P，组织构建步骤为：

Step1：初始化分割阈值 T_{oc}。本次实验中，T_{oc} 设为原始点集数量的 0.2 %。

Step2：使用原始点云集 P 作为树的根节点。

Step3：如果当前节点中的点云数量大于 T_{oc}，计算节点的方向包围盒。

Step4：根据包围盒将空间分解成 8 个子立方体，并将每个子立方体对应的点集作为当前节点的 8 个子节点：

由包围盒的中心位置，转换节点中的点集坐标：

$$(x'_i,y'_i,z'_i)=(x_i-\bar{x}_{obb},y_i-\bar{y}_{obb},z_i-\bar{z}_{obb}) \tag{3}$$

其中，(x'_i,y'_i,z'_i) 表示点集中第 i 个点转换后的坐标，(x_i,y_i,z_i) 表示第 i 个点的原始坐标，$(\bar{x}_{obb},\bar{y}_{obb},\bar{z}_{obb})$ 表示方向包围盒的中心位置坐标。

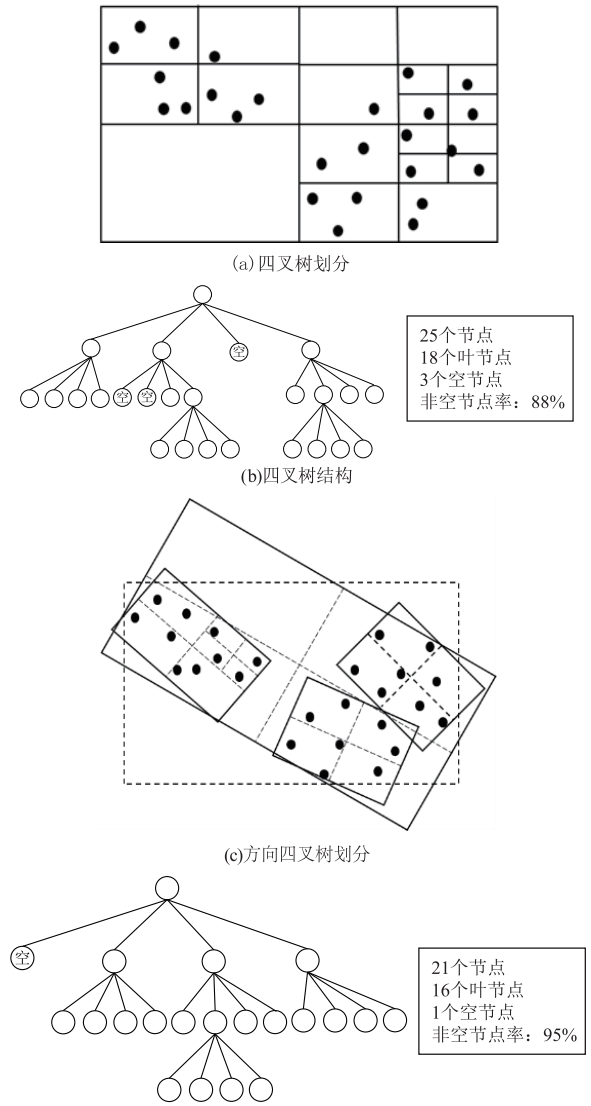
根据转换后的坐标点在每一轴上的正负性，将原始点分配到对应的子节点中。

Step5：如果子节点所存储的点云数量与父节点一样且不为零，则停止当前节点的细分。

Step6：迭代执行步骤(3)至(5)，直到节点中的点云数量均小于或等于 T_{oc}。

2.4　八叉树与方向八叉树的比较

由于三维点集的划分较为复杂，为了清晰明了地展示思路，如图 2 所示，分别使用四叉树和方向四叉树对二维点集的空间划分来示意比较。树的分割阈值设置为 3。图 2(a)、图 2(b)分别表示四叉树的空间划分和划分后形成的树结构，图 2(c)、图 2(d)分别表示方向四叉树的空间划分和划分后形成的树结构。非空节点率定义为所有节点中非空节点的占比。四叉树产生了 25 个节点(18 个叶节点，3 个空节点)，非空节点率是 88 %；方向四叉树产生了 21 个节点(16 个叶节点，1 个空节点)，非空节点率是 95 %。



