基于格矢量量化和倒排文件的快速图像检索方法

陈学青,罗航哉,薛向阳,吴立德
(复旦大学计算机系,上海 200433)

摘 要: 本文提出一种新的快速图像检索方法,它用格矢量量化器对特征矢量进行量化和描述,用倒排文件和 Hash 表存储和索引量化后的特征矢量,利用代数格良好的几何和代数性质实现快速检索. 最后,本文分析了新方法的检索性能,并给出实验结果.

关键词: 检索; 索引; 矢量量化
中图分类号: TP391 **文献标识码**: A **文章编号**: 0372-2112 (2000) 08-0094-03

Fast Image Retrieval Method Based on Lattice Vector Quantization and Inverted File

CHEN Xue-qing, LUO Hang-zai, XUE Xiang-yang, WU Li-de
(*Department of Computer Science, Fudan University, Shanghai* 200433, *China*)

Abstract: This paper first proposes a novel and fast image retrieval framework. In this method, lattice vector quantizer is adopted to quantize and code the feature vectors extracted from images, and the inverted file and Hash table are used to store and index the quantized feature vectors. Fast query can be implemented because of the good geometric and algebraic properties of the lattice. In the end, the performance is analyzed theoretically, and the experimental results are presented.

Key words: information retrieval; index; vector quantization

1 引言

近几年来,人们提出了基于内容的多媒体信息检索新思路,企图利用听觉和视觉的内容实现检索. 国外出现了许多系统: (1) IBM 研制的 QBIC 系统支持以下检索功能:基于样本图像的查询、用户构画草图、用户绘制图形、用户选择期望的纹理和颜色. 在 QBIC 中,用颜色直方图、改进的 Tamura 纹理特征和形状特征等实现图像检索,它是少数几个考虑高维数据索引的检索系统之一; (2) Virage 公司开发的基于内容的图像搜索引擎,它支持基于颜色、颜色布局、纹理和结构等视觉信息的检索; (3) MIT 媒体实验室研制的一组交互式浏览和检索工具,它实现了形状、纹理和人脸特征的抽取和检索.

从绝大多数基于内容的图像检索系统来看,它们首先都是从图像中提取颜色、纹理、形状和布局等特征,然后在这些视觉特征基础上计算图像之间相似度,从而实现基于内容的图像检索. 为了实现快速检索,最常见方法是建立索引。到目前为止,人们陆续提出了多种索引技术[4,5]:B 树、倒排文件、K-d 树、MB 树、R 树及其变种、FastMap 和 VP 树等,这些技术可以用于精确匹配查询和相似查询. 现在研究和使用较多的是 R 树,它最早由 Guttman 提出,适合于对点数据和区域数据进行索引.

本文提出一种新的、适合于对点数据进行存储、索引和检索的一般性方法,它实际上是格矢量量化、倒排文件和 Hash 三种技术的有机结合. 下面分别从二个方面进行介绍,即:如何存储和索引点特征数据和如何实现快速检索.

2 点数据的存储与索引

假定从一幅图像中能够提取出一个 D 维特征矢量 x,那么从 N 幅图像中可以得到 N 个矢量: $\Theta = \{ x_i, 1 \le i \le N \}$. 假设两幅图像之间相似度可以用矢量之间欧氏距离度量,那么存储和索引这些矢量(又称点数据)的方法如下:

2.1 格矢量量化

用格矢量量化将 Θ 中每个矢量 x_i 进行量化,即: $q_i = LVQ(x_i)$,其中 $LVQ()$ 表示将 x_i 量化为 D 维空间中的格点 q_i. 这样,就得到量化后的结果 $\Theta = \{ q_m | q_m = LVQ(x_i), x_i \in \Theta, 1 \le m \le M \}$,因为可能有多个矢量量化为同一格点,所以通常 $M \le N$. 从矢量量化观点来看, $LVQ()$ 将整个数据空间划分为若干大小相同的 Voronoi 胞腔,胞腔的质心就是格点. 因为格的良好代数和几何特性,所以基于格的矢量量化通常具有很低复杂度[1~3].

2.2 存储离散格点

表 1 给出基于倒排文件的数据结构,它能存储 Θ 和对应的图像序号 id(在图像数据库中,每幅图像对应一个唯一的序

收稿日期:1999-05-14;修回日期:1999-09-15
基金项目:自然科学基金(No. 69602003);863 项目;自然科学基金重点项目

号). 如果按 Hash 表方式组织和存储表 1(见图 1),那么就能利用 Hash 方法实现快速键值的查找(当然也可以用二分搜索或 Trie 树实现快速检索!). 在图 1 中,Hash 函数将键值 q_i ($q_i \in \Theta$) 映射为一个整数 p,该整数 p 取值范围是 $[0, P-1]$,其中 P 是一个素数. 数组 U 存放指向单向链表的指针(允许空指针),链表中包含一个或多个节点,每个节点的数据结构是:键值 q_i、对应同样 q_i 的图像数目 l_i、相应的 l_i 个图像序号和指向下一个节点的指针.

