虚拟旅游中海量3D点云数据的细节层次索引技术研究

赵尔平 党红恩 刘 炜
(西藏民族大学信息工程学院 咸阳 712082)

摘 要 虚拟旅游中的3D点云数据特别庞大,批量索引成为了当今的研究热点。许多索引树存在兄弟结点空间区域重叠、不能实现细节层次索引、索引效率低等问题。为此,将点数据反射强度和细节层次技术引入R树,在改进R树的基础上提出LODR树。建树前,将点云数据进行排序、分组、去除空间重叠等预处理。树的每层设有不同反射强度阈值,把叶结点中满足阈值条件的索引记录沿父-祖父-曾祖父的家谱关系上移,并插入对应的非叶结点,利用该方法创建细节层次索引树。利用反射强度控制数据冗余,棱锥裁剪技术实现查询优化。实验结果表明,LODR树在细节层次索引、查询效率等方面具有明显优势。

关键词 虚拟旅游,3D点云数据,细节层次,索引结构
中图法分类号 TP392 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.10.032

Research on Detail Level Index Technology of Massive 3D Point Cloud Data in Virtual Tourism

ZHAO Er-ping DANG Hong-en LIU Wei
(School of Information Engineering, Xizang Minzu University, Xianyang 712082, China)

Abstract 3D point cloud data in virtual tourism are particularly huge and the batch index has become a research hotspot. There are some problems in many index trees, such as spatial overlap of sibling node, not achieving level of detail index and low indexing efficiency. Therefore, point data reflection intensity and level of detail technology were introduced into R-tree, and LODR-tree was presented based on improved R-tree. Before establishing this tree, point cloud data needs to be pre-processed, such as sorting, grouping, removing spatial overlap and so on. The index records which meet the thre-shold conditions in the leaf nodes are inserted into the homologous non-leaf nodes along the parent-grandfather-great grandfather family relationship, and LOD index tree is created by this method. Data redundancy is controlled by reflected intensity, and query optimization is achieved by pyramid cutting technology. Finally, experiments show that LODR-tree has obvious advantages in LOD index and query efficiency.

Keywords Virtual tourism, 3D point cloud data, Level of detail, Index structure

1 引言

“互联网+”旅游是旅游产业线上线下融合的创新模式。它既是真实旅游的前期体验,也是旅游资源对外宣传的有效途径,如今在国内外逐渐兴起。目前,主要借助三维激光扫描仪来采集真实景区数据,通过相应软件把采集的3D点云数据重建为虚拟旅游模型。一个旅游景区的点云模型的顶点数一般可达万亿数量级,正如文献[1]所述,一次大范围扫描运动产生了数十亿点样,因数量太多而不能在大多数计算机中存储管理。因此虚拟旅游3D点云数据存储管理和批量索引一直是数据库领域亟待解决的问题。

基于关键字查询的传统数据库不具备空间索引功能,空间索引性能的提高不仅需要寻求高效索引算法,而且需要选择合适的空间索引结构。k-d(k-dimensional)树、bsp(binary space partitioning)树、octree树、R树、R^+树、R^*树都是常用的空间索引结构。k-d树在最近邻域查找上比较有优势,但随着数据规模和维度的增加,搜索时间成为其主要问题[2]。BSP树的删除、更新操作复杂,不具备大规模数据组织管理和空间索引能力。数据量大、分布不规则时,octree树的搜索性能退化、变差[3],但它具有独特的空间剖分优势。R树族成为最受欢迎的空间索引结构是因为其简单且高效[4],但是R^+树和R^*树的插入与删除操作相当繁琐、低效,因此R树便成为了多维空间数据比较理想的索引结构,它用最小边界矩形MBR(Minimal Bounding Rectangle)来表示索引对象,MBR可以扩展到多维空间。R树中兄弟结点对应空间区域重叠造成多路查询是其索引性能降低的主要原因。如果构造R树前能对点数据进行预处理,那么多路查询低效问题就能得到有效解决,正如文献[5]谈到R^+树、R^*树虽然改进了R树性能,但如果对数据预进行处理,R^+树、R^*树的查询时间则无竞争力。而且R树的动态更新比R^+树和R^*树灵活、简单,

到稿日期:2016-12-29 返修日期:2017-04-13 本文受国家自然科学基金(41361044),西藏自治区自然科学基金(12KJZRYMY07)资助。
赵尔平(1976-),男,硕士,副教授,主要研究方向为数据库技术,E-mail:xdzep@163.com;党红恩(1978-),男,硕士,讲师,主要研究方向为分布式数据库;刘 炜(1976-),男,博士,副教授,主要研究方向为移动数据库。

因此很多学者基于 R 树空间索引做了大量研究工作。

2 研究现状

为了解决 R 树中兄弟结点空间区域的重叠问题，CSR-tree[5]将点云排序并通过按距离相关性将其聚类为 k 个群来构造 R 树。近似棱镜方法[6]采用 R 树管理 3D 模型二维映像凸包构造的若干近似棱镜来降低空间区域重叠。R-Forest[7]把互不相交的 R 树构造成森林来管理空间区域互不重叠数据集，但是将这种复杂索引结构用于海量 3D 点云数据得不偿失。

