

文章编号:1001-9081(2005)11-2624-03

叶结点编码四叉树的邻域寻找算法

吴恒山,段雄文,李晨阳

(华中科技大学 计算机科学与技术学院,湖北 武汉 430074)

(dxw1231@etang.com)

摘 要:设计了一套新的叶子结点编码方式,基于该编码,实现了编码四叉树的邻域寻找。此方法减少了四叉树存储的结点个数,提高了存储效率。同时由于在叶子一级采用位操作实现邻域寻找,使查询效率有所提高。

关键词:四叉树;叶子结点编码;邻域寻找

中图分类号:TP311.12 **文献标识码:**A

Algorithm for neighbor searching of leaf-coding quadtree

WU Heng-shan, DUAN Xiong-wen, LI Chen-yang

(College of Computer Science & Technology, Huazhong University of Science and Technology, Wuhan Hubei 430074, China)

Abstract: A new leaf-coding algorithm in quadtree was designed. Based on the coding, the algorithm for neighbor searching in leaf-coding quadtree was implemented. The method enhanced the storage efficiency by the decrease of quantity of nodes, and it enhanced the query efficiency by doing the neighbor searching with bitwise operating available at the leaf node level.

Key words: quadtree; leaf-coding; neighbor searching

0 引言

四叉树是表示二值图像的重要方法,在计算机图形学、图像处理、计算机视觉、机器人等领域得到广泛应用。将一幅二值图像递归分解,用四叉树来表示,根结点表示整幅图像,叶结点表示黑或白的四分区,非叶结点表示灰四分区,黑分区、白分区、灰分区对应的结点分别称为黑结点、白结点和灰结点。^[1,2]

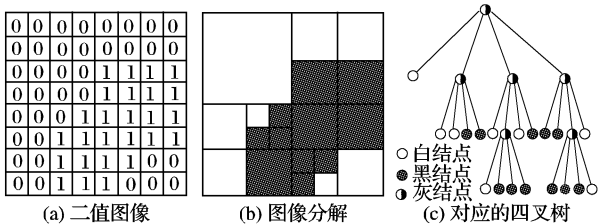


图 1 二值图像及其图像分解与四叉树表示

如果将叶子结点进行编码,就可以减少四叉树的结点,提高存储效率,但这样会影响四叉树的邻域查询。针对此问题,本文设计了一套新的叶子结点编码方式,基于该编码,结合线性四叉树编码的特点^[3]与显式四叉树的邻域寻找^[4,5]设计出一种基于叶子结点编码^[6,7]的四叉树的邻域寻找算法(Neighbors Searching based on Leaf-Coding, NSLC),将四叉树结构与编码结合,减少了四叉树要存储的结点数,提高了存储的效率。同时由于在叶子一级采用位操作实现邻域寻找,使查询效率有所提高。

1 NSLC 算法的基本定义

NSLC 用一系列预定义的尺寸可变的子图像作为四叉树中的一个叶结点来表示二值图像。考查四分区后的 4 个子分

区,它们是黑、白两种子分区的组合(包括全黑、全白的分区),因此以这些组合为模型作为 NSLC 的叶结点比单纯用黑或白两种分区作为叶结点能节省叶结点,降低四叉树的高度,从而达到节省存储空间的目的。