(a)四叉树划分

(b)四叉树结构

(c)方向四叉树划分

(d)方向四叉树结构

图 2　二维点集的空间划分比较

Fig. 2 Comparison of spatial partitioning for a 2D point set

可以看出,与四叉树相比,方向四叉树在节点总数、叶节点数、空节点数和非空节点率上都有更好的结果,有助于减少节点数量和冗余节点。

3 多级索引结构

3.1 KD 树结构

KD 树是一种对多维欧氏空间进行划分而构造的二叉树。每个非叶节点代表一次空间划分。每次划分时,选择其中某一维度进行比较,并使用一个合适的数据点作为划分标准,将当前空间划分成两个子空间[19]。可将垂直于划分维度且经过划分数据点的平面视作一个超平面。节点的左子树代表超平面左边的点,右子树代表超平面右边的点。

为了使 KD 树具有更好的平衡性,每次划分应尽量使数据点集均匀分割成两部分。通常,与其他维度相比,数据点集在划分维度上应分布尽量分散[20]。因此,可选择所有数据点方差最大的维度作为划分维度,并取划分维度上的中位数对应的点作为划分数据点。以划分数据点在划分维度上的值为标准,将其余数据点分配到左、右子树上。如果当前数据点在划分维度上对应的值小于标准值,则将其划分到当前节点的左子树,反之,则划分到当前节点的右子树。如图 3 所示,KD 树将二维及三维空间分割成多个子空间。

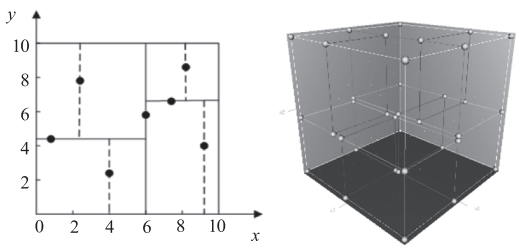


图 3 KD 树分割

Fig. 3 KD tree segmentation

3.2 多级索引结构的思想

KD 树能实现点云的快速查找,但由于建树期间占用内存过大,难以对海量点云进行构建。而方向八叉树构建简单、快速,查询效率受限于分割阈值。因此,可结合二者优势,基于方向八叉树和 KD 树建立混合索引结构。在上层采用方向八叉树对点云进行全局划分,在下层使用 KD 树对局部点云信息进行组织,递归划分方向八叉树叶节点,直至 KD 树叶节点内的点数不超过提前设置的阈值。划分后的空间信息存储在方向八叉树叶节点中,如图 4 所示,形成全局方向八叉树和局部 KD 树的多层混合索引结构。

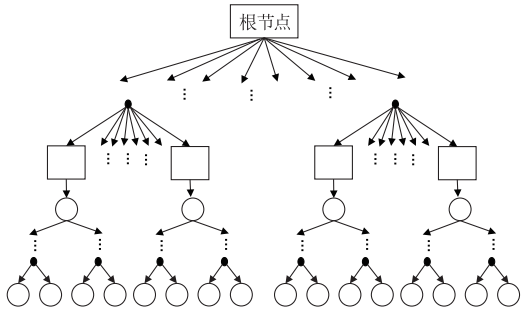


图 4 多级索引结构

Fig. 4 Muti-level index structure

3.3 多级索引结构的构建

建立基于三维点云空间分布特征的多级索引结构的具体步骤如下:

Step1:使用初始点云 P 构建一个方向八叉树。

Step2:为每一个叶子节点 Q_i 构建 KD 树,具体操作如下:

①叶子节点的点集作为 KD 树的根节点。

②如果当前节点中的点数大于 T_{kd},当前节点继续细分。其中,$T_{kd} = N/2^m$,N 表示点云数据集的大小,m 表示 KD 树递归分解层数。

③根据点集空间的轴向包围盒,计算点集在各个维度上的方差:

$$s_x = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{4}$$

$$s_y = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{5}$$

$$s_z = \frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})^2 \tag{6}$$

其中,n 是指点集中点的数量;x_i、y_i 和 z_i 表示的是点集中第 i 个点的三维坐标;$\bar{x}$、$\bar{y}$ 和 $\bar{z}$ 分别表示点集中所有点在 x 轴、y 轴和 z 轴上的平均值。

