引 用 格 式:Zeng Xiangzhao, Li Chuanrong, Zhang Zheng, *et al*. Research on Superimposition Pyramid Index Applied to Massive Points Cloud Fast Display[J]. Remote Sensing Technology and Application, 2015, 30(3): 534-539.
[曾祥钊,李传荣,张正,等. 面向海量点云快速显示的叠加型金字塔索引结构研究[J]. 遥感技术与应用,2015,30(3):534-539.]
doi:10.11873/j.issn.1004-0323.2015.3.0534

面向海量点云快速显示的叠加型金字塔索引结构研究

曾祥钊[1,2],李传荣[1],张 正[1],周 梅[1]
(1. 中国科学院光电研究院定量遥感信息技术重点实验室,北京 100094;
2. 中国科学院大学,北京 100049)

摘要:针对激光雷达三维点云数据量大,当计算机内存有限时进行点云读取与处理存在严重滞后的问题,提出了一种叠加型金字塔索引结构。首先,采用一种基于点云最小外包络的不均匀分块策略,将点云数据划分成若干独立的数据块;待分块完成后,利用提出的叠加型索引结构对每个分块构建金字塔;最后,将生成的金字塔按照指定的文件结构存储,生成索引文件。利用机载实测点云数据开展了验证实验,结果表明:该算法有效地降低了索引文件占据的计算机空间资源,实现了海量三维点云数据在有限内存空间的快速显示。

关 键 词:激光雷达点云;三维显示;金字塔
中图分类号:TP 75 **文献标志码**:A **文章编号**:1004-0323(2015)03-0534-06

1 引 言

激光雷达能够快速直接地获取地物的三维信息,为资源勘探、城市规划、土地利用、防震减灾等应用提供了大量的地理信息,相关技术研究与应用得到了广泛的关注。为了能直观提取并利用点云数据包含的三维信息,点云可视化是一种有效的辅助途径。然而,随着激光雷达硬件技术不断进步,激光扫描设备的精度与频率越来越高,获取点云的数据量剧增,有限的计算机内存资源无疑成为了制约海量点云快速可视化的一个瓶颈。

目前,实现海量三维点云快速显示的有效方法是构建索引树,常用的索引树主要包括 R 树[1]、K-D 树[2]、四叉树[3]和八叉树等。其中,R 树具有较强的灵活性和调节性,但由于中间节点允许重叠,在查找速度、动态操作性能方面存在一定的提升空间;四叉树和八叉树结构简单,对于精确匹配点的查找性能较高,但树的动态性较差,树的平衡性不好,导致视口裁剪性能下降,影响点云显示的交互性。针对上述问题,国内外研究者相继提出了一系列改进的方法。Beckmann 等[4]在 R 树的基础上提出了 R* 树,相比于 R 树,R* 树的优越性体现在查找方式和节点操作的多样性,并且它同时支持点和空间数据的索引,但是它的索引构建时间比 R 树略高。Rusinkiewicz 等[5]提出的 QSplat 渲染系统采用和八叉树类似的建树方式,该系统将球体作为点云的外包络,对点云进行划分,然后以包围球体为节点单位构建索引树,树节点最终以广度优先的原则进行存储,该索引方法灵活性好,并且采用了广度优先的方式,树深相同的节点代表的分辨率相同,因此可以

收稿日期:2014-01-21;**修订日期**:2015-04-11
基金项目:国家重大专项项目"激光雷达数据处理技术",中国科学院光电研究院创新项目"面向精细分类需求的全波形激光雷达数据处理与应用技术研究"(Y12403A01Y)。
作者简介:曾祥钊(1989—),男,河南郑州人,硕士研究生,主要从事激光雷达技术应用方面的研究。E-mail:xiangzhao89@126.com。
通讯作者:李传荣(1956—),男,湖南醴陵人,研究员,主要从事遥感卫星地面应用系统研究。E-mail:crli@aoe.ac.cn。