1.1 NSLC 模型的定义与性质

定义 模型是 $2^k \times 2^k$ (k 是四叉树的层次,为非零正整数)的子图像,或者是一个单色(黑或白)区域,或者是黑和白两种区域的组合。

不同尺寸的模型除了尺寸不同外,组合形式都是相同的。一个模型与 NSLC 中的一个叶结点对应,该叶结点称为 NSLC 的模型结点(Model Node, MN)。

性质 NSLC 中同一尺寸的模型共有 16 种。

证:同一尺寸的模型对应同一尺寸的二值图像区域,该区域被划分为 4 个子分区,它们是单色分区时才对应模型。4 个子分区中有 0, 1, 2, 3, 4 个黑结点时的模型数分别为 C_4^0 , C_4^1 , C_4^2 , C_4^3 和 C_4^4 , 故同一尺寸的模型共有 $C_4^0 + C_4^1 + C_4^2 + C_4^3 + C_4^4 = 16$ 个。NSLC 的构造过程与其他四叉树相同。若给定一幅 $2^n \times 2^n$ 的二值图像与 $2^n \times 2^n$ 尺寸的模型匹配,则整幅图像用与它匹配的模型结点表示。否则将它四分区,并测试这 4 个子分区是否与它下一级尺寸的模型匹配。递归执行该过程,直到整幅图像被分解成不同尺寸的模型为止。

模型的这个性质决定了 NSLC 的模型结点有 16 个。

1.2 NSLC 模型结点的设计

本文参考线性四叉树的编码规则^[3]对模型结点进行编码,编码方向按 NW, NE, SW, SE 方向进行,如图 2(a) 所示。

对图 2(b) 中的模型结点用二进制表示编码就是 0101, 用十六进制表示就是 $(0101)_2 = (5)_{16}$, 这样 NSLC 中同一尺寸

收稿日期:2005-05-27

作者简介:吴恒山(1953-),男,湖北武汉人,副教授,主要研究方向:数据库、多媒体技术;段雄文(1980-),男,湖北孝感人,硕士研究生,主要研究方向:数据库、多媒体技术;李晨阳(1975-),男,湖北武汉人,博士研究生,主要研究方向:数据库、多媒体技术。

所有可能的模型、原二叉树的灰结点、NSLC 模型结点的十六进制编码及模型结点的表示见图 2(c)。

1.3 NSLC 模型对查询的支持

1) NSLC 模型对点查询的支持

NSLC 在结点一级用了模型编码,如果将模型结点看成是普通的结点,那么这个模型结点的查询就与普通二叉树的结点的查询无异。要查询模型结点内的某一点,则可以通过本 NSLC 里的 IsExist 算法来确定,这样就解决了点查询问题。

2) NSLC 模型对邻域查询的支持

本文主要是设计基于叶子结点编码的二叉树的邻域寻找,它的结构是在普通的二叉树的基础上进行改进,它的邻域寻找也只是要在普通二叉树的寻找上,在模型结点一级进行相关的设计。

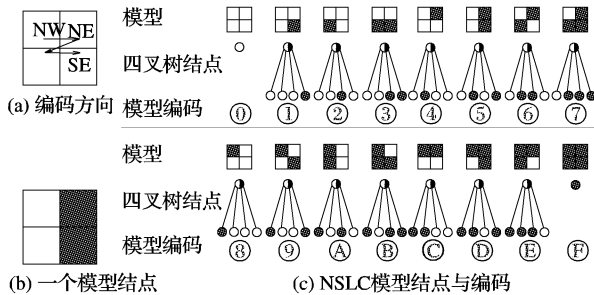


图 2 NSLC 模型编码

2 基于编码的邻域查询算法: NSLC

本文根据基于叶子结点编码的二叉树的模型结点的编码特点,结合传统的显式二叉树的邻域寻找方法,设计出 NSLC。在对 NSLC 算法中先不考虑结点的面积与类型,用公共祖先法进行同一级结点的邻域寻找,在模型结点一级利用编码的位运算进行其区域的确定与邻域的寻找。

2.1 NSLC 算法基础

2.1.1 显式二叉树的邻域寻找方法

该公共祖先方法基于 Samet H 等提出的指针二叉树镜面反射的思想^[1,2],不考虑结点 P(要查找其邻接点的结点)的位置(即坐标)和面积。该方法只用到二叉树的树结构。

我们来看如何寻找结点 P 的 I 边的邻接点 Q(Q 与 P 的大小相等)。算法中用到了两个函数 ADJ 和 REFLECT,其定义如表 1 和表 2。

表 1 ADJ(I,O)