VoR-Tree[8]索引结构中 R 树的叶结点存储空间数据的维诺图(Voronoi diagrams)及其最近邻居指针；Dinko Osmankovic 等[9]给三维坐标分别增加正负 Δ 值来影响 NNS 准确度。文献[8-9]将近邻域搜索算法引入 R 树以提高索引性能。Zhang 等[10]提出了二棵索引树结构，第一棵 ER-tree 树索引不同空间区域，第二棵 R* 树聚类了空间关系特征向量；PHR-Tree[11](Perfect Hash Base R-tree)将两级哈希表集成到 R 树，并为结点内部点建立哈希索引。文献[10-11]中的索引结构复杂，时间和空间复杂度都不适合庞大的点云数据。

细节层次 LOD(Level of Detail)索引结构中 R 树的非叶结点参与目标数据管理，叶结点管理全部数据，代表最精细层，越往树的上层，精度越低。其中 3DRTree[12]和 V-Reactive[13]最有代表性，它们将视点距离与 R 树不同层相对应，从叶结点选取最具对象特征的目标数据，并将其交给 R 树上层管理，实现不同视点距离细节层次索引。由于缺少不同细节层次约束机制，因此造成了大量冗余数据。SDMR 树[14]把分辨率维引入 R 树，允许非叶结点管理较低分辨率的数据，该方法遵循的是传统静态 LOD 思想，不同细节层次数据存在冗余。文献[15]在 Hadoop 平台上利用 R 树并行加载与索引，文献[16]在 GPU 上利用 R 树实现并行窗口查询处理与有效显示，它们都是利用 R 树实现海量点云数据并行索引，是空间索引技术的有益尝试。

尽管 R 树空间索引存在一些不足，但由于 R 树灵活与简单的特点，其在实际应用中依然被青睐。正如文献[17]所述，在不利于设计的数据集和查询中，R 树的性能与顺序扫描方式的性能一样低，然而在真实数据集上 R 树的效率较高，它被广泛应用于商业系统。线上旅游是典型上、下、左、右移动的空间索引，将 R 树作为索引结构是最佳选择。基于以上理由并结合激光数据的固有特性提出细节层次索引树 LODR-Tree (Level of Detail R-Tree)，把点数据自带的反射强度信息作为第四维索引引入 R 树，在树的不同层预设不同反射强度阈值，允许非叶结点参与目标数据管理并直接获取不同反射强度的点数据；为了避免兄弟结点空间区域重叠，建树前对数据进行预处理。

3 LODR 树的索引结构

基于细节层次技术的 LODR-Tree 将树的各层对应不同细节层次，越往树的上层，反射强度阈值越大，管理的数据越少，表示对象较粗细节层；越往树的下层，反射强度阈值越小，叶结点管理全部数据，代表最详细节层。

3.1 LODR 树的引出与定义

3D 点云模型中反射强度值大的点构成物体凸面，相反则构成物体凹面，凸面反射光线最容易且最快进入人的视线，凸面点集最能代表物体特征，利用树的不同层管理反射强度值不同的点数据的方法实现多细节层次索引。传统 R 树的叶结点管理全部数据，不具备细节层次索引，因此对 R 树做以下 3 点改进：1)将反射强度作为第四维引入 R 树，用四维数据实现空间索引；2)允许非叶结点管理数据，即把 R 树的叶结点管理的空间区域互不重叠索引记录上移至非叶结点中进行管理；3)为了适应动态漫游，允许同层访问。改进后的 R 树称作 LODR-Tree，反射强度作为细节层次索引约束条件，每层预设不同反射强度阈值，最粗细节层不一定对应树的根结点。下面是 LODR 树的定义，设 fan 为结点中索引记录的最大数目，即树的扇出，$m(m=fan/2)$ 为非根结点索引记录数下限，用于确保磁盘空间的高效利用。

定义 1 1)除根结点外，每个结点索引记录数都介于 m 和 fan 之间；2)根结点至少有两个孩子，除非它是叶结点；3)每个结点对应一个矩形，非叶结点矩形为所有子结点矩形的外包矩形(可扩展多维空间)；4)叶结点对应的矩形为其所有互不相交对象的外包矩形，且位于同一层；5)允许叶结点索引记录沿父-祖父-曾祖父家谱上移至非叶结点，允许同层访问；6)反射强度作为细节层次索引约束条件，不允许索引小于某层反射强度阈值的点数据。

