文章编号:1671-8860(2005)11-0975-04　　**文献标志码**:A

机载激光雷达点云数据的实时渲染

黄先锋[1]　陶　闯[2]　江万寿[1]　龚健雅[1]

(1 武汉大学测绘遥感信息工程国家重点实验室,武汉市珞喻路 129,430079)
(2 约克大学地球与大气科学系,多伦多市 Keele 大街 4700 号,加拿大)

摘　要:提出了一种实时绘制大规模 LIDAR 点云数据的方法。该方法通过构建一棵顺序四叉树使点云均匀分布在四叉树节点上,来实现快速的数据筛选。阐述了顺序四叉树的快速建立,并通过一个试验系统验证了文中所提方法的有效性。试验表明,使用目前普通配置的计算机,通过自适应控制绘制的数据量,可以实时绘制约 1GB 的原始点云数据。

关键词:LIDAR;点云;实时可视化;顺序四叉树
中图法分类号:P237.3

机载激光雷达(airborne light detection and ranging,LIDAR)测距系统是集激光、全球定位系统(GPS)和惯性导航系统(INS)三种技术于一体的空间测量系统[1,2],能够快速、精确获取地面信息。新型的 LIDAR 系统可以接收多次反射,记录反射的强度信息,提供水平间距小于 1 m、垂直精度在 15～20 cm 左右的密集点阵数据。

由于点云数据量较大,以及计算机速度和内存限制等各方面原因,大量 LIDAR 点云数据的可视化问题目前仍没有得到很好的解决。

对大数据模型进行可视化是计算机图形学领域内一个非常复杂的问题。目前,大数据模型可视化研究的焦点大多集中在面的绘制,而不是点的绘制。LOD 模型(level of detail)[3～5]及基于点的绘制[5]等数据模型,其主要思想就是减少实际交给显卡绘制的数据量,使用少量三角形或者少量片状有向点绘制精度较低的数据,使用密集的三角形或者片状有向点绘制精度较高以及形状变化剧烈的地方。但是,它们不适合渲染点云数据,因为点云只要绘制成点,没有遮挡,所以无法使用三角形简化算法。Carsten 等人在 QSplat[6] 基础上提出了顺序点树模型,它根据绘制过程中点片误差大小,由粗到细构建了顺序树状结构,并使用紧凑的文件布置方式[7],提高了绘制的效率。然而,Carsten 等解决的仍旧是面状物体的绘制,并且,其用来控制显示精度的普通树结构并不适合对点云数据进行裁切处理。四叉树格式规整,可以快速地对树上点云数据进行裁切。

本文借鉴了 QSplat[6] 中紧凑组织数据的做法,采用了多精度排序四叉树存储模型,使得每一层上的点云分布基本均匀,同时给出了一个大数据量下快速构建均匀分布四叉树的方法,来实现实时渲染大量 LIDAR 点云数据,通过绘制时数据量的自适应控制达到实时效果。

1　基于顺序四叉树的点云数据组织

首先将点云空间按照四叉树模型进行细化。四叉树的每一个节点给一个顺序编码,依次将点云均匀地布置在每一个节点上,然后按照编码的顺序紧凑地存储这个树。绘制时,可以通过视场范围快速地进行数据裁切。由于点云数据紧凑地组织在一起,所以可以使用内存映射的方式将数据快速传给显卡的图形处理单元(GPU)进行渲染。

四叉树具有直观而且操作方便的特点,多用于 GIS 的空间索引,这里将点云均匀地布置在四叉树的每一层,不同于 GIS 的空间索引。陈俊华等在 GIS 空间数据库引擎的实现中采用了一种排序四叉树编码索引,通过编码可以实现快速的空间数据查询[9]。然而,在点云数据的可视化中,

收稿日期:2005-09-13。
项目来源:国家 973 计划资助项目(2002CB312101)。

对点云建立 GIS 概念中的四叉树空间索引并不能有效解决问题，因为四叉树空间索引的前提条件是几何对象的范围大小不同，大范围几何对象存储在树的上层，小范围几何对象放到树的下层。在点云中，所有的对象都是点，那么所有的点都将落在最后一层。所以，需要在每个节点上均匀地存储点云数据，而不能够按照 GIS 中构建空间索引的方式进行处理。