I(方向)	O(象限)			
	NW	NE	SW	SE
N	T	T	F	F
E	F	T	F	T
S	F	F	T	T
W	T	F	T	F
NW	T	F	F	F
NE	F	T	F	F
SW	F	F	T	F
SE	F	F	F	T

表 2 REFLECT(I,O)

I(方向)	O(象限)			
	NW	NE	SW	SE
N	SW	SE	NW	NE
E	NE	NW	SE	SW
S	SW	SE	NW	NE
W	NE	NW	SE	SW
NW	SE	SW	NE	NW
NE	SE	SW	NE	NW
SW	SE	SW	NE	NW
SE	SE	SW	NE	NW

下面描述了主要的算法:

1) FindAncestor, 寻找最近公共祖先。如果结点 P 的某祖先结点使 ADJ(I,O) 为真,就沿二叉树上升,找层次更高的祖先结点。当被考查的祖先结点使 ADJ(I,O) 为假时,它的父结点就是结点 P 与结点 Q 的最近公共祖先。

2) FindOffspring, 寻找邻接点 Q。从结点 P 和结点 Q 的

最近公共祖先开始,沿二叉树向下移动(下降时的路径用 REFLECT 函数计算)。下降路径与上升路径沿 P 和 Q 的公共边 I 对称。

2.1.2 NSLC 模型结点一级的算法

NSLC 模型结点一级的算法,是本文的设计重点,要设计判断这个模型结点的某个区域是否存在(IsExist 算法),灰结点中子结点的寻找(InGray 算法),以及模型结点内与黑结点相邻区域的寻找(InModel 算法),其中 IsExist 可以支持模型结点的点查询。下面先介绍这些算法:

Exist(P,side) 对一个模型结点 P,求 side 方向的区域是否存在:

IF(P 编码 \oplus 1000! = 0) 则 side = NW 方向区域存在;

IF(P 编码 \oplus 0100! = 0) 则 side = NE 方向区域存在;

IF(P 编码 \oplus 0010! = 0) 则 side = SW 方向区域存在;

IF(P 编码 \oplus 0001! = 0) 则 side = SE 方向区域存在。

其中 \oplus 是按位异或(bit-by-bit Exclusive OR)。

InGray(P,Q,side) 已知点黑结点 P 与灰结点 Q 等尺寸相邻,欲求灰结点与 P 在 P 的 side 方向相邻的叶子结点:

Q 是 P 的 N 邻域,即 side = N 时,求 Q 的 SW,SE 子结点及 SW,SE 子结点的 SW,SE 子结点,直到叶子结点;

Q 是 P 的 S 邻域,即 side = S 时,求 Q 的 NW,NE 子结点及 NW,NE 子结点的 NW,NE 子结点,直到叶子结点;

Q 是 P 的 W 邻域,即 side = W 时,求 Q 的 NE,SE 子结点及 NE,SE 子结点的 NE,SE 子结点,直到叶子结点;

Q 是 P 的 E 邻域,即 side = E 时,求 Q 的 NW,SW 子结点及 NW,SW 子结点的 NW,SW 子结点,直到叶子结点。

InModel(P,R,side) 已知黑结点 P 与同尺寸或比它小的模型结点 R 相邻,可以进一步判断在 R 内是否有区域与已知点真正相邻,按算法 IsExist 可求出与其真正相邻的区域:

求这个模型结点是否是已知点的 N 方向的邻域,即 side = N 时,则求其 S 方向的区域是否存在,即求模型结点 SW,SE 方向区域是否存在;

求这个模型结点是否是已知点的 S 方向的邻域,即 side = S 时,则求其 N 方向的区域是否存在,即求模型结点 NW,NE 方向区域是否存在;

求这个模型结点是否是已知点的 W 方向的邻域,即 side = W 时,则求其 E 方向的区域是否存在,即求模型结点 NE,SE 方向区域是否存在;

