基于内容的图像检索中 SS-树索引的 Java实现

陈冬霞, 吉根林, 方昭辉

(南京师范大学 数学与计算机科学学院, 江苏 南京 210097)

[摘要] 为了提高基于内容图像检索的效率,设计引入空间索引. SS-树索引是常用的空间数据索引,具有平衡性、动态性、构造和维护的简单性,用 Java实现了 SS-树索引,并在自行开发的基于内容图像检索实验原型系统 IRS中使用. 给出了 SS-树索引文件和类的定义,介绍了主要操作和对应的算法实现. 将 SS-树索引用于基于内容图像检索的相似性检索,比较了顺序扫描和采用 R-树、SS-树进行检索的效率,实验结果表明采用 SS-树索引进行基于内容的图像检索是高效可行的.

[关键词] 基于内容的图像检索, Java, SS-树, 索引

[中图分类号] TP391. [文献标识码] B. [文章编号] 1672-1292(2005)04-0053-04

Realization of the SS-tree Index in Content-Based Image Retrieval Using Java

CHEN Dongxia, JI Genlin, FANG Zhaohui

(School of Mathematics and Computer Science, Nanjing Normal University, Jiangsu Nanjing 210097, China)

Abstract To improve the efficiency of the CBIR, the spatial index is needed. The SS-tree index is the popular spatial index, and has the characteristics of balance and dynamic; and it's easy to construct and maintain. A web-based CBIR system named IRS is realized by using Java, for which the SS-tree indexes are realized. The index documents and the definition of classes are proposed, and the main operation and the corresponding arithmetic are introduced. SS-tree index is used in the comparability index of content image retrieval. The efficiency of order scanning is compared with that of retrieving with R-tree and SS-tree. The experiment shows that with the use of index, the efficiency of CBIR is greatly improved. Using SS-tree index to retrieve is efficient and effective for CBIR.

Key words content-based image retrieval; Java; SS-tree; index

0 引言

如何快速准确地检索到所需的图像,已经成为迫切需要解决的问题. 传统的基于文本注释的图像检索方法,存在着手工注释图像工作量过大和注释主观差异性等缺陷,已不能适应需求. 因此,基于内容的图像检索(Content-Based Image Retrieval, CBIR)已成为研究热点. CBIR 通过直接对图像的内容进行分析,抽取内容特征,通过特征的相似性比较来检索图像. 当特征提取和相似度量方法确定后,要做的工作就是在图像数据库中查找与给定的检索图例最相似的图像. 最简单的检索方法是顺序扫描数据库中的所有图像,计算它们与检索图例的相似性. 但是,特征向量的相似性比较计算量非常大,顺序扫描检索的效率很低. 为了提高检索效率,需要适合 CBIR 的索引.

CBIR中使用的索引属于相似性索引(similarity index),主要目的就是减少检索的时间. 需要能够处理图像的高维特征数据,可以动态调整,而不用因为图象的插入或删除而重新建立. 因此,传统的一维索引不适用于 CBIR,而需要采用空间索引, SS-树是一种广泛使用的空间索引,本文用 Java实现了用于 CBIR 实验系统 IRS[1]的 SS-树索引.

1 SS-树索引

SS-树[3]是 R-树的改进,是一种对多维点数据进行相似检索的索引结构, SS-树的结构如图 1所示. 左图为空间节点分布图,右图为其对应的 SS-树结构示意图.

在自行开发的 CBIR 实验原型系统 IRS中的

收稿日期: 2005-09-20

作者简介: 陈冬霞(1978-),女,助教,主要从事数据库技术方面的教学与研究. E-mail: chendongxia@njnu.edu.cn

SS-树节点的数据结构如下:

```
class SSElem{
  Int immed_children;     //Children in array节点中直接孩子数
  Int total_children;     //Children in subtree结点中所有数据点数
  Int level;              //结点的层高,子结点为0
  Sphere sp;              //结点的边界,包括中心(Float centroid[DIM])和半径(Double radius)
  Int Parent_prt;         //指向父结点的指针
  Int[DATA_SIZE] Oid;     //标识/指针数组
  Sphere[DATA_SIZE] child_sp;  //孩子结点的边界数组
}
```

