引文格式:李松,吴楠,靳海鹏,等.三维激光点云数据索引和近邻查询方法研究[J].武汉大学学报(信息科学版),2025,50(11):2295-2302.DOI:10.13203/j.whugis20230374

Citation:LI Song,WU Nan,JIN Haipeng,et al.3D Laser Point Cloud Data Index and Nearest Neighbor Query Method[J].Geomatics and Information Science of Wuhan University,2025,50(11):2295-2302.DOI:10.13203/j.whugis20230374

三维激光点云数据索引和近邻查询方法研究

李 松[1] 吴 楠[1] 靳海鹏[1] 张丽平[1] 孙广路[1]

1 哈尔滨理工大学计算机科学与技术学院,黑龙江 哈尔滨,150080

摘 要:为了解决激光点云数据分布不均匀和点云数据近邻查询效率较低的问题,提出了一种基于激光点云索引结构的近邻查询方法。基于Z曲线,该方法首先将激光点云数据信息传递到由局部KD树和局部八叉树组成的KD_ZoC索引结构中,在对激光点云数据进行划分时,根据节点的Z地址形成最小立方体,并进一步将空间划分为子空间。其次,为了实现数据的快速检索,采用局部KD树和局部八叉树的方式对叶子节点进行分组,并根据提出的八分圆域进一步减少数据点的数量。最后,将数据存储到链表中,并结合数据的Z地址实现高效的数据近邻查询。理论研究和实验结果表明,所提出的方法能有效地将激光点云下的近邻查询效率提高20%。此外,相比于传统查询方法,八分圆域划分方法能够有效地剪枝对查询结果无影响的数据点从而实现数据点的二次精炼。

关键词:激光点云数据;KD树;八叉树;近邻查询;Z曲线

中图分类号:P237 **文献标识码**:A **收稿日期**:2024-05-29

DOI:10.13203/j.whugis20230374 **文章编号**:1671-8860(2025)11-2295-08

3D Laser Point Cloud Data Index and Nearest Neighbor Query Method

LI Song[1] *WU Nan*[1] *JIN Haipeng*[1] *ZHANG Liping*[1] *SUN Guanglu*[1]

1 College of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China

Abstract: **Objectives:** In order to solve the problem of uneven distribution of laser point cloud data and low efficiency of point cloud data neighbor query, this paper proposes a neighbor query method based on laser point cloud index structure. **Methods:** Based on the Z curve, the method first transfers the information of laser point cloud data to the KD_ZoC index structure composed of local KD tree and local octree. When the laser point cloud data is divided, the minimum cube is formed according to the Z address of the node, and the space is further divided into subspaces. Second, in order to achieve fast data retrieval, leaf nodes are grouped by local KD tree and local octree, and the number of data points is further reduced according to the proposed octant domain. Finally, the data is stored in the linked list, and the Z address of the data is combined to achieve efficient nearest neighbor query. **Results:** This paper makes the following contributions: (1) Aiming at the problem that the single index structure is hard to effectively process laser point cloud data, this paper proposes a new index structure, the KD_ZoC index structure, based on the Z curve. It constructs a hybrid index structure of local octrees and local KD trees, and introduces threshold values for the data division of both components. This effectively reduces laser point cloud data, solving the problem of difficult data processing when the laser point cloud data volume is large. (2)To address the problem of comparing multi-dimensional data, this paper uses the Z curve for dimensionality reduction of point cloud data. It provides a specific conversion formula to transform the data into binary data, which is stored in the leaf nodes of the leaf octree. This effectively reduces the comparison time for nearest neighbor queries and saves storage space. (3)To improve the efficiency of nearest-neighbor queries for point cloud data, this paper proposes an

基金项目:国家自然科学基金项目(62072136);黑龙江省重点研发计划(JD2023SJ20)。

第一作者:李松,博士,教授,主要从事数据索引和查询、激光点云数据处理与分析以及数据挖掘研究。lisongbeifen@163.com

通信作者:吴楠,硕士。1047889768@qq.com

octant circle neighborhood search method based on the KD_ZoC index structure. By utilizing symmetry for effective data reduction, it reduces computational time costs and presents an efficient nearest - neighbor query algorithm. **Conclusion**: Theoretical research and experimental results show that the proposed method can effectively deal with the nearest neighbor query problem under the laser point cloud, and realize the efficient retrieval and management of data points.