由定义 1 可知 LODR 树与 R 树的结构有三点不同：1)LODR 树允许非叶结点直接访问数据和同层访问；2)索引项增加了反射强度维，叶结点的索引记录为(Index，Obj_ID，Intensity)，非叶结点的索引记录为(Index，Child_Pointer，Intensity)，Index 是外接立方体 MBR，Obj_ID 是对象编号，Intensity 是反射强度，每层 Intensity 有固定阈值，且只能索引大于它的数据；3)把叶结点中满足阈值条件的索引记录沿父-祖父-曾祖父的家谱关系上移，并插入对应的非叶结点。LODR 树弥补了传统 R 树不能实现细节层次索引以及因兄弟结点空间区域重叠而造成多路查询的不足，解决了文献[12-14]中细节层次索引带来的数据冗余问题。

3.2 点云数据预处理

定理 1 $v_1,v_2,v_3,\cdots,v_i$ 为立方体，$I_1,I_2,I_3,\cdots,I_j$ 为立方体所包围点数据的最小反射强度($1<i,j\leqslant fan$)，任意两个 v_i，v_j 中点的反射强度的最小值分别为 $I_i,I_j(I_i>I_j)$，假设 $v_i\cap v_j\neq\varnothing$，则 $v_j\cup v_i=v_j'$，v_j'中点的反射强度的最小值 $Intensity=I_j$。

推论 1 假设 $v_i\cap v_{i+1}\neq\varnothing,v_{i+1}\cap v_{i+2}\neq\varnothing,\cdots$，其中 $I_i>I_{i+1}>I_{i+2}>\cdots(1\leqslant i<fan)$，则 $v_{i+1}\cup v_{i+2}\cup v_i\cup\cdots=V$，$V$ 立方体反射强度阈值取 v_{i+2} 的值，即 $Intensity=I_{i+2}$。

创建 LODR 树前需做三方面的预处理工作：1)为了避免兄弟结点空间区域重叠，确保位置相邻点聚集在相同父结点，利用八叉树的空间剖分优势和快速收敛能力对点云模型进行剖分，迭代剖分约束条件为八叉树的叶结点数据个数不大于 LODR 树中叶结点管理的点数据个数。分割阈值计算公式为：

$$t=\frac{size_{disk}}{byte_{point}}\times num_{page} \tag{1}$$

其中，$size_{disk}$ 代表一个磁盘页面的大小，num_{page} 代表磁盘页面个数，$byte_{point}$ 代表一个点云数据的大小。2)反射强度预设阈值由索引树的层数、细节层次数和每个细节层次索引数据比例 3 个因素共同决定。如图 1 所示，LODR 树的高度为 4，细节层次分为 4 级，相邻细节层索引数据量大约相差 1/4，例如

低细节层(low)索引大约 1/4 的点云数据，中等细节层(middle)索引 2/4 的点云数据，高细节层(high)索引 3/4 的点云数据，最高细节层(superhigh)索引全部点云数据。四级反射强度阈值分别用 I_l，I_m，I_h，I_s 表示，且 $I_l>I_m>I_h>I_s$，该 LODR 树为四级细节层次索引树。3)计算 LODR 树的扇出，由于每个结点仅占一个磁盘页面，扇出 fan 是每个结点管理索引记录数的上界，其计算公式为：

$$fan=\frac{size_{disk}}{byte_{record}} \tag{2}$$

其中，$byte_{record}$ 表示一条索引记录的字节数。激光扫描的 3D 点云数据具有大规模、无拓扑结构、无组织、无定向、离散和密度不均匀的特性[22]。利用该特性仅取八叉树叶结点的点云数据来构造 LODR 树，以保证数据量不因预处理而增加。

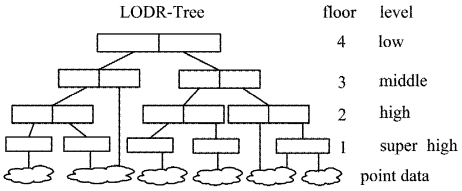

图 1　LODR 树结构

3.3　创建 LODR 树

逐个取八叉树叶结点数据，并按 Intensity 关键字排序，同时将它们分成若干组，分组约束条件为分组数不大于扇出 fan。计算每组数据立方体的范围 MBR_1，MBR_2，…，MBR_i，MBR 用立方体左下角($X_{\min}Y_{\min}Z_{\min}$)和右上角($X_{\max}Y_{\max}Z_{\max}$)坐标表示，每组点云数据的最小反射强度用 I_1，I_2，…，I_l，…，I_s 表示，显然有 $I_1>I_2>I_l>\cdots>I_s$ 成立。利用定理 1 消除它们的空间重叠，由推论 1 可知最坏情况下 LODR 树的一个叶结点只有一条索引记录。当立方体合并后分组数小于 $fan/2$ 时，这些分组合并到下一个叶结点重新处理，以保证每个叶结点充分利用磁盘空间。然后将消除空间重叠的分组作为 LODR 树叶结点外接立方体，每组索引记录($Index$，Obj_ID，$Intensity$)的反射强度取该组最小值，若分组的最小 $Intensity\geqslant I_l$，则索引记录 $Intensity=I_l$；若分组的最小 $Intensity\geqslant I_m$，则索引记录 $Intensity=I_m$，以此类推。为了构建具有细节层次的索引树，需把叶结点中 $Intensity\geqslant I_h$ 的索引记录上移并与叶结点索引记录一起批量插入父结点。此时，该父结点包含的索引记录条数满足$[1,\frac{2}{4}fan+1]$，继续执行下一个叶结点的创建和索引记录的批量插入，插入前要检查父结点剩余空间是否满足批量插入条件，若不满足则该父结点先分裂，然后插入到新分裂的父结点中。按此继续执行，若父结点创建完成，则把父结点索引记录($Intensity=I_m$)和它包围的所有叶结点中$Intensity\geqslant I_m$的索引记录批量插入祖父结点，此时该祖父结点包含的索引记录条数满足$[1,\frac{2}{4}fan+1]$，同理，插入前需检查祖父结点剩余空间是否满足批量插入条件，若不满足则先分裂再批量插入。同理，继续创建叶结点、父结点和祖父结点，把新生成的祖父结点索引记录(将 $Intensity$ 值设置为 I_l)和它包围的所有父结点中所有叶结点的 $Intensity\geqslant I_l$ 的索引记录批量插入曾祖父结点。按此继续创建整个 LODR 树。