更灵活地控制显示分辨率。支晓栋等[6]提出的一种改进四叉树算法可以快速完成索引树的构建,但是该算法构建的索引树的树高较小,降低了数据的查询速度。龚俊等[7]采用三维 R 树索引实现视锥体查询,在保证场景逼真度的前提下剔除了次要目标,保留了重要特征,但查询时间在视锥体计算和更新查询结果集合方面耗费过多,导致查询速度较慢。总的来讲,近年来随着点云数据量的大幅增加,采用单一的索引树虽然构建速度快,但结构简单,导致索引不够充分,在查询速度方面无法满足点云显示的应用需求,因此,学者们相继提出将这些索引结构组合应用,建立两级复合索引结构的思想。路明月等[8]采用的"规则空间八叉树+平衡二叉树"的索引结构可以实现海量点云数据的快速检索和显示,但在构建索引时需要对三维坐标多次比较,因此索引的初始化速度较慢。赵楠[9]提出的基于网格与 R 树的混合索引结构加快了点云的索引速度,但索引空间的开销较大。通过上述分析可以看出,虽然复合索引提高了点云的检索效率,实现了海量数据的快速查询,但它需要占用大量的计算机空间资源并耗费更多的索引构建时间,仍难以有效满足海量点云快速显示的需求。

针对以上问题,本文提出了一种叠加型金字塔索引结构,该索引结构可有效地降低索引文件的存储开销,同时提高点云读取速度,在计算机内存有限的情况下实现海量点云的快速显示。构建索引文件的流程如图 1 所示,首先,针对点云数据分布的不规则性,采用一种基于点云最小外包络的不均匀分块方式,获得大小相近的 N 个点云块,以保证后续金字塔建立的一致性;同时由于是点云的最小外包络,在显示点云时,筛选视口内的分块就更加准确,可提高显示效率。然后,对获得的 N 个点云块,分别建立叠加型金字塔,最终生成索引文件。本文提出的叠加型金字塔结构通过引入金字塔各层数据之间的关联性,在有效降低索引文件对计算机存储空间需求的同时,可极大地提高点云数据的读取效率。

2　原理和方法

2.1　金字塔结构

金字塔是典型的分层数据结构形式,图像金字塔是一个图像序列,其中,底层是具有原始影像分辨率的图像,随着金字塔层数的升高,图像的分辨率逐渐降低[10]。同样,对于海量点云,采用金字塔结构可以充分利用有限的内存,实现点云的层次细节(LOD,Level of Detail)[11]显示效果。当视点距离点云较远时,采用较小层次细节显示,随着视点距离拉近,逐步增加点云的数量,提高点云的层次细节。目前,金字塔结构主要针对二维平面图像,针对三维点云的研究相对较少,传统的金字塔结构各层数据存在冗余,生成的索引文件往往比原始文件还要大,因此,将传统的金字塔结构引入海量三维点云的索引建立会增加计算机的存储空间;同时这种金字塔各层点云的冗余性,提高了点云的读取量,增加了点云调入内存的时间,降低了点云的显示速度。本文提出的叠加型金字塔结构,去除了金字塔各层间数据的冗余性,降低了硬盘的存储空间,同时又减小了金字塔各层读入内存的时间,总体提高了点云的显示速度。

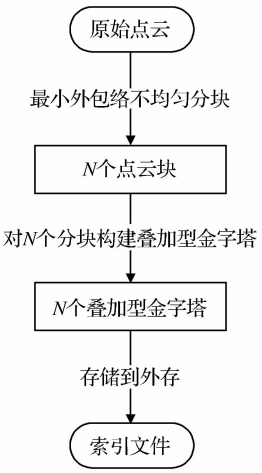


图 1　索引建立流程

Fig. 1　The flow chart of index construction

2.2　叠加型金字塔结构

图 2 表示地物的 64 个激光雷达扫描点,对这些点云建立一个 3 层的金字塔数据结构,方形点为金字塔第一层的数据,圆形点为金字塔第二层的数据,其余的三角形点为金字塔第三层数据。由此可知,第一层中有 4 个点云数据,占总点云的 1/16;第一、二层总共有 16 个点云数据,占总点云的 1/4;第一、二、三层包括了所有的 64 个点云数据。