Key words: laser point cloud data; KD tree; octree; nearest neighbor query; Z curve

随着激光点云扫描技术的快速发展,激光点云已经成为物体的一种重要表达方式,被广泛地应用在多个领域中,例如虚拟现实、自动驾驶、3D扫描及3D打印机等。除了几何体信息,激光点云还包括许多其他属性,如颜色和法线。点云的处理和应用,例如近邻查询[1]、数据压缩[2]、匹配和配准[3-6]、分类和分割[7-10],已成为工业界和学术界的重要研究课题。与传统的2D点云不同,3D点云的数据结构是不规则的,在特定的应用中,3D点云的数据量往往是巨大的。因此,传统的单一索引模型就很难实现有效的数据组织、数据降维和数据查询,研究有效地针对激光点云数据信息的索引和查询方法具有重要的意义。文献[11]提出了一种点云语义分割方法,通过组合优化策略,有效地解决了激光点云覆盖度和丰富度不够的语义分割精度问题。文献[12]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,提出一种基于三维点云数据空间分布特征的多级索引结构,用于点云空间的全局划分。文献[13]提出了一种结合KD树切分思想的类八叉树方法,有效地降低了内存空间占用和邻域搜索耗时。文献[14]提出了一种用于处理点云的网络架构模型,该架构可直接对表示为高维晶格中稀疏样本集的点集合进行操作,同时也可应用于2D-3D推理中的基于点和基于图像的表示结合中获取训练需要得到的模型。文献[15]通过基于可见性滤波的视角追踪点云,实现基于截断最小二乘方程的快速点云配准,实现了高精度点云地图构建与高效三维变化检测。文献[16]提出了一种可以具有无限深度的元结构,以及嵌套结构的自定义、用户定义的组合,使得点云之外的索引可以适应不同类型的数据集,同时考虑数据的空间分布,可以为任何类型的数据集实现合适的空间索引。

KD树索引结构不要求所有的叶子节点都在同一层,能够更加灵活地处理多维和低维的点云数据。但是,在处理海量数据的时候效率不高,考虑到激光点云数据的数据量大的特点,若使用传统的KD树索引结构则生成的树的深度会很大,增加了很多开销。针对3D激光点云数据查询,由于八叉树[17]的局限性,查询时候需要不断地递归执行且随着八叉树的层数逐渐增加时,时间开销会进一步增大。已有的单索引结构具有较大的局限性,并不能很好地对点云数据进行处理,为了弥补已有方法的不足,针对三维激光点云数据,本文基于Z曲线[18]提出了一种新的索引结构,即KD_ZoC索引结构,并进一步提出了有效的近邻查询方法KD_Cnn,并同文献[19]提出的KD-OcTree方法进行了对比。

1 KD_ZoC索引结构的构建

改进的索引结构主要是充分利用KD树和八叉树的索引结构的优点,采用局部KD索引结构和局部八叉树的索引结构形式。

由于激光点云数据坐标为浮点型,因此需将浮点型的数据转化为二进制类型的数据,再进行编码排序,图1给出了八叉树叶节点数据按照Z曲线排序示意图,其中Ⅰ、Ⅱ、Ⅲ、Ⅳ分别表示不同的分区,通过将数据点划分成不同网格分区的方式,利用前缀码来进行邻近点的搜索,p_1、p_2、p_3、p_4、p_5分别表示不同的数据点。

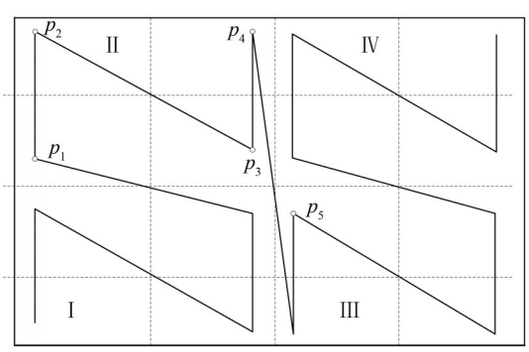


图1 八叉树叶子节点数据按照Z曲线排序示例

Fig. 1 Example of Octopus Leaf Node Data Sorted by Z Curve

该索引结构首先通过局部KD树划分,将激光点云数据划分为多个KD子空间,并在每个子空间中构建八叉树,直到将KD子空间中的数据点集也划分成若干个八叉树。同时为了将KD树

和八叉树进行连接,考虑到Z曲线的优势能够有效地将多维数据转换到低维数据的特点,利用Z曲线的数据划分能够有效地保证局部点的严格相邻,保留了原本三维数据之间的局部特征相似性。将激光点云数据的数据点通过降维存储到局部八叉树的叶子节点中,该索引结构能够有效地解决激光点云数据的近邻查询,索引结构如图2所示,其中圆圈代表不同的数据点,箭头表示具体的执行流程。