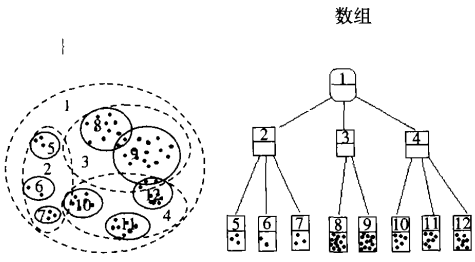

图1 SS-树结构示意图

SS-树保持了R-树索引良好的平衡性和动态性,与R-树相比,主要有以下不同:

(1) 边界形状: R-树采用超立方体对多维空间进行划分, SS-树采用了超球边界代替超立方体边界,节点只要存储中心坐标和半径,使节点的存储尺寸明显减少.

(2) 节点插入时选择路径的标准:树型索引中节点的重叠是造成检索效率下降的主要原因, R-树用增长容积作为防止或减小因插入节点造成重叠的判断标准, SS-树是用插入对象与超球中心的距离作为标准.

(3) 节点分裂策略:在分裂时, R-树是要实现分裂后的两个节点的边界的表面积和重叠部分的体积最小化;而SS-树是要实现两个节点中心的数据的方差最小化,所以选择在两个方差最大的点作为新节点的中心.

(4) 节点的重插入: SS-树延续并改进了R^*-树使用重插入策略来减少节点间的重叠的方法.

根据文献[4]的实验和研究,正因为SS-树有这些不同点,它能更好地聚集相似的数据,整体性能较R-树有明显的提高.

2 SS-树索引的Java实现

2.1 SS-树索引文件和类的定义

IRS系统中定义的索引文件分为文件头和文件体两部分.文件头用于对索引文件的总体进行描述和控制,文件体按R-树的方式组织空间对象,如图2所示.

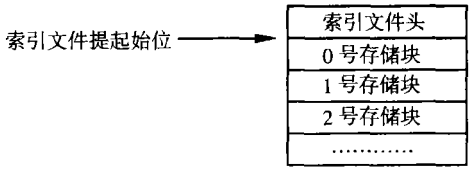

图2 IRS系统中SS-树索引文件的存储结构示意图

索引文件体部分划分为长度相等的若干个存储块,并对这些存储块从0开始进行编号,每个存储块用于存放一个树节点,其编号也就是节点在文件中定位的ID.本文使用Java实现索引, Java的主要特点之一就是不支持指针.因此,将此ID作为节点的指针使用.为实现SS-树索引,主要定义的类包括:

(1) AbstractNode定义节点的抽象类,包含节点的属性.

(2) Index实现中间节点的类,包括节点的插入、删除、分裂、路径选择和节点调整等算法的实现.

(3) Leaf实现叶节点的类,包括特征数据的插入、删除、页节点分裂算法的实现.

(4) Point定义图像特征数据点的类.

(5) Sphere实现节点的覆盖区域边界的类,包括边界间包含、相交判断的拓扑操作算法,计算MINDIST和MINMAXDIST算法的实现.

(6) Tree实现树的类,包括树的建立、遍历、k-NN相似性检索算法的实现.

(7) PageFile实现把索引写入文件操作的类,负责把树的节点信息保存为文件的内容,并向树提供节点在文件中的位置标识,作为节点的指针使用.

以下是2维的SS-树在索引文件中保存的数据实例. Level为0的节点为叶节点,他们的父节点为根节点.叶节点中数据项指针指向的是图像在数据库中的序号,父节点中数据项指针指向的是孩子节点的ID.

```
< Page 1, Level 0, UsedSpace 2, Parent 0>
  1) Center(30.0, 10.0): radius 0.0--> page 6
  2) Center(40.0, 70.0): radius 0.0--> page 4
< Page 2, Level 0, UsedSpace 2, Parent 0>
  1) Center(70.0, 15.0): radius 0.0--> page 8
  2) Center(13.0, 21.0): radius 0.0--> page 18
< Page 3, Level 0, UsedSpace 4, Parent 0>
  1) Center(0.0, 50.0): radius 0.0--> page 14
```