图 2　激光雷达扫描点

Fig. 2　LiDAR scan points

依此类推，对于一个点云文件，可以建立一个 n 层金字塔结构，第一层金字塔数据表示点云的 $1/4^{(n-1)}$，前两层金字塔数据表示点云的 $1/4^{(n-2)}$，前 $n-1$ 层数据表示点云的 1/4，所有 n 层数据表示全部点云。

为了确定金字塔的层数，假设刚打开索引文件时金字塔顶层点云占用内存 50% 的存储空间。由于顶层点云占总点云的 $1/4^{(n-1)}$，可以得出公式：

$$\frac{1}{4^{n-1}} \times \text{Size} \times \text{Count} = \frac{1}{2} \times \text{Memory\_size} \quad (1)$$

其中：Size 表示单个点云所占内存的大小，Count 表示总点数，Memory_size 表示内存大小，n 表示金字塔的层数。根据式(1) 可以计算得到金字塔的最优层数 n。

$$n = \left[\left| \log_4 \frac{2 \times \text{Size} \times \text{Count}}{\text{Memory\_size}} \right| \right] + 1 \quad (2)$$

其中：|| 表示取绝对值，[] 表示向下取整。因此，构建叠加型金字塔索引，可以采取如图 3 所示的构建流程，首先对点云进行分块处理，然后根据式(2) 确定金字塔层数 n，最后逐一对各分块的点云构建叠加型金字塔，具体构建方式如下：① 对点云数据进行 4 阶采样，将采样后剩下的数据作为金字塔的第 1 层；② 对上一步采样得到的数据再进行 4 阶采样，采样后剩下的数据作为金字塔的第 2 层；③ 重复过程 2，直到生成金字塔的第 n 层为止。

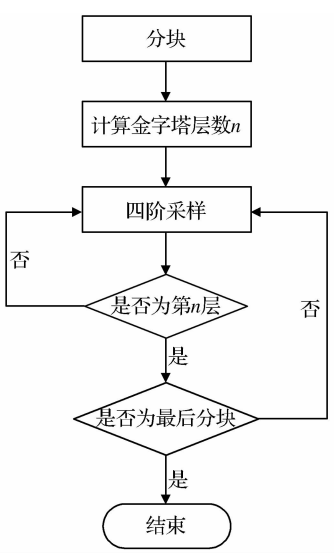


图 3 金字塔构建流程图

Fig. 3 The flowchart of pyramid construction

和传统的点云金字塔相比，本文提出的叠加型金字塔每层的点云量小，将金字塔影像调入内存的速度快。

2.3 点云最小外包络的分块方式

由于点云的海量性，以及计算机内存有限的原因，直接构建叠加型金字塔会降低显示的效率。分块处理策略被视为提高显示效率的有效途径，针对上节的叠加型金字塔结构，本节介绍一种点云最小外包络的分块方式，通过对海量点云进行分块，提高了金字塔的构建速度和显示效率。首先，确定点云的包络立方体，由于其在 x、y 方向上的覆盖范围要远远大于 z 方向上的范围，仅在 x、y 方向上分块。根据图 4 所示的流程，将点云的包络立方体在 $x-y$ 平面上平均分割成 4 份，根据式(3) 和式(4) 确定 4 个矩形各自的顶点坐标。

$$XTile = X\min + \left(\frac{X\max - X\min}{2^n} \right) \times N \quad (3)$$

$$YTile = Y\min + \left(\frac{Y\max - Y\min}{2^n} \right) \times N \quad (4)$$

其中：$N = 0,1,\cdots 2^n$ 表示点云块的个数，XTile 和 YTile 分别表示分块在 x 和 y 方向上的顶点坐标值，Xmin、Xmax、Ymin、Ymax 分别表示起始包络立方体在 x 和 y 方向上的最大最小值，n 表示划分的次数，为防止循环次数过多，设定 n 最大取 4。