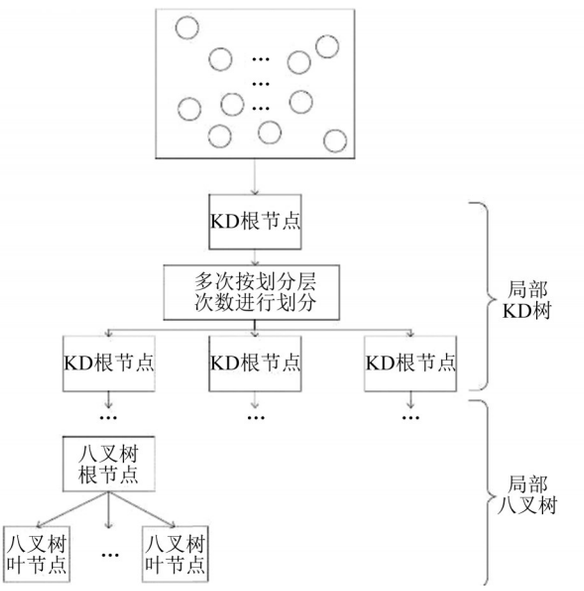


图2　KD_ZoC索引结构构建

Fig. 2　KD_ZoC Index Structure Construction

该索引结构主要分为两个部分,第一部分为局部的KD树,对激光点云数据进行有效的划分,第二部分为局部的八叉树索引结构,所提索引结构利用Z曲线进行数据的处理,因此八叉树的叶子节点中存储的并不是真正的点云数,能够有效地减少时间的开销。此外,针对海量数据也能进一步降低因存储点云数据而消耗的时间和空间存储。考虑到经过Z曲线进行划分的数据得到的结果为二进制的数据,因此只需要根据前缀即可直接定位到所在的上层节点的位置,能进一步对近邻查询的效率进行提升。索引结构中的局部八叉树索引并不建立真正的KD树,只是根据KD树将点云数据进行划分,即只保存局部子空间中的点集。

基于以上论述,所提出的索引结构的具体构建过程如下:

1) 获取激光点云数据中的三维激光点云数据p的3个维度$\{X,Y,Z\}$,并选择维度中方差最大维度$d_{\max}=\max(X_d, Y_d, Z_d)$,其中$X_d$、$Y_d$、$Z_d$满足:

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

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

$$Z_d=\frac{1}{n}\sum_{i=1}^{n}(z_i-\bar{z})^2 \tag{3}$$

2) 选择该维度上的中值点$X_m=\{x_m,y_m,z_m\}$作为局部KD树索引结构的根节点,同时对空间中的平面进行分割;

3) 根据激光点云数据的最大和最小值坐标即$\{X_{\min},Y_{\min},Z_{\min},X_{\max},Y_{\max},Z_{\max}\}$中构建包围激光点云数据的最小外接立方体;

4) 以步骤2)中根节点为轴进一步将立方体划分为两个子空间并保留其包围盒空间,同时子空间被进一步划分,直到达到KD树划分设定的阈值S_{KD}则停止对数据的划分,其中$S_{\mathrm{KD}}=N/2^m$,N为数据点数,m为KD树划分的层数;

5) 获取叶节点表示分割后的n个数据点集Q,完成KD树的数据划分;

6) 将KD树划分后的数据点集作为局部八叉树的根节点,将数据点集中的数据点按照Z曲线进行划分得到Z点集;

7) 依据获得的Z点集将空间划分为8个子立方体,作为局部八叉树的子节点,同时创建一个单独的缓存区域用于保留子空间中的数据Z地址编码;

8) 对八叉树的子空间做进一步的划分直到八叉树中的激光点云数量达到预设八叉树的分割阈值S_{oC},停止划分。如果数据点树大于八叉树分割阈值S_{oC},八叉树被分割成8个子立方体,每个子立方体中点集被分割成8个子立方点,保存其Z地址并存入缓冲区ε中;

9) 记录叶子节点LoC和其Z地址;

10) 返回构建的索引结构。

基于以上论述,下面进一步给出所提的局部八叉树索引构建算法LD_oC。

算法1:局部八叉树索引构建算法LD_oC

输入:激光点云数据P;

输出:含有叶子节点的八叉树。

1	$s_{\mathrm{count}}\leftarrow 0, j\leftarrow 0$;//初始化$s_{\mathrm{count}}$和$j$数值
2	数据点p进入队列
3	读取队列的首元素
4	update s_{count};//更新数据点个数计入s_{count}中
5	if 数据点数$N>S_{\mathrm{KD}}$ then
6	计算第首元素τ所在的分割区,并将其空间集合划分为两个新的子集
7	把两个子集放入队列,并将第一个元素τ从队列中取出放入堆栈中
8	else 将首元素τ出队 then

9	if queue！=empty;//判断队列是否为空,若为空则执行下一步
10	$i \leftarrow i+1$ then
11	返回读取队列的首元素
12	else将堆栈集合中所有数据作为KD分离集合的点集Q输出
13	$n \leftarrow \text{size}(Q)$;//获取数据点集$Q$的大小
14	使用Q_j作为八叉树的根节点,并记录其Z地址
15	递归对八叉树进行拆分保存节点数值,直到节点数$N' > S_{oC}$
16	if $j<n$ then;//判断j和n的大小关系
17	$j \leftarrow j+1$
18	else 返回八叉树叶子节点作为分割的子空间
19	return 含有叶子节点的八叉树索引

算法1中第1～13步表示对于输入的激光点云数据进行KD树的创建,接着对其进行划分并同KD树的阈值S_{KD}进行比较,并不断对三维激光点云数据进行拆分从而得到点集Q,其中点集Q作为局部八叉树的根节点进行进一步的划分,最后按照分割区域进行拆分并将激光点云数据的信息存入到叶子节点中,并记录点云数据的Z地址值,便于后续的检索和查询。