2) Center(10.0, 20.0): radius 0.0- - > page 2
3) Center(30.0, 55.0): radius 0.0- - > page 16
4) Center(3.0, 8.0): radius 0.0- - > page 22
< Page 4, Level 0, UsedSpace 3, Parent 0>
1) Center(110.0, 80.0): radius 0.0- - > page 12
2) Center(100.0, 70.0): radius 0.0- - > page 10
3) Center(54.0, 78.0): radius 0.0- - > page 20
< Page 0, Level 1, UsedSpace 4, Parent -1>
1) Center(35.0, 40.0): radius30.413812651491 1- - > page 1
2) Center(41.5, 18.0): radius28.6574597618142- - > page 2
3) Center(10.75, 33.25): radius29.0452233594441 47- - > page 3
4) Center(88.0, 76.0): radius 34.0587727318528 05- - > page 4

2.2 主要操作及算法实现

2.2.1 插入操作

(1) 寻找合适的叶结点 L 来存放新插入的空间对象 O.

(2) 将新的空间对象记录到叶结点 L 中, 如果 L 不满, 就直接插入. 如果 L 满了, 对当前溢出的结点 L 进行重插入或分裂.

(3) 调整树的结构, 从叶结点 L 开始, 对树结构进行调整.

(4) 若根结点也溢出, 则分裂之并使树高增 1.

选择合适的叶结点:

(1) 初始化, 设 N 为根结点.

(2) 如果 N 为叶结点, N 就是要找的叶结点.

(3) 选择合适的子树: 如果 N 不是叶结点, 计算 O 与当前结点的每个子结点中的中心 C_i 的距离 $d(O, C_i)$, 选择最近的子树作为路径. 如果有多个距离相同的点, 选择其子树最少的点. 对选中的路径结点, 更新中心、半径、孩子数等属性. 令 N 为这个子结点, 从 (2)开始重复进行上面的步骤.

节点的重新插入:

当一个结点在插入过程中发生了溢出时, 并不急于进行分裂, 而是对结点中的单元重新判断, 并在索引树中同层的其他结点中进行动态调整, 以推迟结点的分裂, 这样使得结点的 MBR 间重叠较小, 获得较高的结点存贮利用率, 有时还可以避免结点的分裂. 当结点 N 溢出时, 若 N 是所在层的第一个溢出结点, 则调用重新插入过程, 选择结点 N 中若干个适当的单元重新插入. 反之, 则分裂结点 N. 重新插入机制是指从结点 N 中选择若干个适当的单元, 重新插入到同层的其他结点中. 选择重插单元的规则是: 计算 N 中的 $M+1$ 个单元的 MBR 的中心到结点 N 的 MBR 的中心的距离 D; 将各单元按 D 值从大到小排列, 取出前 P ($1 \leqslant P \leqslant M-m+1$) 个单元作为重插单元.

调整树的结构:

(1) 初始化: 设 N 为叶子结点 L, N'为因插入 O 而分裂生成的新结点 (如果有的话).

(2) 如果 N 为根结点, 则停止, 返回.

(3) 调整父结点中相应单元的 I: 调整结点 N 的父结点 P 中与其相应单元的 I.

(4) 根据需要进一步分裂父结点: 如果存在因分裂生成的新结点 NN, 则在 N 的父结点 P 中加入指向该结点的单元. 如果 P 结点中单元个数超过 M, 则需分裂结点 P, 从而生成一个新的结点 PP, 令 $N=P$, $NN=PP$. 从 (2)步开始重复上面的步骤.

2.2.2 删除操作

(1) 首先通过检索算法, 确定删除对象所在的叶子结点 L.

(2) 将结点 L 中空间对象所在的单元删除.

(3) 从叶子结点 L 开始调整, 保证结点 (根结点除外)中单元数不低于下限 m.

(4) 调整根结点, 如果调整后, 根结点只有一个孩子, 则将根结点删除, 让其孩子结点成为根结点, 树减少一层.

为了保证每个结点 (除根)的单元数不小于 m, 调整算法:

(1) 初始化: 令 N 为被删除单元的叶子结点 L. Q 为结点的集合, 用以存放被删除的结点, 初始化为空.