求这个模型结点是否是已知点的 E 方向的邻域,即 side = E 时,则求其 W 方向的区域是否存在,即求模型结点 NW,SW 方向区域是否存在。

2.2 NSLC 算法描述

NSLC 包括两种情况,它们调用上面所设计的算法,来实现对 NSLC 的邻域寻找。

第一种情况 二叉树中编码为 F 的模型结点 P 的邻域寻找

1) 如果按照 FindAncestor, FindOffspring 算法还没有完全下降到与 P 同层,就已经是 F 结点,则可以知道这个 F 结点就是要找的比 P 大的区域结点。

2) 如果按照 FindAncestor, FindOffspring 算法找到 P 的邻域是同一层的 F 结点,则可以知道这个结点就是要找的与 P 同尺寸的区域结点。

3) 如果按照 FindAncestor, FindOffspring 算法找到 P 的

邻域是同一层的结点且是模板结点中的非 F 结点,则可以根据算法 InModel 判断此模板结点中是否有区域与 P 相邻,且找出是哪一个区域,找到的区域尺寸小于 P 的尺寸。

4) 如果按照 FindAncestor, FindOffspring 算法下降到与 P 同层后只是得到一个灰结点,则按照算法 InGray 求灰结点可能与 P 相邻的子结点。然后按照 InModel 求这些子结点中与 P 相邻的真正区域。

第二种情况 四叉树中编码不是 F 的模型结点 P 的邻域寻找

1) 只求与 P 相邻的结点

按照 FindAncestor, FindOffspring 算法还没有完全下降到与 P 同层,就已经是 F 结点,则可以知道这个 F 结点就是要找的比 P 大的区域结点;按照 FindAncestor, FindOffspring 算法下降找到与 P 同层的结点。

2) 求 P 内某区域的相邻的结点

按照算法 IsExist 求出自己想求的点是否存在,不存在则算法结束。如存在,则可以继续算法,进行 P 内 NW 区域结点的邻域寻找:如果 NW 方向区域存在即 IsExist(P, NW) 为真,想求其 S 方向和 E 方向的邻域,则可以在块内求得:

S 方向 IsExist(P, SW) 为真,则此结点的 SW 方向区域就是所求,反之则无。

E 方向 IsExist(P, NE) 为真,则此结点的 NE 方向区域就是所求,反之则无。

N 方向 按步骤 1) 求出此模型结点的 N 方向的邻域或其结点:如果还未有下降到与 P 相同层就已经是 F 结点,则 F 即为所求;如果到了同层是 F 结点,则 F 结点为所求;如果到了同层是模型结点 R,只需要验证此 IsExist(R, SW) 是否为真,为真则此区域为与所求结点 P 的 NW 相邻的同尺寸区域;不为真则没有求得相邻区域;如果到了同层后只是灰结点,只要求此 SW 方向的子结点的是否有与其相邻的子区域,这时就类似第一种情况中求同尺寸邻域结点是否有相邻区域的情况,即第一种情况中 2) ~ 4) 的情况。

W 方向的求法与 N 方向类似。

P 内 NE, SW, SE 区域结点的邻域寻找与 NW 区域结点的邻域寻找类似。

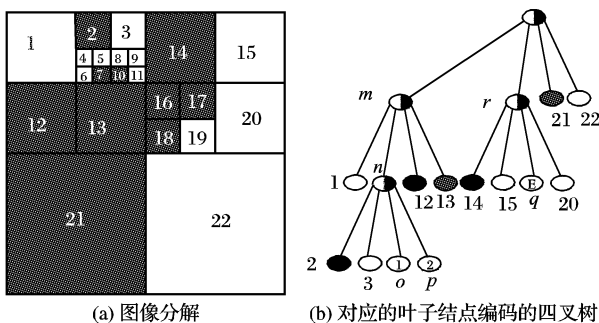


图3 图像分解与其对应的叶子结点编码的四叉树表示

2.3 NSLC 算法分析

1) 空间效率

采用叶子结点编码的四叉树的结点个数(N_t)是灰结点(G_t)和模型结点(P_t)的个数之和,即 $N_t = G_t + P_t$,且 $N_t = 4G_t + 1$,由 $G_t + P_t = 4G_t + 1$,可求得 $G_t = (P_t - 1)/3$,因此, $N_t = 4G_t + 1 = 4(P_t - 1)/3 + 1 = (4P_t - 1)/3$;而一般四叉树的叶子结点的个数 N 可以由其灰结点数 G 求得,即 $N = 4G + 1$ 。这样 $N - N_t = 4(G - G_t)$ 。如果四叉树的灰结点中平均有