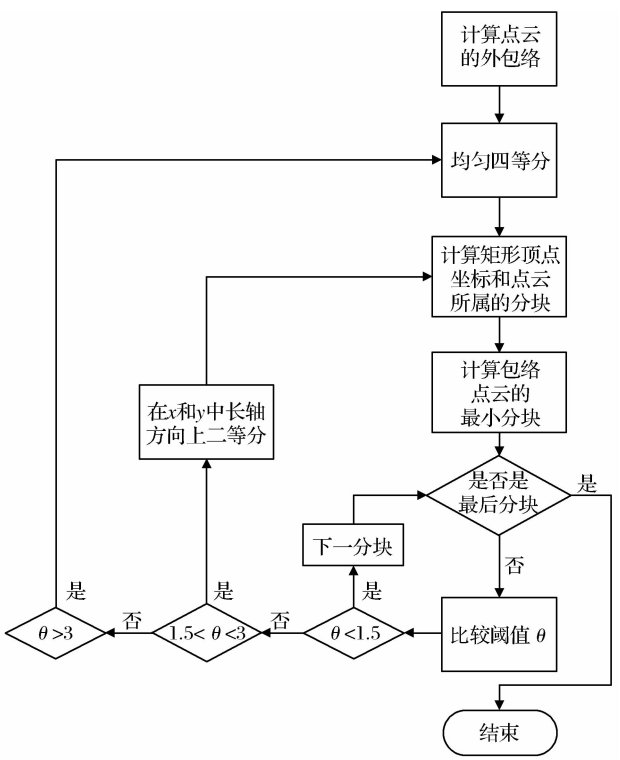


图 4 点云的分块流程

Fig. 4 The flowchart of point cloud division

根据分割的 4 个矩形各自的顶点坐标，判断点云数据属于哪个分块，在判断的同时记录坐标点在

x、y 轴上的最大和最小值,以确定点云数据所在分块的最小外包络。为了尽量保证各分块内的点云数据量相近,以满足对各分块操作效果的一致性,选取其中最小的点云数据量作为阈值,定义分块的点云数据量和阈值的比值为 θ,如果 θ 小于 1.5,不用继续分块;如果 θ 在 1.5 和 3 之间,比较此分块 x、y 方向上的边长,在边长较大的方向上对分块进行二等分;如果 θ 大于 3,对分块进行四等分。重复这一过程,直到所有分块的 θ 都小于 1.5。

最后,按照 2.1 节中叠加型金字塔的生成步骤,对各分块构建叠加型金字塔,生成一个索引文件。读取索引文件的点云数据,根据视口范围和缩放比例决定是否将分块读入内存以及是否释放分块所占用的内存。读取点云时,每层的点云单独读入内存,这样如果需要增加显示的层次细节,只需将点云叠加并显示,如果需要减少显示的层次细节或释放一定内存,将底层的点云所占内存释放即可。根据各分块点云的地理范围和视景体范围,判断点云块和视景体的空间关系,如果该块在视景体内,则将相应的点云层读入内存中;如果该点云块和视景体相交,同样将相应的点云层读入内存中;如果该块在视景体外,则无需读取其中的点云数据。

3　试验与结果分析

本文采用的点云文件为 las 格式,大小分别为 1.88 和 2.53 GB。实验平台参数如下:中央处理器为 Intel Pentium IV,主频 2.4 GHz,系统内存 2 G,图形处理器 NVIDIA Geforce4 M X400。

表 1 对传统金字塔和叠加型金字塔构建 1.88 GB点云数据生成的索引文件进行比较。由此表可知,两种金字塔索引的构建速度都相对较慢,但是一旦完成索引的构建,都可以快速显示点云数据,但叠加型金字塔生成的索引文件要远远小于传统金字塔的索引文件,这是由于传统金字塔层间的点云冗余,增加点云的读取量,降低金字塔的构建速度。

表 1　索引文件对比

Table 1　The comparison between index file

金字塔	性能			
	原文件大小/GB	索引文件大小/GB	索引构建时间/s	文件打开时间/s
叠加型金字塔	1.88	1.34	243	0.8
传统金字塔	1.88	2.60	251	1.2