(2) 寻找 N 在父结点中的相应单元: 如果 N 为根结点则转至步骤 (6), 否则查找其父结点 P, 并找到结点 N 相对应的单元 E_N.

(3) 删除单元数小于 m 的结点: 如果结点 N 的单元数小于 m, 则在结点 P 中删除单元 E_N, 并将节点 N 加入集合 Q 中.

(4) 调整相应的结点中的数据, 如中心、半径、孩子个数等.

(5) 调整上一层: 令 $N=P$, 从步骤 (2)开始重复上述操作.

(6) 采用插入算法, 将 Q 中的所有被删除的节点重新插入到树中.

删除指定的实体对象 O, 这是唯一可能减小树的高度的操作. 当由于实体的删除导致结点 L 下溢即所含实体数小于 m 时, 先将整个结点 L 删去, 再将 L 中除 O 以外的其它实体重新插入到树中.

2.2.3　分裂

当需向一个已包含 M 个实体的结点 L 中插入新实体对象时,将导致结点的分裂,此时需将 $M+1$ 个实体分裂到两个结点 L 和 L' 中.选择两个最大的点作为新节点的中心,然后将剩余点分配到距离最近的中心的节点中.从而实现两个节点中心的数据的方差最小化.

3　SS-树索引用于 CBIR 的相似性检索

基于内容的图像相似性检索,通常分为两类:一种是给定相似度和示例图像,进行相似性范围检索(range search);另一种是对给定的示例图像,返回最相似的 k 个图像的 k-最近邻检索(k-Nearest Neighboring Search).相对于需要提供相似度的范围检索,k-最近邻检索更符合用户的使用习惯.因此我们使用 k-最近邻检索算法.

Rousspoulos等[5]提出了分支边界 R-树遍历算法以检索给定点的最近邻,SS-树的 k-NN 算法与基于 R-树的 k-NN 算法基本相同,只需修改用于剪枝的两个距离 MINDIST 和 MINMAXDIST[4],其定义如图 3所示,具体算法见文献[1].

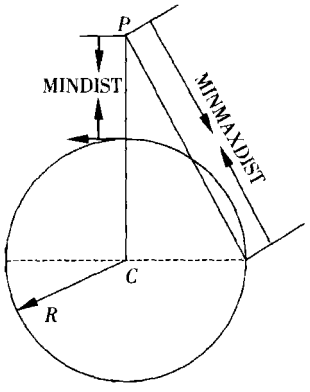

图 3　SS-树的 MINDIST 和 MINMAXDIST 示意图

公式定义如下:

$$MINDIST = (d(P, C) - R)^2 \quad (1)$$

$$MINMAXDIST = d(P, C)^2 + R^2 \quad (2)$$

4　实验与结果

实验比较了顺序扫描检索和采用 R-树、SS-树进行检索的效率.实验环境:Web服务器和数据库服务器均采用 Dell OptiPlex Gx400 P4的 PC机,CPU 1.5GHz,256M 内存.从 Web 上收集 JPG 和 GIF图像作为实验数据,使用颜色直方图表示的颜色特征作为图像内容特征,相似性度量使用欧氏距离.进行随图像数据量变化和随索引维数变化的情况下,20-NN 检索的效率对比实验.索引中单元数的限制 $m=2$ $M=4$ 采用 IRS系统中基于浏览系统图像选择图例的方式,对每种检索方式检索相同的 10幅图像,每幅图像的检索时间取 10次检索时间的平均值.

实验表明,使用 3种方式检索获得的 CBIR检索结果一致.检索效率对比效果如图 4和图 5所示.随图像数据量变化的实验中,索引是 8维的,每次检索图像数量递增 100幅,顺序扫描的检索时间增长很快,R-树和 SS-树的变化趋势相似,但 SS-树的检索时间低于 R-树.随图像特征维数变化的实验中有 400幅图像,6次实验的特征维数依次为 2、4、8、16、32、64,使用索引的检索效率明显优于顺序扫描,SS-树的检索效率优于 R-树.