2 基于KD_ZoC索引结构的近邻查询

对点云数据的Z地址进行划分为点云数据的K近邻查询提供了有力的方法,即通过编码对数据点的相邻空间进行搜索,最后在相邻空间中选择K个点作为K近邻查询的基础,即可将激光点云数据的近邻查询转化为基于所提索引结构的近邻查询。

根据点云数据点$p(X,Y,Z)$中的位置和每个KD子空间(KD子空间表示为KD_S)中包围盒的$\{X_{\min},Y_{\min},Z_{\min},X_{\max},Y_{\max},Z_{\max}\}$,根据子树的叶子节点判断数据点$p$所在的子空间。若子空间$KD_S$的包围盒为$\{X_{\min},Y_{\min},Z_{\min},X_{\max},Y_{\max},Z_{\max}\}$,八叉树的最大分割数为$n$,则计算$p(i,o,k)$的索引值,其中$i$、$o$、$k$满足式(4),并对所得数据向下取整分别赋值给$i$、$o$、$k$:

$$\begin{cases} i = X/[(X_{\max}-X_{\min})/2^n] \\ o = Y/[(Y_{\max}-Y_{\min})/2^n] \\ k = Z/[(Z_{\max}-Z_{\min})/2^n] \end{cases} \tag{4}$$

考虑到激光点云数据表示的为向量信息,需要对其进行转化。其中,具体将(i,o,k)二进制数转化的形式如下:

$$\begin{cases} i = i_0 \cdot 2^0 + \cdots + i_\varpi \cdot 2^\varpi + \cdots + i_{n-1} \cdot 2^{n-1} \\ o = o_0 \cdot 2^0 + \cdots + o_\varpi \cdot 2^\varpi + \cdots + o_{n-1} \cdot 2^{n-1} \\ k = k_0 \cdot 2^0 + \cdots + k_\varpi \cdot 2^\varpi + \cdots + k_{n-1} \cdot 2^{n-1} \end{cases} \tag{5}$$

式中,i_ϖ、o_ϖ、$k_\varpi \in \{0,1\}$;$1<\varpi<n$,集合P的线性八叉树节点对应于点集$Q=\{q_1,q_2,\cdots,q_\varpi\}$,同时下式成立:

$$q_\varpi = 3i_\varpi + 2o_\varpi + 2k_\varpi \tag{6}$$

式中,q_ϖ表示最大划分数子空间的编码。如果位于立方体上的由Q进行编码的八叉树为已知的,则也可以根据下式反向寻找其子立方体的空间索引值(i,o,k):

$$\begin{cases} i = \sum_{\varpi=1}^{n-1} (q_\varpi \bmod 2) \cdot 2^\varpi \\ o = \sum_{\varpi=1}^{n-1} [(\frac{1}{2} \cdot q_\varpi) \bmod 2] \cdot 2^\varpi \\ k = \sum_{\varpi=1}^{n-1} [(\frac{1}{4} \cdot q_\varpi) \bmod 2] \cdot 2^\varpi \end{cases} \tag{7}$$

式中,mod表示取余运算。搜索点$|p|$个相邻八叉树子空间构成点集R,若p中的空间子立方体的索引值为(i,o,k),其相邻的子立方体可表示为$\{i\pm\lambda,o\pm\lambda,k\pm\lambda\}$,其中$\lambda\in\{0,1\}$,保留相邻空间中的点构成候选点集$RoC$,根据下式完成距离之间的计算,同时取距离最短的$k$个点作为$p$点的邻近点:

$$D_{i,j} = \sqrt{(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2} \tag{8}$$

该过程中,邻域空间是通过八叉树分割后的点云空间进行搜索的,且其中的分割阈值仍然要小于八叉树分割阈值S_{oC},局部区域中的近邻查询只需要K近邻搜索即可。考虑到相邻空间的划分未知,因此,给出算法2对相邻空间的立方体进行划分。

算法2:局部八叉叶子节点空间划分算法LoC_GS。

输入:叶子节点空间LoC_s和数据点p;

输出:候选点集RoC。

1	查找p的空间编码$Q_p = q_0 q_1 \cdots q_{n-1}$
2	查看p是否在八叉树的叶子节点空间LoC_s中
3	if 叶子节点空间中有$Q_s = Q_p$ then
4	搜索叶子节点子空间$LoC_{s'}$
5	将$LoC_{s'}$添加到候选空间集R'中
6	if LoC_s的编码长度$l(s)<l(p)$且有$l(p)$的前缀码包含$l(s)$ then;//其中$l(s)$、$l(p)$表示编码长度
7	则将LoC_s添加到候选空间集R'中
8	return 候选空间集RoC

算法2中的第1～4步主要判断八叉树中的叶子节点和子空间中的编码关系,5～6步再将得到的二进制编码进行比较和约减,并将前缀码重复的区域进行合并,进一步对数据做约减处理得到候选集。将得到的候选集空间R'中删除其中重复的空间,得到空间集合R作为点p的空间邻集合,最后