对于 1.88 GB 的点云数据,表 2 和表 3 统计了显示的点云数据量和视口内分块数目对应的三维漫游速度,可以看出虽然叠加型金字塔的交互速度比传统金字塔的交互速度要快,但是并不明显。

表 2　叠加型金字塔的交互速度

Table 2　The display speed by superimposition pyramid

显示的点数	视口内分块数	漫游速度/s
4 357 032	1	0.8
7 467 825	3	1.0
19 327 641	7	1.8
39 807 368	9	2.2
48 908 580	10	2.8

表 3　传统金字塔的交互速度

Table 3　The display speed by conventional pyramid

显示的点数	视口内分块数	漫游速度/s
4 038 952	1	1.0
7 437 028	3	1.6
19 523 601	7	2.3
39 805 362	9	3.1
48 908 580	10	3.7

表 4 对传统金字塔和叠加型金字塔构建 2.53 GB点云生成的索引文件进行比较。由此表可知叠加型金字塔的索引构建时间比传统金字塔的索引构建时间短,并且由于传统金字塔的顶层点云过多,导致内存不足,无法显示点云。因此随着点云的增多,叠加型金字塔的优势逐渐凸显。

表 4　索引文件对比

Table 4　The comparison between index file

金字塔	性能			
	原文件大小/GB	索引文件大小/GB	索引构建时间/s	文件打开时间/s
叠加型金字塔	2.53	2.12	482	1.2
传统金字塔	2.53	4.62	589	无法打开

表 5 统计了由 2.53 GB 点云数据构建的叠加型金字塔的交互速度,对比表 2 和表 3 可以看出三维漫游速度取决于当前显示的点云数,点云过多,细节信息过多导致点云的绘制速度下降,进而影响三维漫游速度。同时视口内的分块数也影响点云的漫游速度,当视口内只有一个分块时,不需要继续从外存读入点云,因此漫游速度较快。