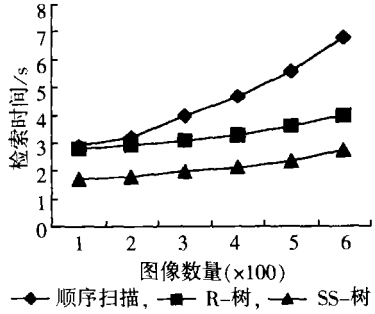

图 4　随图像数量变化的检索效率比较实验结果

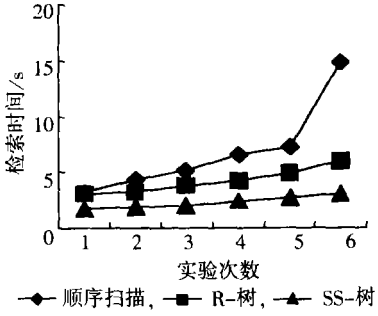

图 5　随特征维数变化的检索效率比较实验结果

5　结语

由于 CBIR中的图像集合具有海量性、高维性的特点,顺序扫描检索的效率很低.需要合适的多维索引组织特征向量数据,提高检索的效率.CBIR正成为图像检索研究领域的热点,通过各种特征提取和度量方法的研究,可以不断提高检索的有效性.本文用 Java实现了 SS-树索引,并在 CBIR实验原型系统 IRS中使用.实验结果表明采用索引进行基于内容的图像检索是高效可行的.相信随着更多的科研工作的开展,在不久的将来索引技术将会在 CBIR的应用中发挥更大的作用.

(下转第 81页)

设. 如何实现 GPS基站的构成和合理配置, 通过站网建设提高调查数据空间点位精度, 有待进一步研究.

在市区和城郊结合部有关地带, 可能存在 GPS 信号的盲区, 因此, 后续的研究还需要实现 GPS-PDA土地变更调查数据采集系统与常规测量仪器组合, 进一步扩展本系统在城郊结合部土地利用现状变更调查中的应用.

[参考文献]

[1] 冯宝红, 郑小元, 王庆. GPS-PDA在土地变更调查中的应用 [J]. 测控技术, 2003, 22(8): 23-25.
[2] 孙彩敏, 吴亚光, 武新明. GIS型 GPS接收机在土地利用数据库变更中的应用 [J]. 地矿测绘, 2001, 17(4): 20-22.
[3] Zhong Sidong, Li Deren, He Saixian, *et al*. LD2000 system with 3S and multi-sensor [J]. Geo-spatial Information Science, 2002, 5(1): 74-78.
[4] 付永吉, 聂志锋, 郝彤途, 等. 基于 WINDOWS CE掌上电脑对 GPS OEM 板的控制和数据处理 [J]. 全球定位系统, 2002(2): 23-27.
[5] 尤红建, 李树楷. GPS采集 GIS数据的原理及应用 [J]. 测绘科学, 1997(1): 35-39.
[6] 杨雪峰, 刘力. GPS与 GIS集成初论 [J]. 干旱区地理, 2000, 23(4): 376-380.
[7] 李夕银. GPS在 GIS数据采集中的应用 [J]. 测绘通报, 2002(5): 23-28.

[责任编辑: 严海琳]

(上接第 56页)

[参考文献]

[1] 方昭辉. 基于内容的图像检索中索引的研究与实现 [D]. 南京: 南京师范大学, 2004.
[2] 方昭辉, 陈冬霞. 用 Java实现分布式基于内容的 Web 图像检索系统 [J]. 南京师范大学学报: 工程技术版, 2004, 4(1): 60-63.
[3] White D A, Jain R. Similarity indexing with the SS-tree [C] // Proc 12th IEEE Int Conf on Data Engineering. New Orleans, 1996: 516-523.
[4] Wang S, Hellerstein J, Lipkind I. Near-neighbor query performance in search trees [C] // Technical Report California: UC Berkeley, 1998: CSD-98-1012.
[5] Roussopoulos N, Kelly S, Vincent F. Nearest neighbor queries [C] // Proc ACM SIGMOD Int Conf on Management of Data. California: San Jose, 1995: 71-79.
[6] Zhang H J, Zhong D. A Scheme for visual feature-based image indexing[C] // Proc of SPIE Conf on Storage and Retrival for Image and Video Databases III. California: San Jose, 1995: 36-46.

[责任编辑: 刘 健]