选择具有最大方差的维度,来确定分割维度:

$$s_d = \max(s_x, s_y, s_z) \tag{7}$$

以空间中分割维度上的中值点作为分割点,根据分割维度,将空间划分成两个子空间,子空间中的

点集作为当前节点的子节点。

④迭代执行②至③，直至节点中的点数均小于或等于 T_{kd}。叶子节点中的点集以及每个节点点集的包围盒，即为划分结果。

索引构建的流程如图 5 所示。

4　实验结果与分析

为验证所提方向八叉树在节点冗余方面的提升，本文使用经典八叉树与方向八叉树进行对比实验。同时，为验证本文基于三维点云空间分布特征的多级索引的结构有效性，将 KD 树、八叉树、四叉树—KD 树与基于三维点云空间分布特征的多级索引进行比较。比较指标包括构建索引消耗时间、邻域搜索时间。最后，通过实验探索方向八叉树分割阈值对多级索引结构构建时间以及构建内存的影响。

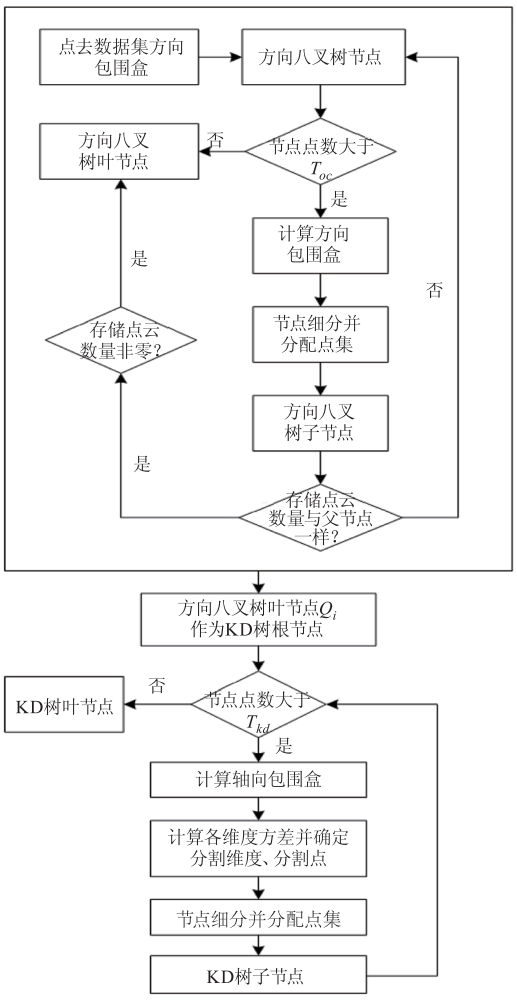


图 5　多级索引构建流程

Fig. 5 Muti-level index building process

4.1　实验数据与环境

根据空间分布特征和数据量，本文选择了三种类型的点云数据进行实验。第一类数据是 The Stanford 3D Scanning Repository 中的典型点云数据；第二类数据是 RGB-D Object Dataset 中的常见室内环境点云数据；第三类数据是大型自然场景的海量三维点云数据。数据点的数量级分布是10^3 ~ 10^8。实验环境是 Intel(R) Core(TM) i5 - 7200U CPU @ 2.50 GHz，12 GB 内存。

4.2　方向八叉树与八叉树节点实验

表 1 显示了八叉树与方向八叉树的节点对比，对比项包括节点总数、空节点数和非空节点率。在所有数量级的点云数据上，与八叉树相比，方向八叉树生成的总节点数和空节点数均更少。同时，方向八叉树结构的非空节点率有较为显著的提升。八叉树的非空节点率集中分布在 83 % ~ 90 %，而方向八叉树的非空节点率集中在 88 % ~ 96 % 范围内，空间利用率提高了 5 %。

4.3　索引构建时间实验

四种索引的构建时间对比如表 2 所示。其中，KD 树建树耗时最长，远大于其他三种方法，且随着点云数量级的增加，差异逐渐扩大。当点云数量达到 100 万时，KD 树建树耗时比其他方法多65 ~ 91 s。当点云数量超过 1000 万时，KD 树会由于内存不足而无法完成建树。四种方法中，八叉树索引构建速度最快。由于引入了方向信息，基于三维点云空间分布特征的多级索引建树时间增加，但明显快于四叉树—KD 树混合索引，与八叉树相差不大。