一半可转成模型结点,即 $G_t = G/2$,那么, $N - N_t = 2G$ 。这说明在平均情况下采用叶子结点编码的四叉树比普通四叉树减少 $2G$ 个结点。

2) 邻域查询效率

对于一幅 $n \times n$ 维图像,为确定一区域的邻域就可能需要进行 $n^2(n^2 - 1)$ 次判断,显然这种方法的效率较低。本文的 NSLC 如果有一半的灰结点可以转换成模型结点,那么少存储 $2G$ 个结点,这样即使在最坏的情况下,为确定一区域的邻域要进行的判断是 $(n - 2G)^2((n - 2G)^2 - 1)$ 次。至于模型结点内部的邻域判断,只需要扫描模型代码,进行位运算就可以完成。

3) 实验对比分析

如果分别用线性四叉树^[8],插值二叉树^[9]与本文的叶子结点编码四叉树来表示如图3所示的一个图像,可以比较它们的邻域寻找时间复杂度与需要存储的结点的个数。图4中的邻域查找时间对比的纵坐标是将时间复杂度进行一定的量化后得到的值;需要表示的结点数的图中,线性四叉树的结点数用 $2^{2L+1} - 1$ 来计算(L 表示层数),插值二叉树的结点数用 $(4^{L+1} - 1)/3$ 来计算。

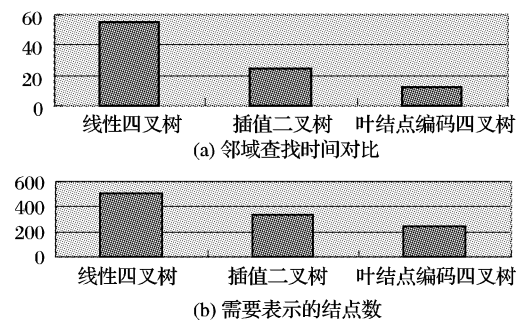


图4 三种算法表示图3的时空对比

由此可见,与同类的几种结构相比,本文设计的这种叶子结点编码的四叉树邻域寻找算法,不仅使四叉树所要存储的结点变少,还可以提高邻域寻找的速度。

参考文献:

- [1] SAMET H. Applications of spatial data structure: Computer Graphics, Image Processing, and GIS[M]. Addison-Wesley, 1990.
- [2] SAMET H. The design and analysis of spatial data structures: Computer graphics, Image Processing, and GIS[M]. Addison-Wesley, 1990.
- [3] SHAFFLE CA, SAMET H. Optimal quadtree construction algorithms[J]. Computer Vision, Graphics and Image Processing, 1987, 37: 402 - 419.
- [4] HUNTER GM, STEIGLITZ K. Operation on images using quadtree[J]. IEEE Transactions on Model Analysis and Machine Intelligence, 1979, 2(1): 145 - 153.
- [5] SAMET H. The quadtree and related hierarchical data structures[J]. ACM Computer Surveys. 1984, 16: 187 - 260.
- [6] GOLCHIN F, PALIWAL KK. Quadtree-based classification in sub-band image coding[J]. Digital Signal Processing, 2003, 13: 656 - 668.
- [7] BARJERSKI P, FRICZEK J, MROZEK D. Spatial Distribution Query Language[A]. International Symposium on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences[C], 2004.
- [8] 肖乐斌, 龚建华, 谢传节. 线性四叉树和线性八叉树邻域寻找的一种新算法[J]. 测绘学报, 1998, 27(3): 195 - 203.
- [9] 刘钦, 晏明辉. 二值图像邻域寻找的一种快速方法[J]. 计算机应用与软件, 1997, (5): 32 - 36.