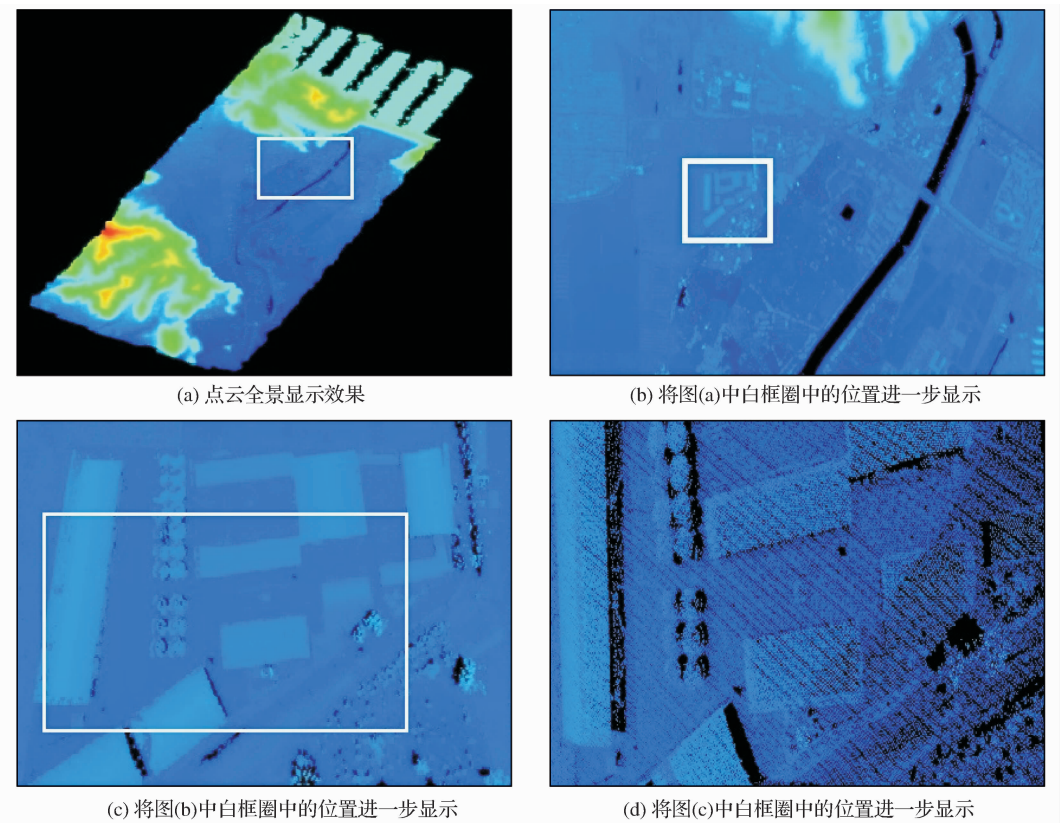
表 5 叠加型金字塔交互速度

Table 5 The speed of point cloud display by superimposition pyramid from 2.53 GB data

显示的点数	视口内分块数	漫游速度/s
4 959 052	1	1.5
8 536 239	5	2.3
35 603 566	9	4.2
50 838 560	13	5.3
50 908 320	14	6.7

结合表 1 和表 4 可知，对于叠加型金字塔，虽然构建速度较慢，但构建完毕后，索引文件的打开速度很快，并且索引文件占据的存储空间较小。对比表 5和表 2 可知，随着点云数据量的增加，漫游速度明显下降，这是由于实验平台的 2G 内存无法很好地满足点云显示的需求。

点云的显示效果如图 5 所示，其中图 5(a)是点云的整体显示，图 5(b)是观察点拉近后图 5(a)方框中的点云显示，随着观察点继续拉近，图 5(b)中白框圈中区域的细节如图 5(c)所示，继续拉近观察点，图 5(c)方框中的进一步细节如图 5(d)所示。由图 5 可以看出在不同视距下，地面建筑物、河流、树木的轮廓、形状和阴影等信息显示清楚。



(a) 点云全景显示效果 (b) 将图(a)中白框圈中的位置进一步显示

(c) 将图(b)中白框圈中的位置进一步显示 (d) 将图(c)中白框圈中的位置进一步显示

图 5 点云显示效果

Fig. 5 The display of point clouds

4 结 语

针对激光雷达三维点云数据量大，计算机内存有限时进行点云读取与处理存在严重滞后的问题，本文在原有金字塔结构的基础上，提出了一种叠加型金字塔索引构建算法，可以有效提高内存利用率，对提高三维点云显示、点云滤波和点云分类等应用的效率具有重要意义。该算法首先对点云进行不规则分块，实现对各分块的单独控制，以充分利用内存；在此基础上，分别对分块内的点云构建叠加型金字塔，实现点云的层次细节显示效果。该索引算法有以下优点：①实现了层次细节显示，提高了内存的利用率；②去除了金字塔每层数据的冗余性，进而降低了金字塔每层的点云数据量，逐层叠加显示点云数据，提高了点云显示的交互速度；③降低了索引文件的大小，节约计算机的存储空间。由实验结果可知，该算法实现了海量三维点云的快速显示，可以达到流畅的漫游速度，并且金字塔读取速度快，索引文件占用内存较小。然而由于采用不规则分块，需要求取点云的最小外包络，在一定程度上影响了金字塔的构建速度，此外三

维漫游速度还有提高的空间。

参考文献(References):

[1] Guttman A. R-Trees: A Dynamic Index Structure for Spatial Searching[M]. San Francisco, California, USA: Association for Computing Machinery (ACM), 1984, 14(2): 47-57.

[2] Ooi B C, McDonell K J, Sacks-Davis R. Spatial Kd-tree: An Indexing Mechanism for Spatial Databases[C]//IEEE International Computer Software and Applications Conference, 1987, 87(10): 85.

[3] Samet H. The Quadtree and Related Hierarchical Data Structures [J]. ACM Computing Surveys (CSUR), 1984, 16(2): 187-260.