4.4　邻域搜索耗时实验

邻域搜索耗时是指搜索指定的每一个点的最邻近点所消耗的平均时间。本文方法与其他三种方法相应指标对比如表 3 所示。通过对比可知，八叉树搜索耗时最长，查询效率远低于 KD 树。与单一索引结构的查询效率相比，混合结构的索引方式的查询效率有着明显优势。本文方法和四叉树—KD 树结构查询效率均在 KD 树基础上有所提升，相比于八叉树结构提升了1 ~ 2 个数量级。与四叉树—KD 树混合索引结构相比，本文方法邻域搜索速度更快，具有更好的查询性能。

表 1 各方法针对不同点云数据的节点比较

Tab. 1 Node comparison of each method for different point cloud data

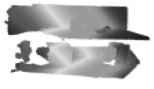
点云图	点云数量	索引类型	节点总数	空节点数	非空节点率
	1494	八叉树 方向八叉树	57 **39**	16 **10**	71.93 % **74.36** %
	2903	八叉树 方向八叉树	78 **73**	11 **0**	85.90 % **100** %
	35947	八叉树 方向八叉树	1042 **987**	191 **86**	81.67 % **91.29** %
	156112	八叉树 方向八叉树	2646 **2414**	75 **35**	97.17 % **98.55** %
	172974	八叉树 方向八叉树	2353 **2202**	399 **248**	83.04 % **88.74** %
	437645	八叉树 方向八叉树	3609 **3267**	486 **296**	86.53 % **90.94** %
	737587	八叉树 方向八叉树	4817 **4666**	696 **367**	85.55 % **92.13** %
	917991	八叉树 方向八叉树	6230 **5767**	786 **355**	87.38 % **93.84** %
	990096	八叉树 方向八叉树	6376 **6256**	681 **321**	89.32 % **94.87** %
	1033625	八叉树 方向八叉树	6931 **6451**	910 **518**	86.87 % **91.97** %
	1064032	八叉树 方向八叉树	7046 **6933**	723 **284**	89.74 % **95.90** %
	1272877	八叉树 方向八叉树	8523 **8068**	981 **502**	88.49 % **93.78** %
	12000000	八叉树 方向八叉树	18144 **16966**	3043 **2137**	83.23 % **87.40** %

表 2　各方法索引构建时间对比(单位:s)

Tab. 2 Comparison of index construction time of each method(Unit:s)

点云数量	KD	八叉树	四叉树 + KD 树	本文方法
2903	0. 32	0. 03	0. 06	0. 04
35947	3. 91	0. 62	1. 13	0. 94
156112	13. 35	2. 93	4. 92	4. 03
437645	37. 87	7. 92	13. 67	10. 51
917991	91. 48	21. 39	37. 36	27. 16
1272877	124. 07	32. 40	58. 82	39. 13
12000000	——	361. 48	603. 89	438. 71

表 3　各方法邻域搜索时间对比(单位:ms)

Tab. 3 Comparison of neighborhood search time of each method(Unit:ms)

点云数量	KD 树	八叉树	四叉树 + KD 树	本文方法
437645	0	310	0	0
917991	9	626	7	5
1272877	20	951	16	13
12000000	——	6574	517	491

4. 5　方向八叉树分割阈值实验

不同数量级的点云数据构建混合索引所消耗的时间、空间与方向八叉树分割阈值的关系如图 6、图 7 所示。索引构建时间及内存占用主要与点云数据量相关,不同方向八叉树分割阈值对同一点云的构建影响较小。同时,点云数据量越大,方向八叉树分割阈值产生的影响越大。对于数据量较大的点云,当上层方向八叉树分割阈值逐渐增大时,构建索引所消耗的时间和空间先有明显的下降,之后下降幅度逐渐减弱,最后趋于平缓。因此,对海量点云的索引构建而言,选择合适的方向八叉树分割阈值能节省大量时间及空间占用,具有重要意义。