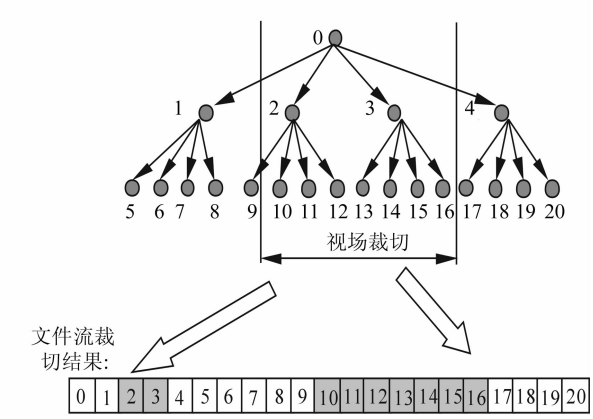


图 1 树节点的编码方法与裁切结果

Fig. 1 Coding Method of Quardtree and Culling Result

将四叉树按照如图 1 上部分所示的规则进行编码，采用从上到下的四叉树编码方式。在文件的组织上，将具有小序号编码的节点存储在文件前边，大序号编码的节点存储在文件后边。同时，使用一个按照节点编码顺序排列的表头，这个表头中记录了每个节点上点的个数。

2 绘制过程

在理想情况下，按照视角范围裁切时，可以在每一层内得到理想的具有顺序编码的数据，这些数据在文件存储结构上也是连续的。如图 1 上部分所示，落入视场范围的编码为 2～3 和 10～16 的节点，在文件中，这些节点上的数据是连续存储的，所以就可以直接把这段数据读出并给显卡绘制，不需要额外的计算。

在实际情况下，由于被裁切的点云数据在二维空间分布，所以，得到的结果没有图 1 所示那么好。例如，在图 2 上面部分，实际的裁切范围为虚线落入的深色节点，在第一层中得到 1、2 两个节点，在顺序文件结构上得到的也是 1、2 两个节点（如图 2 下面部分）；第二层中得到 6、7、9、12 四个节点，由于为了尽量保持文件中裁切结果的连续以便于快速绘制，文件中得到 6～12 共 7 个节点（如图 2 下面部分）。总共需要 6 个节点，在文件结构上得到 9 个节点，比实际上多了 3 个。但是随着裁切向下进行，可以考虑控制裁切数据的冗余度，这样会得到越来越好的结果。

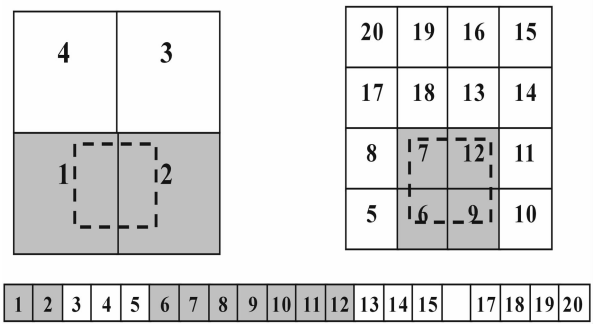


图 2 顺序四叉数裁切过程

Fig. 2 Culling Sequential QuardTree

由此可见，使用顺序四叉树可以快速简化数据。然而，如何构建这样的一棵四叉树呢？LIDAR 在获取地面数据时，一条一条地扫描地面，所以，LIDAR 获取的原始点云在存储上具有很大相关性，如果顺序地从原始点云文件中读取数据构建树，则将导致点云聚集在某个子树的某个区域内，在绘制时，按照从上到下、由粗到精的渲染方式，上层的显示结果将不能反映整个点云的分布情况。

3 均匀分布四叉树的构建

建立点云均匀分布四叉树首先要打散原始点云数据文件流的聚集性。比较常见的方法是将数据按照空间分布划分为网格，建立网格索引，然后从索引中按照一定规则读出数据。这种方法在处理小数据量时有效，在处理大数据量时效果不是很好，因为，如果要得到比较好的打散效果，需要将网格划分得非常细密，这样需要大量内存空间来存储网格信息和网格中的数据，即使使用虚拟内存的方式，仍旧会花费大量时间用于内存和硬盘的数据交换，从而导致效率的降低。本文将点云文件分为很多段，将段内的数据映射到内存，随机地从内存中读取点云。