表 1　用倒排文件存储格点和图像序号

键值	图像序号集合
q_1	$id_{11}, id_{12}, \cdots, id_{1l_1}$
q_2	$id_{21}, id_{22}, \cdots, id_{2l_2}$
…	…
q_M	$id_{M1}, id_{M2}, \cdots, id_{Ml_M}$

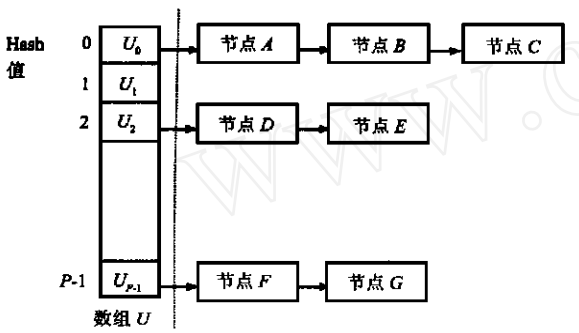

图 1　用 Hash 表存储倒排文件

3　快速检索方法

本文主要解决范围查询问题. 即:输入查询矢量 y,在 Θ 中寻找离 y 距离小于 T 的那些矢量,从而得到查询结果 $R_Q = \{x_i \mid \|x_i - y\|^2 \le T, x_i \in \Theta\}$. 具体算法如下:

3.1　格型矢量量化　用相同的量化器对查询矢量进行量化:$y_q = LVQ(y)$.

3.2　第一次检索:搜索格点 y_q 的相邻格点集合,获得候选结果.

设 D 维格为 Λ,首先定义邻近格点集合 $NE = \{q_k \mid \|q_k\|^2 \le T_n, q_k \in \Lambda\}$. 当 Λ 和 T_n 给定时,根据格的代数性质很容易得到邻近格点集合 NE. 然后,以格点 y_q 为中心定义 y_q 的相邻格点集:$\Phi(y_q) = \{y_q + q_k \mid q_k \in NE\}$. 接着,利用 Hash 技术快速判断邻近格点 ϕ_i ($\phi_i \in \Phi(y_q)$) 是否存在于倒排文件之中. 如果有,就从 Hash 表对应节点中读出对应的图像序号集,作为查询的初步候选结果 R_Q^0 的一部分. 假设 NE 中有 $|NE|$ 个邻近格点,则本步骤复杂度是 $|NE|$ 次 Hash 搜索,可以看出它和数据库大小无关,只和维数有关.

3.3　第二次检索:对候选结果按相似度排序.

在候选结果 R_Q^0 中,计算每个图像所对应的特征矢量 x_i ($i \in R_Q^0$) 和查询矢量 y 之间的距离,将距离与 T 比较,剔除大于 T 的那些候选者,就得到最终查询结果 R_Q. 本步骤复杂度和 R_Q^0 中候选图像数目有关. 显然,一旦空间划分的方案确定后,数据库规模越大,则落在每个 Voronoi 胞腔中的点数越多,R_Q^0 中候选图像数目就越多,则本步骤复杂度也就越大!

因此,整个检索算法的复杂度由两部分组成:第一次的 Hash 复杂度和第二次的顺序比较复杂度.

4　实验研究

4.1　常用的一些代数格

表 2 给出 7 种代数格的 Theta 级数 $\Pi(m)$[1],它实际上是格点矢量之能量为 m 的格点数目. 对 E_8 来说,当 $T_n = m = 2$ 时,$|NE| = \Pi(0) + \Pi(1) + \Pi(2) = 241$. 从表 2 还可以看出:随着 m 增加,$\Pi(m)$ 变大;随着维数增加,$\Pi(m)$ 也很快增加;这就意味着增加 Hash 搜索的次数,也就是增加整个查询的复杂度. 例如,对 Λ_{16} 和 $T_n = m = 4$,$|NE| = 4321$,检索复杂度尚可;对 Λ_{24} 和 $T_n = m = 4$,Λ_{24} 的 $|NE| = 196561$,检索复杂度显著增加! 因此,从表 2 可以断定,本文算法比较适合 16 维以下点数据的索引,维数太高时将出现维数灾难. 不过,无论如何这个结果比 R 树好,因为 R 树在 10 维左右时性能就很差了.

表 2　几种代数格的 Theta 级数 $\Pi(m)$

m	Z	A_2	D_4	E_8	K_{12}	Λ_{16}	Λ_{24}
0	1	1	1	1	1	1	1
1	2	6	0	0	0	0	0
2	2	0	24	240	0	0	0
4	2	6	24	2160	756	4320	196560
6	2	0	96	6720	4032	61440	16773120

4.2　用 E_8 对 8 维点数据进行索引的模拟实验

为了表明本文方法的有效性,我们进行了模拟实验. 在实验中,首先生成有 N 个随机矢量的数据库,具体生成方法如下:(1) 生成 L 个均匀分布在超立方体 $[-A, +A]^D$ 中的随机矢量;(2) 生成 N 个高斯分布的随机矢量(每个坐标的均值为 0、方差为 σ,且相互独立);(3) 依次取出一个高斯矢量,再从 L 个均匀分布的随机矢量中随机取一个矢量,两者相加,最终就得到有 L 个中心的随机矢量的数据库. 用类似方法再生成 L_q 用于测试的查询矢量.