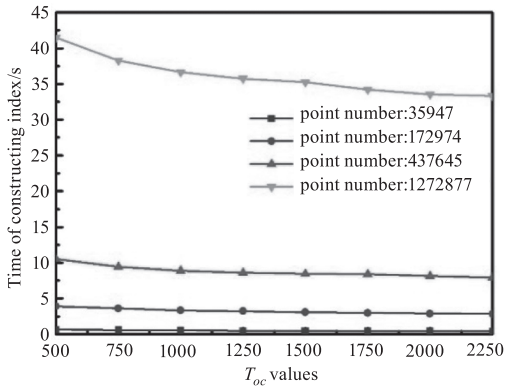


图 6　不同方向八叉树阈值下构建索引时间

Fig. 6 Index constructing time under different oriented octree thresholds

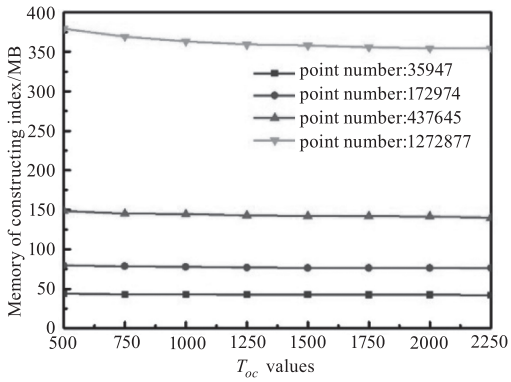


图 7　不同方向八叉树阈值下构建索引占用内存

Fig. 7 Memory consumption of index construction under different oriented octree thresholds

5　总　结

本文针对海量不规则点云管理困难,时空消耗大的问题,根据点云分布特征,将空间分布特征引入八叉树,提出了一种新的空间划分方法——方向八叉树,来适应点云数据的非均匀空间分布。在此基础上分析了方向八叉树和 KD 树的优缺点,并设计了方向八叉树与 KD 树结合的双层点云管理模型,进一步提高索引查询效率,对点云数据进行合理管理。实验表明,方向八叉树结构相比于传统八叉树,空间利用率提高了 5 % 左右。且本文提出的基于空间分布特征的多级索引结构,构建索引耗时接近于八叉树,相比于 KD 树、四叉树—KD 树混合索引提高了 25 %;与其他三种索引结构相比,邻域搜索效率提高了 18 %,充分验证了本文方法的有效性。

但本文提出的索引方法仍有很多可改进之处。例如,对海量点云数据而言,上层方向八叉树分割阈值对索引构建时间、空间有较大影响,但目前难以通过简单、自动的方法确定最优的分割阈值;方向八叉树与八叉树的树结构高度相差不大,如何更加充分的利用点云数据分布特征来构建索引,使树具有更好的平衡性,是未来的研究目标。

参考文献:

[1] Zeng Xun. Visualization of Airborne LIDAR point cloud data based on WebGL[J]. Journal of Hunan University of Science and Technology: Natural Science Edition, 2012, 27(4):60 - 64. (in Chinese)

曾珣. 基于 WebGL 的机载激光雷达点云数据可视化[J]. 湖南科技大学学报:自然科学版,2012,27(4):60 – 64.

[2] Chen Chi, Wang Ke, Xu Wenxue, et al. Fast visualization method of massive vehicle laser scanning point cloud data [J]. Journal of Wuhan University: Information Science Edition, 2015, 40(9): 1163 – 1168. (in Chinese)
陈驰,王珂,徐文学,等. 海量车载激光扫描点云数据的快速可视化方法[J]. 武汉大学学报:信息科学版,2015,40(9):1163 – 1168.

[3] Jing Lu, Wu Bin, Fang Xi-lu. Point cloud registration method based on the SIFT feature points combined with ICP algorithm[J]. Laser & Infrared, 2021, 51(7): 944 – 950. (in Chinese)
荆路,武斌,方锡禄. 基于 SIFT 特征点结合 ICP 的点云配准方法[J]. 激光与红外,2021,51(7):944 – 950.

[4] Han Fengze, Li Guodong, Han Yifei, et al. ToF point cloud intensity and location-related target extraction algorithm [J]. Laser & Infrared, 2020, 50(12): 1521 – 1528. (in Chinese)
韩丰泽,李国栋,韩一菲,等. ToF 点云强度与位置相关联的目标提取算法[J]. 激光与红外,2020,50(12):1521 – 1528.

[5] van Oosterom, Peter. Massive point cloud data management: Design, implementation and execution of a point cloud benchmark [J]. Computers & Graphics, 2015, 49 (jun.): 92 – 125.

[6] Yu Anbin, Mei Wensheng. A massive subway tunnel point cloud management method based on the combination of R-tree and grid[J]. Journal of Wuhan University: Information Science Edition, 2019, 44(10): 1553 – 1559. (in Chinese)
于安斌,梅文胜. 一种 R 树与格网结合的海量地铁隧道点云管理方法[J]. 武汉大学学报:信息科学版,2019,44(10):1553 – 1559.