对空间集合中的所有点进行K近邻搜索。

进一步利用改进后的索引结构的对称性对数据进行处理进而提高数据的查询速度。设激光点云数据$\rho(p_1,p_2,\cdots,p_N)$,$p_N\in pR^3$,其中ρ表示三维点云集,pR^3表示以查询点p为中心R为半径的圆,其邻域内的点满足条件:

$$N(p,R)=\{p\in\rho|p-q|<r\} \tag{9}$$

式中,q表示近邻查询划分的中心点,该近邻查询将近邻稀疏的数据点按照八叉树的结构对其进行进一步划分,不再需要每次都对其中的距离反复进行计算,若划分后的数据点完全在查询范围内,则对八叉树的子树进行修剪。递归地将八叉树中的区域不断进行划分,直到小于给定的阈值S_{oC},其中对八叉树的每一次地划分表示一个新的八叉树子空间(图3),用八分圆$L(O)$的中心O和边长的1/2表示一个八分圆,对划分后的数据进行近邻计算。

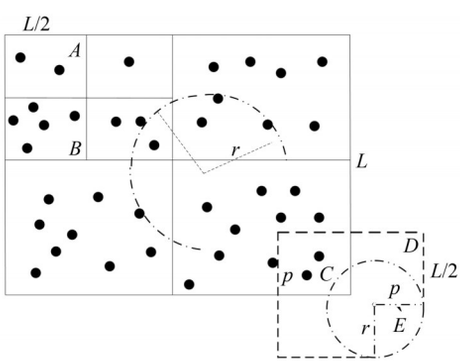


图3　圆域划分

Fig. 3　Circle Division

近邻计算的具体过程如下:

1) 遍历所有的数据点信息;

2) 判断O是否在范围$S(p,R)$中,若是,则执行步骤3),否则,执行步骤4);

3) 将O为圆心的八分圆内的所有点加入候选集中;

4) 判断O是否在以K为中心、r为半径的叶八分圆中,若是,则执行步骤5),否则,执行步骤9);

5) 遍历O中数据点,执行步骤6);

6) 判断是否有$|p-q|<r$成立,若成立,则执行步骤7);

7) 将点p加入到候选集中,并执行步骤8);

8) 返回更新候选集;

9) 遍历O中子八分圆K;

10) 判断$S(p,R)$是否覆盖子八分圆K,若覆盖则执行步骤11),否则,执行步骤8);

11) 返回半径邻域查询结果。

将八分圆域划分中的数据点表示为γ_0,其中$\gamma_0\in\rho$,八分圆域中的数据点被划分为若干互不相交的子集γ_s,满足:

$$\gamma_0\cup\sum_1^n\gamma_s\text{且}\gamma_i\cap\gamma_j=\varnothing \tag{10}$$

式中,$1<i<n$,$1<j<n(i\neq j)$,且$\gamma_s\in\gamma_0$。

建立一个列表用于存储在中心为k、半径为r的八分圆中的点,若经过划分后的八叉圆域包含的数据点数小于阈值,则将该叶子节点信息加入到列表中进行存储。其中,数据点p_i在八分圆域中心O中$p_i\in\rho$,相对应的子八分圆域的圆心为O_c,半径为r_c,其所表示的圆域范围为e_{O_c},则式(11)成立。

$$\max_j|p_i^{(j)}-r_c^{(j)}|<e_{O_c} \tag{11}$$

式中,$p_i^{(j)}$表示p_i的第j个分量。从八叉树的根部开始遍历八叉树,并找出以查询点p和查询半径为r的范围$S(p,r)$,同八分圆的范围的覆盖情况进行判断,即$|p-p'|<R$成立,其中$r\in R$。搜索圆域$S(p,r)$内的所有点也在$N(p,R)$中,因此只需要针对重叠的叶八分圆内区域中的数据进行对比即可,可以有效地约减对查询不相关的子集的计算,减少时间开销。

为了更快地提高数据的检索效率。根据数据的对称性,将查询点p变换到八分圆的局部坐标中,使$p'(j)=|p(j)-O(j)|$成立,从而得到变化后的数据点p',若变换后的邻域圆$S(p',r)$的中心点p'位于图3中的虚线区域,则称其同八分圆重叠。若重叠区域满足:

$$\max p'^{(j)}<(1/2)\cdot L+r \tag{12}$$

$$\min p'^{(j)}<(1/2)\cdot L \tag{13}$$

$$\|p'-(1/2)\cdot L\|<r \tag{14}$$

若满足式(12)~(14)中的任意一个,即可认为其变换后的邻域圆和八分圆相交。式(12)表示图3中的区域E,式(13)表示图3中的区域D。当数据点都位于图3中的C区域时,由式(11)可知八分圆内的点此时一定在搜索圆域$S(p,r)$内,即可停止遍历子树中的八叉树。因此,可将图3区域C中的全部数据添加到候选结果集R'中,即可停止在该子树中对数据的遍历。由于八分圆中的数据都存储在叶八分圆中,因此可以将八分点存储在一个邻接表中来对数据进行存储,方便对近邻数据进行查询。此外,当数据点完全在圆域内时,可以有效地避免对于距离的不断计算,能够有效地节约大量时间开销。

