PCPF:一种面向多媒体数据库中高维向量匹配的并行索引结构

陈慧中[1),2)] 陈永光[3)] 景 宁[1)] 陈 荦[1)]

[1)](国防科学技术大学电子科学与工程学院 长沙 410073)

[2)](西南电子电信技术研究所上海分所 上海 200434)

[3)](军械工程学院 石家庄 050003)

摘 要 提高特征向量的匹配效率是将高维局部特征运用于多媒体数据检索的关键.面向多核处理器架构,提出一种新的 PCPF 索引以及 PCPF 并行构建与并行查询匹配算法.PCPF 并行构建算法通过量化特征向量构建近似向量空间上的高维索引结构,并进行空间划分并行构建多个子索引分支;PCPF 并行查询匹配算法利用优先队列在邻近子分支上并行过滤得到近似近邻候选集,精确计算候选实际特征向量得到最终近邻.实验及分析表明,与经典的 BBF 算法相比较,PCPF 通过降低了磁盘 I/O 和浮点运算次数以及并行优化,显著提升了查询匹配效率,总体匹配精度也有所提高.

关键词 特征向量匹配;多媒体检索;BBF;高维 kNN 查询

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2011.02009

PCPF: A Parallel Index for Matching the High-Dimensional Vectors in Multimedia Databases

CHEN Hui-Zhong[1),2)] CHEN Yong-Guang[3)] JING Ning[1)] CHEN Luo[1)]

[1)](*College of Electronic Science and Engineering, National University of Defense Technology, Changsha* 410073)

[2)](*Shanghai Branch, Southwest Electronic and Telecommunication Research Institute, Shanghai* 200434)

[3)](*Ordnance Engineering College, Shijiazhuang* 050003)

Abstract The key point in applying high-dimensional local features to retrieval in multimedia databases is to improve the efficiency of feature matching. Facing the multi-processor architecture, we have investigated a novel Parallel Compressed Priority Filter (PCPF) index, together with the corresponding parallel construct and query algorithms. The PCPF quantizes the feature vectors to compress the search space, constructs a high-dimensional index with several branches, searches candidates via priority queue in different branches, and calculates the exact feature vectors to get the nearest neighbors in parallel. It has been proved by experiments and via analysis that PCPF can reduce disk I/O and float-pointing calculation. It is also optimized by parallel. It is much faster and more precise than the classical BBF algorithm with no increase of constructive time.

Keywords feature matching; multimedia retrieval; BBF; high-dimensional kNN search

---

收稿日期:2011-07-18;最终修改稿收到日期:2011-08-19.本课题得到国家"八六三"高技术研究发展计划项目基金(2008AA12A211,2011AA12A306)、国家自然科学基金(60902036)资助.**陈慧中**,女,1982 年生,博士研究生,主要研究方向为基于内容图像检索、智能信息处理.E-mail: chen_huizhong@yahoo.cn. **陈永光**,男,1962 年生,博士,教授,博士生导师,主要研究领域为电磁环境模拟与评估技术、电子战作战模拟. **景 宁**,男,1963 年生,博士,教授,博士生导师,中国计算机学会(CCF)会员,主要研究领域为地理信息系统、空间数据库. **陈 荦**,男,1973 年生,博士,教授,中国计算机学会(CCF)会员,主要研究领域为地理信息系统与数据库技术.

1 引 言

多媒体数据库由于存储对象(如图片、音频、视频等)内容丰富,信息量大,通过传统的元数据检索方式获取对象不能完全满足需求.基于内容的检索技术能根据视觉等内容特性将与用户感兴趣的多媒体对象从大容量的数据库中检索出来,以实现更为高效而广泛的应用.

近来,随着基于内容的多媒体检索技术的不断发展,局部特征由于其稳定性高、独特性好、信息量丰富等优点已经成为多媒体对象特征描述的研究热点[1].然而,在实际检索过程中,由于局部特征数量大,描述向量维数高,使得在大规模多媒体数据库中进行向量匹配计算量大,效率低下.因此,如何高效地匹配检索样例与多媒体数据库对象间的大量高维特征向量成为提高检索效率的关键所在.