[7] Gong Jun, Ke Shengnan, Zhu Qing, et al. An efficient management method for point cloud data based on octree and 3D R-tree[J]. Acta Geodaetica et Cartographica Sinica, 2012, 41(4): 597 – 604. (in Chinese)
龚俊,柯胜男,朱庆,等. 一种八叉树和三维 R 树集成的激光点云数据管理方法[J]. 测绘学报,2012,41(4):597 – 604.

[8] Zhao Jianghong, Wang Jiwei, Wang Yanmin, et al. A new multilevel spatial index for scattered point cloud data[J]. Journal of Earth Information Science, 2015, 17(12): 1450 – 1455. (in Chinese)
赵江洪,王继伟,王晏民,等. 一种新的散乱点云数据多级空间索引[J]. 地球信息科学学报,2015,17(12):1450 – 1455.

[9] Zhang Yi. The D-FCM partitioned D-BSP tree for massive point cloud data access and rendering. [J]. ISPRS Journal of Photogrammetry & Remote Sensing, 2016, 20(oct): 25 – 36.

[10] Zhang Rui, Li Guangyun, Wang Li, et al. A new hybrid index method for vehicle LIDAR point cloud[J]. Journal of Wuhan University: Information Science Edition, 2018, 43 (7): 993 – 999. (in Chinese)
张蕊,李广云,王力,等. 车载 LiDAR 点云混合索引新方法[J]. 武汉大学学报:信息科学版,2018,43(7):993 – 999.

[11] Lv Min, Meng Yun. Research on point cloud management strategy based on class octree index[J]. Progress in Laser and Optoelectronics, 2020, 57(14): 233 – 242. (in Chinese)
吕敏,孟芸. 基于类八叉树索引的点云管理策略研究[J]. 激光与光电子学进展,2020,57(14):233 – 242.

[12] Wang Y, Lv H, Ma Y. Geological tetrahedral model-oriented hybrid spatial indexing structure based on Octree and 3D R*-tree[J]. Arabian Journal of Geosciences, 2020, 13 (15): 1 – 11.

[13] Mei Wensheng, Li Tianjiao, Yu Anbin. Massive subway tunnel point cloud data management and web fast visualization method[J]. Journal of Geomatics, 2021, 46(2): 1 – 6. (in Chinese)
梅文胜,李天骄,于安斌. 海量地铁隧道点云数据管理与 Web 快速可视化方法[J]. 测绘地理信息,2021,46(2):1 – 6.

[14] Jackins C L, Tanimoto S L. Oct-trees and their use in representing three-dimensional objects [J]. Computer Graphics & Image Processing, 1980, 14(3): 249 – 270.

[15] Wang Y, Lv H, Ma Y. Geological tetrahedral model-orien-

ted hybrid spatial indexing structure based on Octree and 3D R*-tree[J]. Arabian Journal of Geosciences, 2020, 13(15): 1 – 11.

[16] LI Yunfeng, LIU Xiuguo. Algorithm of contours splicing based on OBB projection transformation[J]. Journal of Computer Applications, 2011, 31(12): 3353 ~ 3356. (in Chinese)
李运锋,刘修国. 基于方向包围盒投影转换的轮廓线拼接算法[J]. 计算机应用,2011,31(12):3353 – 3356.

[17] Gottschalk S, Lin M C, Manocha D. OBBTree: a hierarchical structure for rapid interference detection[J]. Acm Siggraph Computer Graphics, 1996, 30(Annual Conference Series).

[18] Wang Wei, Ma Jun, Liu Wei. Research and application of collision detection based on OBB bounding box[J]. Computer Simulation, 2009, (9): 5. (in Chinese)
王伟,马峻,刘伟. 基于 OBB 包围盒的碰撞检测研究与应用[J]. 计算机仿真,2009,(9):5.

[19] Behley J, Steinhage V, Cremers A B. Efficient radius neighbor search in three-dimensional point clouds[C]// Proceedings-IEEE International Conference on Robotics and Automation, 2015: 3625 – 3630.

[20] Lu Y, Cheng L, Isenberg T, et al. Curve complexity heuristic KD-trees for neighborhood-based exploration of 3D curves[J]//Computer Graphics Forum, 2021, 40(2): 461 – 474.