由于每一次对八分圆进行划分时都需要保证每次划分中的数据不相交,考虑到重新构建局部八叉树时可以任意地对其子集进行排列,为此,将每个八分点都变成有序的,然后将数据点映射到八分圆内的下一个点或者是子八分圆的第一个数据点,并通过一个单链表结合邻接表的形式表示其中的后继关系。存储了数据的半径、邻域圆的中心和对应的查询范围S、内部的第一个节点索引O_{ccf}和最后一个索引O_{cla},此外,还存储包括叶子节点中所在的Z地址数值和其内部所含有的数据点数N',从第一个索引值向后访问直到访问,到最后一个节点值O_{ccf},向后访问其他的数据点,直到访问的数据点数为$N'-1$时终止搜索。

基于以上论述,进一步给出了基于所提索引结构的近邻查询算法。

算法3:编码地址近邻查询算法KD_Cnn。

输入:八分圆中心O、范围S、起始索引节点O_{ccf}、结束索引节点O_{cla}、八叉树数据点数m_{ocd};

输出:八分圆$L(O)$中包含的近邻查询结果。

1	初始化八分圆O、S、O_{ccf}、O_{cla}、S_k(起始索引)、t_k(结束索引)、m_k(第k个子节点的点数)、C_k(子中心)
2	$S_{ck}\leftarrow\frac{1}{2}S$//其中$S$表示其范围
3	if $m_{\mathrm{ocd}}>b$ then
4	$i\leftarrow O_{\mathrm{ccf}}$,$j\leftarrow 0$
5	if $m_k=0$
6	$S_k\leftarrow i$
7	else if $m_k>0$ then
8	$\mathrm{succ}(t_k)\leftarrow i$
9	update $t_k\leftarrow i$ and $m_k\leftarrow m_k+1$
10	$i\leftarrow\mathrm{succ}(i)$;//获取一个索引$i$的数值
11	$j\leftarrow j+1$
12	设置ϕ为最后一个Z地址编码
13	$k\leftarrow m_k$;//遍历m_k直到满足$m_k>0$赋值给k
14	根据C_k、S_{ck}、t_k、m_{ocd}和S_k构建八分圆并赋值给C_k
15	用C_k的新的开始和结束更新S_k和t_k
16	if C_k是第一个子节点 then
17	$O_{\mathrm{ccf}}\leftarrow S_k$; //更新八分圆的第一个节点
18	else $\mathrm{succ}(t_\phi)\leftarrow S_k$
19	更新叶子节点$O_{\mathrm{cla}}\leftarrow t_k$
20	设置$\phi\leftarrow k$
21	return $L(O)$
22	return $L(O)$中的近邻查询结果

算法3中的第6~12步使用Z地址将点细分为子集,以索引子八分圆并更新后继点,以便可以用于初始化子八分圆。由于后继关系的变化,这可能改变子八分圆的开始和结束索引,因此,必须更新算法3中的第16和20步的开始和结束索引,使其指向后继中的正确位置。最后通过查询邻接表中的叶八分的邻接表中数据即可找到近邻k查询的数据点。

3 实验对比分析

为了评估所提方法的性能,本文设计了两部分实验,即索引构建性能的对比实验和近邻查询的性能对比实验。本文选取的数据集为RGB-D Object Dataset中点云数据(http://rgbd-dataset.cs.washington.edu/index.html)和大型自然场景的海量三维点云数据(http://graphics.stanford.edu/data/3Dscanrep/)。数据点的数量级为10^3~10^8。分别将本文提出的KD_Cnn方法同KD树方法、八叉树方法和文献[19]提出的KD-OcTree方法进行对比。

实验环境为Microsoft Windows 11,Core(TM)i7-12700H 2.30 GHz处理器,运行内存为16 GB。

3.1 索引构建时间对比实验

本节分别就KD树索引结构、八叉树、KD-OcTree和KD_Cnn这4种不同的索引结构在全局KD叶子节点的阈值点数为2 000时的索引结构构建时间进行比较,见表1。

表1 构建索引时间对比

Table 1 Comparison of Index Construction Time

点云数量/个	索引时间/s			
	KD树	八叉树	KD-OcTree	KD_Cnn
3 942	0.48	0.04	0.08	0.05
15 643	3.88	0.21	0.46	0.34
421 865	34.66	3.89	5.17	4.02

从表1可知,相较于KD树索引构建的时间,其他3种方法索引时间都较少,且随着点云数量的增加,KD树构建索引的时间开销愈加明显。由于KD_Cnn方法需要为后续查询创建链表,因此在构建索引结构时时间开销有所增加,而KD-OcTree方法中由于测试大量的地面点,考虑到并不是所有的激光点云数据都是均匀分布的,因此,构建索引的时间开销也有所增加。