另外,我们定义加速比 $Speed\_up = t_{slow}/t_{fast}$,其中 t_{slow} 是用顺序全搜索方法检索整个特征数据库所需要时间,t_{fast} 则是用本文索引方法后所需要的时间,显然 $Speed\text{-}up$ 则在一定程度上反映了索引所带来的检索性能的提高. 在本文实验中,我们没有考虑磁盘对索引方法的影响,也就是说测试时所有数据均在机器内存中,t_{slow} 和 t_{fast} 中不包括访问外存的时间. 我们选择 E_8 格进行实验,并固定以下参数:$A = 4$, $D = 8$, $L = 5000$, $P = 250007$, $T = 1$, $T_n = 2$, $L_q = 1000$,表 3 给出实验结果.

表 3　不同 N 和 σ 时的索引性能

N	$Speed\_up(\sigma = 0.3)$	$Speed\_up(\sigma = 0.5)$
100,000	140	172
500,000	214	545
1000,000	373	795

从表 3 实验数据可以看出以下几个规律:(1) 固定 σ 时,若数据库规模 N 变大,则加速比有所增加,但不是按比例增加,原因在于:查询算法的第一次检索复杂度几乎不变,而第二次检索复杂度有所增加,两者综合的结果正好反映了上述

现象;(2)固定 N 时,若 σ 增加,则加速比增加,其原因在于:σ 越大,意味着数据在空间分布就越稀疏,落在同一个 Voronoi 胞腔中的点数将减少,因此检索算法的第二次检索复杂度将下降,从而导致加速比增加;(3)虽然不同参数会导致了不同的索引性能,但是加速比总是很明显,这也说明本文方法的有效性.

5 结论

本文提出一种新的用于点数据的组织、索引和检索方法,它的主要特点是既利用了代数格的良好性质,又利用了 Hash 高速查找能力.为了进一步提高检索性能,支持更高的维数(克服维数灾难)和其它距离测度(使用加权欧氏距离),还有很多问题需要进一步深入研究.

参考文献:

[1] J. Conway and N. J. A. Sloane. Sphere Packings, Lattice and Groups [M]. New York: Springer-Verlag. 1991.

[2] G. D. Forney. Coset codes ——part Ⅱ: Binary lattices and related codes [J]. IEEE Trans. Information Theory, Sept. 1988. 34: 1152 - 1187.

[3] J. H. Conway and N. Sloane. A fast encoding method for lattice codes and quantizers [J]. IEEE Trans Information Theory, Nov. 1983, 29: 820 - 824.

[4] Guttman A. Rtrees: A dynamic index structure for spatial searching [A]. Proc. ACM SIGMOD Int. Conf. on Management of Data, Boston, MA, 1984: 47 - 57.

[5] Beckmann N., Kriegel H., Schneider R. Seeger B.: The R * tree: An efficient and robust access method for points and rectangles [A]. Proc. ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ, 1990: 322 - 331.

(上接第 93 页)

参考文献:

[1] Y. Linde, A. Buzo and R. M. Gray. An algorithm for vector quantizer design [J]. IEEE Transactions on Communications, 1980, 28(1): 84 - 95.

[2] A. Gersho and R. M. Gray. Vector quantization and signal compression [M]. Kluwer Academic Press, Massachusetts, 1990.

[3] C. D. Bei, and R. M. Gray. An improvement on minimum distortion encoding algorithm for vector quantization [J]. IEEE Transactions on Commun., 1985, 33(10): 1132 - 1133.

[4] T. Torres, and J. Huguet. An improvement on codebook search for vector quantization [J]. IEEE Transactions on Communications, 1994, 42(2): 208 - 210.

[5] M. R. Soleymani, and S. D. Morgera. An efficient nearest neighbor search method [J]. IEEE Transactions on communications, 1987, 35(4): 677 - 679.

[6] S. W. Ra and J. K. Kim. A fast mean-distance-ordered partial codebook search algorithm for image vector quantization [J]. IEEE Transactions on Circuit and Systems-Ⅱ, 1993, 40(9): 576 - 579.

[7] A. P. Wilton and G. F. Carpenter. Fast search methods for vector lookup in vector quantization [J]. Electronics Letters, 1992, 28(5): 2311 - 2312.

[8] R. L. Joshi and P. G. Poonacha. A new MMSE encoding algorithm for vector quantization, Proc [J]. IEEE ICASSP'91, 1991: 645 - 664.