4　LODR 树的细节层次索引机制

4.1　LODR 树索引方法

R 树索引机制是从根结点开始到叶结点逐层检查与查询窗口交叉的所有结点，获取与叶结点外接立方体重合区域的全部数据，此结构是导致 R 树不能实现多细节层次索引的根本原因。LODR 树允许非叶结点参与管理一定比例的目标数据，管理具有层次性、分散性，以图 1 给出的 LODR 树为例，叶结点符合条件索引记录，仅沿父-祖父-曾祖父家谱关系上移至树的非叶结点中，使得曾祖父层可以管理 1/4 的点云数据，祖父层管理 2/4 的点云数据，父层管理 3/4 的点云数据，叶结点层管理全部点云数据，从而实现 LODR 树多细节层次索引机制。为了适应漫游过程中视点位置不断变化的需求，允许同层结点访问。细节层次索引原理：首先，查询窗口在曾祖父层搜索与其区域重合区域，直接查询该区域包含的所有索引记录管理的数据($Intensity\geqslant I_l$ 的 1/4 点数据)，也可以继续访问该层其他结点，实现最粗细节层次索引；其次，若查询窗口进入祖父层，则直接查询与该区域重合的所有索引记录管理的 2/4 的点数据，也可以继续访问该层其他结点，实现中等细节层次索引；然后，父结点层实现较高细节层次索引，查询重合区域 3/4 的点数据；最后，在叶结点层实现最详细节层次索引，查询重合区域全部点数据。LODR 树细节层次索引机制能够实现 3D 点云模型按客户端精度请求来查询不同分辨率的数据。同层访问机制符合虚拟漫游时查询窗口不停地上、下、左、右移动的特点，当查询范围变化后不必重新返回 R 树的根结点，而是直接访问同层结点，由于 LODR 树中一个结点独占一个磁盘页面，同层访问比从根结点开始访问减少了 I/O 次数，正如文献[19]所述查询算法性能主要取决于 I/O 代价。由于 LODR 树下一层结点包含了上一层对应结点管理的目标数据，因此同层结点直接访问时不会丢失上一细节层次数据。故 LODR 树细节层次索引是动态 LOD 而非传统静态 LOD，且无数据冗余，有效缓解了庞大点云数据漫游过程中的网络带宽瓶颈问题。

4.2　数据冗余控制

为了在同层访问时不丢失上层结点管理的数据，上层结点中包含的叶结点索引记录又被全部插入其对应的下层结点中。索引记录重复造成进入下一层查询时有可能产生冗余数据，假如视点从 LODR 树上层以垂直方式进入下一层区域，则会造成最大数据冗余，因此必须进行有效控制。用符号 R_s，R_h，R_m，R_l 分别表示叶结点、父结点、祖父结点、曾祖父结点包含的索引记录对应的数据集，则存在关系 $R_s\supset R_h\supset R_m\supset R_l$，很显然它们存在数据冗余。利用空间区域包含关系和树的各层反射强度阈值的不同来控制因细节层次索引机制造成的数据冗余。假设 W_i 代表当前视点在树的上层空间区域，I_i 为该层反射强度；W_j 代表下一层空间区域，I_j 为该层反射强度；W_i'代表 W_i 在下一层的投影，若 $W_i'\cap W_j=w$，则 w 代表从上层进入下层时的数据冗余区域，在 w 区域内，利用 $I_i>Intensity>I_j$ 作为查询过滤条件进行数据冗余控制。最坏情况是 $W_i'=W_j$，最好情况是 $W_i'\cap W_j=\varnothing$，则不存在数据冗余。不同细节层重复插入叶结点索引记录虽然在一定程度上增加了树的高度，并且在查询时还要进行数据冗余处理，