3.2 激光点云数据量查询的对比实验

本节对比了KD树、八叉树、KD_OcTree和本文方法KD_Cnn在不同数据量下CPU运行时间的关系。

本实验中,近邻查询的时间评价指标表示为找到指定点云数数据的最近邻点所消耗的平均时间开销(图4)。八叉树消耗的时间最大,查询的时间

效率相较于其他3种方法的时间开销较大。此外,由于KD树索引往往适用于单点近邻搜索,因此若数据点进一步增加时,时间开销将进一步增加。KD-OcTree方法基于以块为单位的设计,相对于八叉树和KD树的查询效率要高,而本文的方法KD_Cnn中采用了基于八分圆的数据约减方式,且由于数据存储在叶子节点中的数据仅为其地址值,因此根据其所划分的Z前缀码即可快速查找所需的近邻查询点,时间开销相对较小。

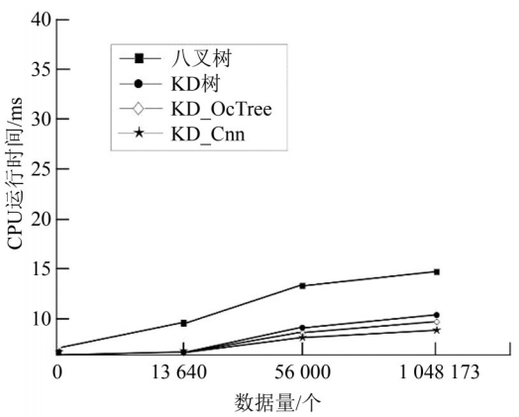


图4　数据量和CPU运行时间的关系

Fig. 4　Reltionship Between Data Volume and CPU Running Time

3.3　不同阈值下构建索引的运行时间开销

图5展示了不同阈值下针对不同数据点数的CPU运行时间。针对不同的数据点数,分别对阈值为2 000、4 000、6 000、8 000和10 000的情况进行对比。通过实验可知点云数量越大,受阈值的影响越大。因此,选用合理化的索引机构能够有效地针对不同阈值的设定,进一步节省时间开销。

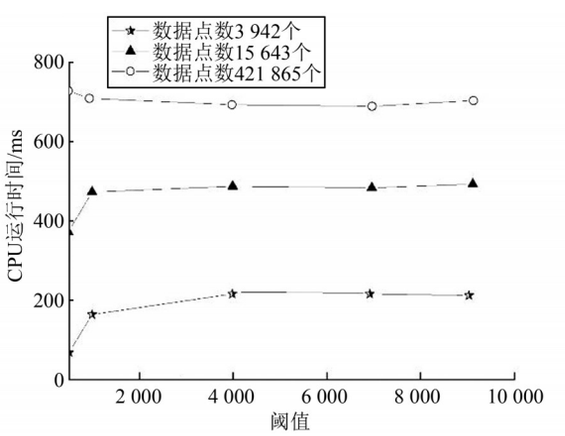


图5　不同阈值下不同数据点数运行时间

Fig. 5　Running Time of Different Data Points Under Different Thresholds

3.4　邻域半径对查询效率的对比

图6展示了3种方法在搜索半径变化时的搜索时间。随着邻域半径的变化,修剪策略的优势越大,这是因为八分圆更可能完全在搜索圆域内,进行比较时,只需进入到对应分区中查找数据即可,不必访问点云数据;而传统的八叉树的方法必须对叶子节点进行处理,并反复将所包含的点与查询点进行对比,因此时间开销会显著增加。

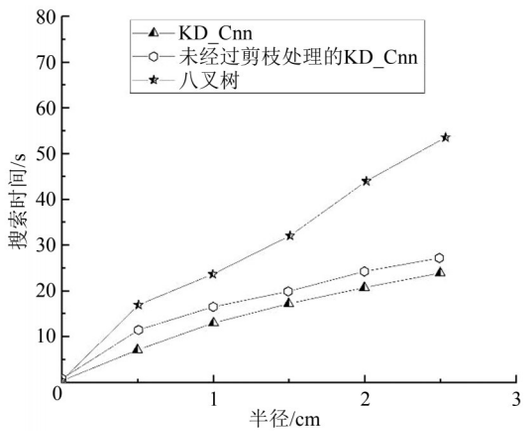


图6　搜索半径对搜索时间的影响

Fig. 6　Effect of Search Radius on Search Time

4　结　语

本文研究了基于三维激光点云数据的索引结构设计和近邻查询问题。基于Z曲线,首先对三维激光点云数据进行降维处理,并对其进行分区,接着构建局部KD树和局部八叉树来获取叶子节点的地址信息,最后针对叶结构构建八分圆域对其进行近邻查询。实验结果表明,本文提出的方法具有较好的性能。对于障碍环境下的激光点云数据信息复杂查询问题,本文所提出的KD_Cnn方法具有一定的局限性,因此,在未来的研究中,拟引入分布式索引的技术以提升查询的性能,进一步将研究方法应用于三维地形智能提取领域[20]。

参　考　文　献

[1]　孙殿柱, 刘健, 李延瑞, 等. 三维散乱点云的Voronoi拓扑近邻点集查询算法[J]. 武汉大学学报(信息科学版), 2011, 36(1): 86-91.
SUN Dianzhu, LIU Jian, LI Yanrui, et al. An Algorithm Inquiring Voronoi Topological Neighbors for 3D Scattered Point-Cloud[J]. *Geomatics and Information Science of Wuhan University*, 2011, 36(1): 86-91.