3.1 原始点云文件的分段

直接将文件分成 100 000 段或者更多段，不建立任何索引。这个分段过程不对文件作任何操作，只是记录每一段开始的位置，下一段的开始位置就是上一段的结束位置。同时将分段中的某一部分数据映射到内存中，在构建树的时候随机读取某个分段中的数据，当所有分段内都没有数据时，则表示已经读完了点云中所有的点。分段的

数目是任意的,而且是分段越多越好,但是由于各个段内的数据需要映射一部分到内存,分段太多则需要占用大量的内存,同时分段越细,数据的读取就变得越零碎,也会降低其读取效率,所以,在分段数和内存大小之间需要进行一定的取舍。

3.2　四叉树构建

本文采用常用的从上到下构建四叉树的方法。从分段中读取点云数据,依次添加到树上,这是一个递归的过程,过程如下。

1) 从原始点云中随机读取某段中的一个点,从第一层开始计算。

2) 计算落在该层的某个节点编码,如果该节点上点云个数已经达到上限,则从下一层开始计算,继续执行 2)。

3) 根据编码存储到该层的该节点位置。

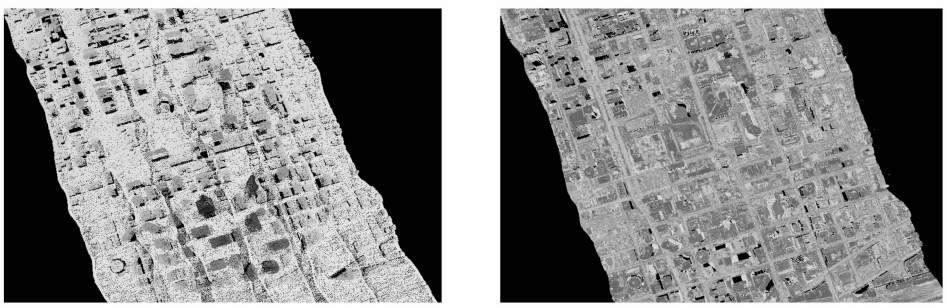
4　试　验

在上述思想的基础上,本文建立了一个试验系统,使用 OpenGL 图形渲染引擎。点云格式采用 ASPRS 的 LIDAR 数据交换标准中 Point Data Record Format 0 格式,每个节点占 20 字节。使用 11 层四叉树,每个节点最多允许存储 512 个点云,这棵树存储容量大于$(4^{11}+4^{10})\times512\times20$ GB,约 53GB。为了防止原始点云在一块区域聚集而导致小区域内超过 11 层,在树的最下层附加一张$2^{11}\times2^{11}$的网格,所有超过 11 层的点都落入到这个网格中,这个网格也采用与四叉树一致的编码方式。

构建四叉树首先需要计算节点的编码,节点编码是一个递归过程,由于编码计算过程中全部是整型数据的计算,在 10 层四叉树中为每个节点计算编码需要 7.4 s。为了避免运行时计算节点的编码,要将节点的编码预先计算好放在内存中,在运行时直接读取,避免运行时不断切换内存状态,因为运行时需要占用大量的内存和虚拟内存,频繁地切换内存状态会导致效率的降低。另外,由于层数越多,数据的表头就会越大,并且越是下层,没有数据的节点就会越多,被使用的频率就越少,所以,在 11 层以后可以采用映射的存储模式来存储表头。为了减少递归的次数,记录下有数据的最大层号,添加数据时直接试探着添加到最大层号减 1 层,如果该层已满,则继续试探着添加到下一层;如果该层未满,则看看上层该位置是否已满。如果上层节点已满,则添加到该层;如果上层未满,则试探着添加到上一层。笔者试验了一个有 8 154 752 个点的点云,函数总共调用 15 892 241 次,而没有优化时调用了 92 548 926 次。在构建过程也需要建立主存与辅存之间的交换机制,因为要避免构建四叉树受到内存大小和数据量的限制。

由于 LIDAR 点云具有多次反射,因而在文件中为每次反射建立一个四叉树,将多次反射的四叉树表头信息放在一条,点云数据集中放在文件后边,用变量记录每次反射的四叉树布置信息。这样可以将可见的那次反射数据快速传给显卡图形处理单元(GPU)绘制,而不需要判断每个点属于哪次反射、该反射数据是否可见。