但是这种重复并不多余，它能使 LODR 树既能按深度索引也能按广度索引，符合虚拟漫游查询请求实时上、下、左、右变化的需求，避免了空间位置改变时 R 树必须从根结点开始搜索的弊端。

4.3 棱锥裁剪查询优化

虚拟旅游海量数据实时漫游时系统的开销非常大，一般用户难以承受，为此人们想尽一切办法进行改进。通常做法是裁剪、丢弃视景体外的数据，仅仅渲染某时刻相机视景体内的局部数据。但这种做法仅减少了渲染阶段的系统开销，未能解决网络资源瓶颈问题，也未能从源头减少有效数据传输量。采用锥体裁剪技术进行查询优化可以从源头解决数据冗余问题，减少资源开销。远大近小的规律被大家熟知，漫游过程中远视点可以看到整个场景，近视点只能看到场景的一部分区域。LODR 树不同层不仅代表空间对象的不同细节层，还可以对应到视点远近距离，最上层对应最远视点，可以查询整个对象最粗细节层；叶子层对应最近视点，可以查询对象局部区域的最详细数据。引入锥体裁剪查询技术，把棱锥不同层和 LODR 树层次一一对应，视点由远到近移动时可视区域范围不断缩小，虽然不能达到连续缩小可视区域、实时减少数据量的效果，但是当视点从棱锥当前层进入下一层时，对应查询就从 LODR 树的当前层进入下一层，通过裁剪技术缩小了下一层的查询区域。图 2 给出了图 1 中的索引树对应的裁剪棱锥。以 xoy 面垂直投影为例来计算裁剪区域（其他面投影同理），已知视点最远视距 H、最近视据 h、可视棱锥两层之间的距离 d 以及当前层立方体 MBR 左下角（$X_{\min}Y_{\min}Z_{\min}$）和右上角（$X_{\max}Y_{\max}Z_{\max}$）坐标。因为投影后的立方体大小在原来立方体基础上对称均匀缩小，x，y 坐标缩小绝对值用 m 表示，计算公式为：

$$m=(1-\frac{hd}{H})\times\frac{\sqrt{(X_{\max}-X_{\min})^2+(Y_{\max}-Y_{\min})^2}}{2}\times\frac{\sqrt{2}}{2} \tag{3}$$

z 坐标缩小绝对值用 n 表示，计算公式为：

$$n=\frac{(1-\frac{hd}{H})\times(Z_{\max}-Z_{\min})}{2} \tag{4}$$

所以可得 xoy 面垂直投影后的立方体 MBR_s 左下角坐标（$X_{\min}+mY_{\min}+mZ_{\min}+n$）和右上角坐标（$X_{\max}-mY_{\max}-mZ_{\max}-n$）。对于垂直方向投影，先计算垂直方向投影再进行平移。视点由当前层进入下一层，棱锥裁剪技术能裁剪索引区域使其按视点距离缩小查询范围，很大程度地减少了查询、传输、渲染阶段的有效数据量，加快了传输和渲染速度。锥体裁剪技术最适合较低层查询，因为树的较底层代表较近视点，此时可视范围缩小的幅度更大，裁剪更有效益，系统开销减少得更多，查询效率提高程度最大。

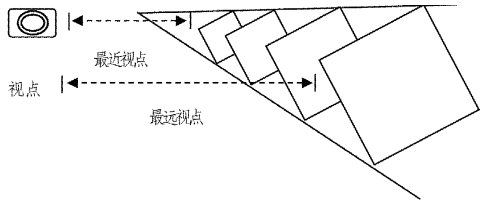

图 2 裁剪棱锥二维图

5 实验结果与结论

实验数据集是 TrimbleVX 激光扫描仪采集的 3D 点云数据，通过 Trimble RealWorks 三维软件处理得到 3D 点云模型，测试过程中 LODR 树所有结点独占一个存盘页面，大小为 4kB，机器内存为 4GB。将 LODR 树与 R 树、3DRTree 树和 V-Reactive 树的性能进行对比分析，因为 3DRTree 树、V-Reactive树和 LODR 树都是基于 R 树改进的细节层次索引结构，具有可比性，3 种树的细节层次都设置为 4 级。