[2]　张卉冉, 董震, 杨必胜, 等. 点云压缩研究进展与趋势[J]. 武汉大学学报(信息科学版), 2023, 48(2): 192-205.
ZHANG Huiran, DONG Zhen, YANG Bisheng, et al. Progress and Perspectives of Point Cloud Com-

pression[J]. *Geomatics and Information Science of Wuhan University*, 2023, 48(2): 192-205.

[3] KAMENCAY P, SINKO M, HUDEC R, et al. Improved Feature Point Algorithm for 3D Point Cloud Registration [C]//2019 42nd International Conference on Telecommunications and Signal Processing (TSP), Budapest, Hungary, 2019.

[4] GOJCIC Z, ZHOU C F, WEGNER J D, et al. The Perfect Match: 3D Point Cloud Matching with Smoothed Densities[C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019.

[5] AOKI Y, GOFORTH H, SRIVATSAN R A, et al. PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet[C]//2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019.

[6] 伍济钢, 马佳康, 杨康, 等. 基于改进ICP的复杂机械零件测量点云配准方法[J]. 光电子·激光, 2023, 34(6): 620-627.
WU Jigang, MA Jiakang, YANG Kang, et al. Measurement Point Cloud Registration Method for Complex Mechanical Parts Based on Improved ICP [J]. *Journal of Optoelectronics·Laser*, 2023, 34(6): 620-627.

[7] LI M J, YIN D. A Fast Segmentation Method of Sparse Point Clouds[C]//2017 29th Chinese Control and Decision Conference (CCDC), Chongqing, China, 2017.

[8] JIANG H B, JIAO Q S. TLS Point Cloud Segmentation Based on Points Features[C]//2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, USA, 2017.

[9] TCHAPMI L, CHOY C, ARMENI I, et al. SEGCloud: Semantic Segmentation of 3D Point Clouds [C]//2017 International Conference on 3D Vision (3DV), Qingdao, China, 2017.

[10] ROVERI R, RAHMANN L, ÖZTIRELI A C, et al. A Network Architecture for Point Cloud Classification via Automatic Depth Images Generation[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018.

[11] 王渊. 车载激光点云数据增强与语义分割[J]. 武汉大学学报(信息科学版), 2024, 49(10): 1973.
WANG Yuan. Data Enhancement and Semantic Segmentation of Mobile Laser Scanning Point Cloud [J]. *Geomatics and Information Science of Wuhan University*, 2024, 49(10): 1973.

[12] 杨丽娟, 崔钰琳, 杨紫骞, 等. 基于点云空间分布特征的多级索引结构[J]. 激光与红外, 2023, 53(1): 137-145.
YANG Lijuan, CUI Yulin, YANG Ziqian, et al. Multi-Level Index Structure Based on Spatial Distribution Characteristics of Point Cloud[J]. *Laser & Infrared*, 2023, 53(1): 137-145.

[13] MIN L, YUN M. Study on Point Cloud Management Strategy Based on Octree-Like Index[J]. *Laser & Optoelectronics Progress*, 2020, 57(14): 141025.

[14] SU H, JAMPANI V, SUN D Q, et al. SPLATNet: Sparse Lattice Networks for Point Cloud Processing[C]//2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 2018.

[15] 丛阳滋. 联合多源点云的激光雷达三维建图与更新[J]. 武汉大学学报(信息科学版), 2024, 49(10): 1972.
CONG Yangzi. LiDAR-Based 3D Mapping and Update Leveraging Heterogeneous Point Clouds [J]. *Geomatics and Information Science of Wuhan University*, 2024, 49(10): 1972.

[16] OGAYAR-ANGUITA C J, LÓPEZ-RUIZ A, RUEDA-RUIZ A J, et al. Nested Spatial Data Structures for Optimal Indexing of LiDAR Data[J]. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2023, 195: 287-297.

[17] WANG Z C, WAN S, WEI L. Local Geometry-Based Intra Prediction for Octree-Structured Geometry Coding of Point Clouds[J]. *IEEE Transactions on Circuits and Systems for Video Technology*, 2023, 33(2): 886-896.

[18] ZHANG D F, WANG Y, LIU Z L, et al. Improving NoSQL Storage Schema Based on Z-Curve for Spatial Vector Data[J]. *IEEE Access*, 2019, 7: 78817-78829.

[19] 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.
ZHANG Rui, LI Guangyun, WANG Li, et al. A New Method of Hybrid Index for Mobile LiDAR Point Cloud Data[J]. *Geomatics and Information Science of Wuhan University*, 2018, 43(7): 993-999.

[20] 郑志. 基于亚米级立体卫星影像的三维地形智能提取[J]. 武汉大学学报(信息科学版), 2025, 50(3): 616.
ZHENG Zhi. Intelligent Extraction of 3D Terrain Based on Sub-Meter Satellite Stereos[J]. *Geomatics and Information Science of Wuhan University*, 2025, 50(3): 616.