在绘制时,按照视场范围裁切,并按照树的编码从小到大绘制点云,绘制过程中不断检测是否需要重新绘制,并计算每次绘制的时间。当设定绘制速度为 100 帧/s 时,每当一次绘制的时间大于 0.01 s,则立刻停止绘制,CPU 空闲时则不断地绘制细化数据。由于点云均匀分布在四叉树上,所以无论以多少帧率绘制,结果基本上能够表现出数据的全貌。图 3(a)是对 1 GB 原始数据共约20 000 000个点,以 100 帧/s 速度绘制的结果,图 3(b)是以 20 帧/s 对显示强度着色效果。



(a) 100帧 / s　　(b) 20帧 / s, 显示强度

图 3　原始点云 1 GB 数据
Fig. 3　Result of 1 GB Raw Point Clouds

5 结 语

本文通过将点云均匀布置在顺序编码四叉树上,绘制中实时对节点进行裁切,并通过自适应控制绘制的数据量,解决了实时渲染大量 LIDAR 点云数据的问题,为将来的海量 LIDAR 点云渲染奠定了一定的基础。进一步的工作将集中于以下方面。

1) 树的编码计算是递归计算方式,需要寻找一种更加快捷的计算方式。

2) 原始文件进行分段的方式并不是构建均匀分布四叉树的最好方式,如何让四叉树在从上到下的显示过程中绘制的结果均匀分布仍旧是需要继续研究的问题。

3) 如果要得到更好的光照显示效果,点云中每个点都需要计算法向量。计算法向量时如果只考虑一层内的数据,则可能会导致法向量失真,如果考虑所有的层,则对大量数据计算法向量将是一个需要解决的问题。

参 考 文 献

1 李清泉,李必军,陈 静. 激光雷达测量技术及其应用研究. 武汉测绘科技大学学报,2000,25(5):387～392

2 李英成,文沃根,王 伟. 快速获取地面三维数据的 LIDAR 技术系统. 测绘科学,2002,27(4):35～39

3 杨崇源,张继贤,林宗坚. 虚拟地形场景绘制中的实时 LOD 算法. 测绘学报,2001,30(2):133～139

4 Hu Y. Automated Extraction of Digital Terrain Models, Roads and Buildings Using Airborne Lidar Data: [Ph. D Dissertation]. Calgary: Univ. of Calgary, 2003

5 Fabio R. From Point Cloud to Surface: the Modeling and Visualization Problem. ISPRS Tarasp-Vulpera, Switzerland, 2003

6 Rusinkiewicz S, Marc L. QSplat: A MultiResolution Point Rendering System for Large Meshes. Procession of SIGGRAPH, New Orleans, 2000

7 Dachsbacher C, Vogelgsang C, Stamminger M. Squential Point Trees. Procession of SIGGRAPH, San Diego, 2003

8 陈俊华,宋关福,李绍俊. 基于 RDBMS 的空间数据库的设计与实现,中国 GIS 年会,成都,2001

9 ASPRS. LIDAR Data Exchange Format Standard Version 1.0. http://www.lasformat.org, 2003

第一作者简介:黄先锋,博士生,主要从事 LIDAR 数据分析处理、模式识别和 GIS 研究。

E-mail:hxf_whu@163.com

Real-time Render Large Amount of LIDAR Point Clouds Data

HUANG Xianfeng[1] *C. Vincent Tao*[2] *JIANG Wanshou*[1] *GONG Jianya*[1]

(1 State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, 129 Luoyu Road, Wuhan 430079, China)

(2 Department of Earth and Atmospheric Science, York University, 4700 Keele Street, Toronto, Canada)

Abstract: The authors propose a method to real-time render large amount of LIDAR points clouds. In order to fast culling invisible data and make result of progressive rendering reasonable, a sequential coded Quardtree is built with points clouds evenly distributed to nodes of every layer from top to down. The authors discussed the procedure of building a sequential coded QuardTree, explain how to add points clouds to nodes of tree. The authors also implement a system to prove the method proposed in this paper, which can real-time render 1GB raw point clouds(about 20 million points) data with common commodity computer.

Key words: LIDAR; points clouds; real-time visualization; sequential coded Quardtree.

About the first author: HUANG Xianfeng, Ph.D cadidate. He is engaged in procession of LIDAR data, pattern recognition and GIS.

E-mail: hxf_whu@163.com