5.1 创建索引性能比较

创建索引是一项非常重要的性能指标，LODR 树是为具体 3D 虚拟旅游模型创建的索引结构，创建后在系统运行期间不存在插入、删除、分裂结点等操作，即结构不发生改变，它是一种静态索引树，除非点云模型改变。表 1 列出了创建 4 种索引树的时间开销，单位为分钟。由表 1 结果可知构造 R 树索引结构的时间最少，构造多细节层次 V-Reactive 树的时间最多，LODR 索引结构的时间花费第三。R 树比其他 3 种树用时少的原因在于创建 R 树时不需要对 3D 点云数据预处理（排序、划分、位置相邻结点聚集），直接按照输入数据的顺序为其创建索引结构。而 LODR-Tree 和 3DRTree 为了避免兄弟结点区域重叠，其位置相邻结点聚集于同父结点，创建索引树前用八叉树对模型空间进行剖分；其次，LODR-Tree 按反射强度维进行排序、分组、去重等预处理，同时细节层次处理耗时，因此时间花费较大。3DRTree 也存在细节层次处理耗时的问题，但仅是某些叶结点整体上移，不及 LODR 细节层次处理细腻，因此时间开销略小。V-Reactive 虽然没有数据预处理开销，但按权值进行细节层次处理时要不断地进行结点分裂、调整树的平衡，因此建树时间花费最大。表 1 反映了创建 4 种索引结构的时间开销的总体趋势会随数据量减少而减少，但这种变化是非线性的，例如第二组数据集的大小几乎是第一组的一半，但 4 种索引树创建时间都未减少一半；第三组数据证明创建索引结构的时间开销也不会随数据量的增加而跳跃式增长，实验证明 4 种索引结构在实际应用中都是可行的。

表 1 创建 4 种索引结构的时间

数据/GB	LODR-Tree/min	R-Tree/min	3DR-Tree/min	V-Reactive/min
37.47	29.25	13.70	26.40	33.90
18.60	21.30	8.64	20.95	24.70
51.29	38.70	17.05	33.65	43.45

5.2 查询效率测试

3 种树的细节层次虽然都是 4 级，但是每棵树为实现细节层次所选参数不同，LODR-Tree 采用反射强度取值范围，3DRTree 采用离重心远近距离，V-Reactive 采用比例尺权值，选取标准不统一，因此查询性能不适合在漫游时进行动态计算，通过查询不同位置、数据量不等的多个物体进行等价测试，比较计算平均查询性能，平均查询效率的测试结果如图 3 所示。LODR 树明显优于其他两种索引结构，因为 LODR 树的细节层次划分比较均匀（每提高一个细节层次，大约增加 1/4 的数据量），它能直接从非叶结点查询数据，而且这些数

据属于同一家谱，减少了查询时的 I/O 开销，且数据冗余得到了有效控制；再者，LODR 利用棱锥裁剪技术进行查询优化处理，查询优化能使查询系统快速响应用户提交的查询请求[20]，所以查询效率高。3DRTree 把符合条件的叶结点上移并插入非叶结点，使得本来聚集且相邻的结点上移后分布在不同区域，造成索引范围变大，数据冗余增多，从而导致查询时间变长。V-Reactive 树管理物体不同细节层，属于传统 LOD，其存在大量数据冗余。R 树索引是从树根结点开始进行搜索和区域匹配，直到在叶结点集中获取数据，该过程需要多次访问磁盘页面，因此 R 树查询效率最差。动态漫游需经常上、下、左、右移动窗口，LODR 树允许同层访问，当查询范围发生变化时不必重新从根结点开始，只需直接在当前层搜索，由 3.2 节论述可知，这样不会造成上层数据丢失，但是其他 3 种树必须从根结点开始搜索，因此在动态漫游环境中 LODR 树的平均查询性能更优。

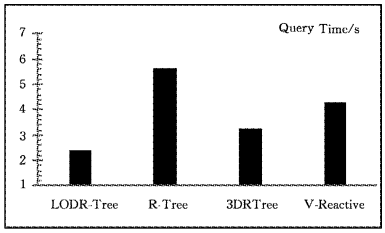

图 3　平均查询效率

5.3　数据冗余比较

细节层次索引的目的是最大程度地减少索引多余数据，真正减轻网络和客户端的资源负载。目前许多细节层次索引结构由不同分辨率模型实现，虽然实现了细节层次功能，却带来了大量数据冗余，使得海量的点云数据雪上加霜。数据冗余是指查询时数据库服务器输出无效数据量。在 4.2 节查询性能测试的基础上通过统计服务器输出数据量的方法来测试冗余性能。由于 R 树不存在细节层次功能，查询中不会产生数据冗余，实验以 R 树查询数据量作为标准，用其他索引结构查询输出的数据量除以标准数据量的商来判断哪种索引树的查询冗余较大，多物体平均查询冗余性能测试结果如图 4 所示。由 3.2 节可知 LODR 树的细节层次索引机制有效控制了查询时产生的数据冗余，因此它的数据访问量几乎与 R 树相同，在图 4 中其非常接近于 1 倍线位置，而 3DRTree 树的多细节层次由于没有控制重复数据和增大查询区域导致数据冗余较大，V-Reactive 树的每个细节层次是不同分辨率的独立数据，因此它查询时的重复数据最多，查询冗余最大。