[4] Beckmann N, Kriegel H P, Schneider R, *et al*. The R*-tree: An Efficient and Robust Access Method for Points and Rectangles[J]. International Conference on Management of Data, 1990, 19(2): 322-331.

[5] Rusinkiewicz S, Levoy M. QSplat: A Multiresolution Point Rendering System for Large Meshes[C]//Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. Association for Computing Machinery (ACM) Press/Addison-Wesley Publishing Company, San Francisco, California, USA, 2000: 343-352.

[6] Zhi Xiaodong, Lin Zongjian, Su Guozhong, *et al*. Research on Organization of Airboren LiDAR Points Cloud based on Improved Quadtree Algorithm[J]. Computer Engineering and Applications, 2010, 46(9): 71-74. [支晓栋, 林宗坚, 苏国中, 等. 基于改进四叉树的 LiDAR 点云数据组织研究[J]. 计算机工程与应用, 2010, 46(9): 71-74.]

[7] Gong Jun, Xie Xiao. Three Dimension Visualization Query Method based on R2 Tree[J]. Geomatics and Information Science of Wuhan University, 2011, 36(10): 1140-1143, 1153. [龚俊, 谢潇. 基于 R 树索引的三维可视化查询方法[J]. 武汉大学学报(信息科学版), 2011, 36(10): 1140-1143, 1153.]

[8] Lu Mingyue, He Yongjian. Organization and Indexing Method for 3D Points Cloud Data[J]. Geo-information Science, 2008, 10(2): 190-194. [路明月, 何永健. 三维海量点云数据的组织与索引方法[J]. 地球信息科学, 2008, 10(2): 190-194.]

[9] Zhao Nan. A Hybrid Structure of Spatial Multilevel Index based on Grids and R-tree[J]. Computer Technology and Development, 2009, 19(3): 91-94. [赵楠. 一种基于网格与 R 树的多级混合索引[J]. 计算机技术与发展, 2009, 19(3): 91-94.]

[10] Liu Yi, Chen Luo, Jing Ning, *et al*. Parallel Batch-building Remote Sensing Images Tile Pyramid with Map Reduce[J]. Geomatics and Information Science of Wuhan University, 2013, 38(3): 278-282. [刘义, 陈荦, 景宁, 等. 利用 MapReduce 进行批量遥感影像瓦片金字塔构建[J]. 武汉大学学报(信息科学版), 2013, 38(3): 278-282.]

[11] Luebke D, Martin R, Jonathan D, *et al*. Level of Detail for 3D Graphpics[M]. San Francisco, California, USA: Morgan Kaufmann Publish, 2003.

Research on Superimposition Pyramid Index Applied to Massive Points Cloud Fast Display

Zeng Xiangzhao[1,2], Li Chuanrong[1], Zhang Zheng[1], Zhou Mei[1]

(1. *Key Laboratory of Quantitative Remote Sensing Information Technology, Academy of Opto-electronics, Chinese Academy of Sciences, Beijing* 100094, *China*;
2. *University of Chinese Academy of Sciences, Beijing* 100049, *China*)

Abstract: With the development of LiDAR technology, the size of acquired data, known as point cloud has increased rapidly. However, the visualization of massive point cloud proves difficult due to the limited computer memory. To effectively visualize the massive point cloud, numerous data structures have been proposed. Among them, pyramid index structure is an effective method of the fast display for massive point cloud. In order to overcome the display delay caused by limited computer memory, this paper introduces pyramid index structure and proposes an innovative index structure that called superimposition pyramid index structure to improve the performance of the fast display for massive point cloud. First, point cloud data is divided into several cells. Within each cell, the point cloud data is then rearranged according to the structure of superimposition pyramid index proposed in this paper. Finally, each cell is organized to form an index file. The above structure has been implemented and the experiment on point cloud data from airborne LiDAR proves that the superimposition pyramid index structure effectively reduces the consumption of memory cost and accomplishes the fast display for massive point cloud.

Key words: LiDAR point cloud; Fast display; Index; Pyramid structure