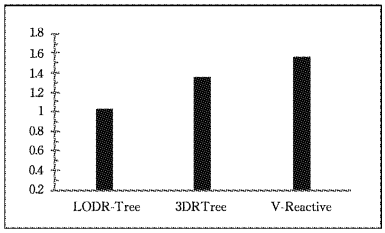

图 4　数据冗余性能

结束语　线上虚拟旅游逐渐兴起，虚拟旅游模型中海量 3D 点云数据的高效索引是当前亟待解决的问题。文中展望了空间索引技术的发展，对现有索引树的性能进行了分析和比较，R 树比较适合作为 3D 点云数据索引结构，结合点云数据自带的反射轻度信息的特点提出细节层次 LODR 树索引结构。首先介绍 LODR 树的概念和规范定义，阐述建树前的数据预处理，即排序、分组、消除空间重叠等；其次详细阐述创建细节层次索引结构的原理；然后论述细节层次索引、数据冗余控制机制以及棱锥裁剪查询优化技术；最后设计 3 类实验，从创建索引时间、平均查询效率、查询时的平均数据冗余量 3 个方面进行比较，证明了 LODR 树查询效率高和数据冗余量少的优点，它非常适合作为海量 3D 点云数据空间的索引结构。

旅游景点地域广阔，构建一个 3D 模型不现实，必须构建多个 3D 场景模型。传统集中式索引结构已不能胜任多场景点云模型，因此下一步研究工作是利用大数据和分布式存储系统把多个点云模型分别部署在分布存储系统的不同设备结点上，这既能解决大规模数据的存储问题，又能保证系统具有较高吞吐量，进而可有效解决传统系统资源不足的问题，提高 3D 点云数据索引性能。将 LODR 索引结构与 Hadoop 或 Spark 平台相结合，利用 HDFS 存储 3D 点数据及索引，Spark 在内存中进行数据计算，实现并行加载和细节层次分布式索引，是今后的研究重点。

参 考 文 献

[1] MERRY B, GAIN, MARAIS P. Fast in-place binning of laser range-scanned point sets[J]. ACM Journal on Computing and Cultural Heritage, 2013, 6(3): 1-19.

[2] LIN J H, NOOSHABADI S, AHMADI M. Parallel Randomized KD-tree Forest on GPU Cluster for Image Descriptor Matching [C]// Proceedings of 2016 IEEE International Symposium on Circuits and Systems. New York: IEEE Press, 2016: 582-585.

[3] RIZKI P N M, PARK J, OH S, et al. STR-Octree Indexing Method for Processing LiDAR Data[C]// Proceedings of IEEE SENSORS 2015. New York: IEEE Press, 2015: 1-4.

[4] YANG H, QI W T, GAO X F. Efficient R-Tree Based Indexing Scheme for Server-Centric Cloud Storage System [J]. IEEE Transactions on Knowledge and Data Engineering, 2016, 28(6): 1503-1571.

[5] HE Z W, WU C L, WANG C. Clustered Sorting R-tree: An Index for Multi-Dimensional Spatial Objects[C]// Proceedings of 2008 Fourth International Conference on Natural Computation. New York: IEEE Press, 2008: 348-351.

[6] SUN W B, WANG Z P, LIU X L, et al. The Method for Indexing Global-scale 3D Vector Data with R-tree Based on Approximate Prism[C]// Proceedings of The 18th International Conference on Geoinformatics: GI-Science in Change. New York: IEEE Press, 2010: 1-4.

[7] SHARIFZADEH M, SHAHABI C. VoR-Tree: Rtrees with Voronoi Diagrams for Efficient Processing of Spatial Nearest Neighbor Queries[J]. VLDB Journal, 2010, 3(1): 1231-1242.

[8] OSMANKOVIC D, SUPIC H, VELAGIC J. Performance and quality assessment of R-tree based nearest neighbour search in

the scalar field mapping technique[C]// Proceedings of the 2013 39th Annual Conference of the IEEE Industrial Electronics Society. New York: IEEE Press, 2013: 2455-2459.

[9] ACHAKEEV D, SEEGER B, WIDMAYER P. Sortbased Query-adaptive Loading of R-trees[C]// Proceedings of 2012 the 21st ACM International Conference on Information and Knowledge Management. New York: ACM, 2012: 2080-2084.

[10] ZHANG J, PAN H, YUAN Z M. A Novel Spatial Index for Case based Geographic Retrieval[C]// Proceedings of 2009 International Conference on Interaction Sciences. New York: ACM, 2009: 342-347.

[11] PATEL P, GARG D. Perfect Hashing Base R-tree for Multiple Queries[C]// Proceedings of 2014 IEEE International Advance Computing Conference. New York: IEEE Press, 2014: 636-640.

[12] GONG J, ZHANG H W. A Method for LOD Generation of 3D City Model Based on Extended 3D RTree Index[C]// Proceedings of 2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery. New York: IEEE Press, 2011: 2004-2008.

[13] LI J, JING N, SUN M Y. A Mechanism of Implementing Visualization with Level of Detail at Multiscale[J]. Journal of Software, 2002, 13(10): 2037-2043. (in Chinese)
李军, 景宁, 孙茂印. 多比例尺下细节层次可视化的实现机制[J]. 软件学报, 2002, 13(10): 2037-2043.

[14] DENG H Y, WU F, ZHAI R J, et al. R-Tree Index Structure for Multi-Scale Representation of Spatial Data[J]. Chinese Journal of Computers, 2009, 32(1): 177-184. (in Chinese)
邓红艳, 武芳, 翟仁健, 等. 一种用于空间数据多尺度表达的 R 树索引结构[J]. 计算机学报, 2009, 32(1): 177-184.

[15] ACHAKEEV D, SEIDEMANN M, SCHMIDT M, et al. Sort-based Parallel Loading of R-trees[C]// Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data. New York: ACM, 2012: 62-70.

[16] YOU S M, ZHANG J T, GRUENWALD L. Parallel Spatial Query Processing on GPUs Using R-Trees [C]// Proceedings of the 2nd ACM Sigspatial International Workshop on Analytics for Big Geospatial Data. New York: ACM, 2013: 23-31.

[17] TAO Y F, YANG Y, HU X C, et al. Instance level worst-case query bounds on R-trees[J]. The VLDB Journal, 2014, 23(4): 591-607.

[18] YIN K X, HUANG H, ZHANG H, et al. Morfit: Interactive Surface Reconstruction from Incomplete Point Clouds with Curve-Driven Topology and Geometry Control[J]. ACM Transactions on Graphics, 2014, 33(6): 1-12.

[19] SONG J, LI T T, ZHU Z L. Research on I/O Cost of MapReduce Join[J]. Journal of Software, 2015, 26(6): 1438-1456. (in Chinese)
宋杰, 李甜甜, 朱志良. MapReduce 连接查询的 I/O 代价研究[J]. 软件学报, 2015, 26(6): 1438-1456.

[20] WANG C K, MENG X F. Relational Query Techniques for Distributed Data Stream: A Survey[J]. Chinese Journal of Computers, 2016, 39(1): 80-96. (in Chinese)
王春凯, 孟小峰. 分布式数据流关系查询技术研究[J]. 计算机学报, 2016, 39(1): 80-96.

(上接第 146 页)

protocol with backward privacy[J]. Computer Science, 2016, 43(8): 128-130, 158. (in Chinese)
刘道微, 凌捷, 杨昕. 一种改进的满足后向隐私的 RFID 认证协议[J]. 计算机科学, 2016, 43(8): 128-130, 158.

[10] WEIS S A, SARMA S E, RIVEST R L, et al. Security and privacy aspects of low-cost radio frequency identification systems[M]// Security in Pervasive Computing. Springer Berlin Heidelberg, 2004.

[11] OHKUBO M, SUZUKI K, KINOSHITA. K Hash-Chain based forward secure privacy protection scheme for lowcCost RFID [C]// Proceedings of the 2004 Symposium on Crypto-graphy and Information Security(SCIS 2004). 2004: 719-724.

[12] COHEN M, DAM M. A completeness result for BAN logic[EB/OL]. [2011-06-22]. http://www.access.ee.kth.se/reports/2007/13.pdf.

[13] MOLNAR D, WAGNER D. Privacy and security in library RFID: issues, practices, and architectures[C]// Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS' 04). Washington, DC, USA, 2004: 210-219.

[14] RHEE K, KWAK J, KIM S, et al. Challenge-response based RFID authentication protocol for distributed database environment[C]// Proceedings of the 2nd International Conference on Security in Pervasive Computing (SPC 2005). Berlin: Springer-Verlag, 2005: 70-84.

[15] SHEN J, TAN H, ZHENG Y, et al. An enhanced ID-updating Hash-based RFID authentication protocol with strong privacy protection[J]. Frontiers in Artificial Intelligence & Applications, 2016, 274: 2070-2079.

[16] YUAN J S, XU Y, QI Y C, et al. Mutual authentication protocol for RFID based on asymmetric keys and hash function[J]. Journal of Cryptologic Reseatch, 2014, 1(5): 456-464. (in Chinese)
苑津莎, 徐扬, 戚银城, 等. 基于非对称密钥和 Hash 函数的 RFID 双向认证协议[J]. 密码学报, 2014, 1(5): 456-464.

[17] DING Z H, LI J T, FENG B. Research on Hash-based RFID security authentication protocol[J]. Journal of Computer Research and Development, 2009, 46(4): 583-592. (in Chinese)
丁振华, 李锦涛, 冯波. 基于 Hash 函数的 RFID 安全协议研究[J]. 计算机研究与发展, 2009, 46(4): 583－592.

[18] SAFKHANI M, PERIS-LOPEZ P, HERNANDEZ-CASTRO J C, et al. Protocol: a hash-based RFID tag mutual authentication protocol[J]. Journal of Computational & Applied Mathematics, 2014, 259(6): 571-577.

[19] JIN Y M, WU Q Y, SHI Z Q, et al. RFID lightweight authentication protocol based on PRF[J]. Journal of Computer Research and Development, 2014, 51(7): 1506-1514. (in Chinese)
金永明, 吴棋滢, 石志强, 等. 基于 PRF 的 RFID 轻量级认证协议研究[J]. 计算机研究与发展, 2014, 51(7): 1506-1514.