建立高维索引是解决问题的有效途径[2],相关的研究结果包括:va-file[3]向量近似的思想,是使用二进制串近似实际向量,通过过滤与精确查找两个步骤使高维情况下的查询效率仍然优于顺序查找.va$^+$-file[4]是通过将数据集转换到 KLT 域并实施均匀的量化划分,提高了 va-file 的过滤效率.iDistance[5]是将高维数据空间进行划分,并为每个划分建立一个参考点,使用 B$^+$-树结构组织参考点,在此基础上实现 kNN 近邻查询.LSH[6]通过一组 Hash 函数将邻近的高维点集映射到 Hash 表的同一个桶中,以此得到对近邻的快速查询.LSB-tree[7]使用 B-tree 取代 Hash 结构,从而以较小的代价得到了相似的查询效率优化.基于聚类分解的 B$^+$-tree[8]采用类 B$^+$-tree 的索引结构,通过聚类分解,对数据进行更细致的划分来减少查询的数据访问.MS-tree[9]通过一种最大间隙空间映射策略,设计了索引的构建与范围查询处理算法.MRSVQH[10]根据随机选择的若干子向量的 L2 范数对特征向量进行量化和散列,搜索时仅考虑与查询向量有相同 Hash 值的特征向量集合,缩减了搜索范围.

上述研究大多是针对通用 k-NN 查询做出优化,其中一些对于高维向量的查询匹配效率并不理想,还有一些存在索引结构理想化复杂化等问题,因而在多媒体数据库高维特征向量的实际匹配中应用较少.

Lowe[2]为其著名的 SIFT 局部特征推荐了一种 Best-Bin-First(BBF)近似查询匹配算法[11],BBF 算法基于 k-D 树索引,查找最近邻时,首先遍历到查询点所在的桶,遍历过程中将每一个节点通过计算比较到查询点的距离压入一个优先队列堆栈中.然后从该桶包含的点开始向上回溯,计算比较当前近邻,当搜索到叶节点时,下一搜索节点通过优先队列顶部出栈获得,以继续搜索,直至达到最大搜索次数时终止.BBF 算法是针对高维特征向量的匹配问题设计,采用优先队列和限制查询回溯次数等方法提高了查询效率,成为目前使用最为广泛的高维特征向量匹配方法.

此外,当前多核处理器得到了广泛的应用,在单机多核的架构下,由于单个线程在某一时刻仅能被一个处理器核执行,传统的单线程串行算法不能充分有效地利用多核处理器的计算资源.由此,本文从多媒体数据检索实际应用需求出发,以提高高维向量查询匹配效率为目的,针对 BBF 算法遍历及回溯阶段磁盘 I/O 较大的缺陷,基于单机多核架构下的多线程并行模式,提出一种新的并行压缩优先过滤索引(Parallel Compressed Priority Filter,PCPF).本文的主要贡献包括:

(1)在 BBF 算法的基础上,引入向量近似的思想,考虑多核处理器条件下的并行优化,提出一种新颖的高维向量匹配并行索引结构 PCPF,给出了索引构建与查询匹配算法.

(2)对提出的 PCPF 并行构建与查询匹配算法进行了详细的算法时间复杂度分析.

(3)在 Benchmark 图片库和实际多媒体数据库基础上,对 PCPF 索引进行了大量的测试实验,结果表明:PCPF 算法在多媒体数据库高维向量查询匹配效率上较 BBF 算法有大幅提高,同时取得了较好的匹配效果.

2 问题描述

基于内容检索多媒体数据库时,首先通过对检索样例提取多个局部特征,产生高维特征描述向量集合.然后将样例的特征向量与存储于数据库中单个多媒体对象的特征向量集合逐一匹配,计算得到匹配度,匹配度越高则相关度越高.这里,特征向量的匹配与匹配度的定义如下.

定义 1(特征向量的匹配). 令 $\boldsymbol{q}$ 为检索样例 S 的一个特征向量,$\boldsymbol{V}$ 为数据库中一个多媒体对象 G 的特征向量集合.$\boldsymbol{o}_1$,$\boldsymbol{o}_2$ 分别为 $\boldsymbol{q}$ 在 $\boldsymbol{V}$ 中的最近邻与次近邻.δ 为匹配门限,$0<\delta\leq 1$.若

$$\|\boldsymbol{o}_1-\boldsymbol{q}\|/\|\boldsymbol{o}_2-\boldsymbol{q}\|<\delta \tag{1}$$

则称 $\boldsymbol{q}$ 在 $\boldsymbol{V}$ 中与 $\boldsymbol{o}_1$ 成功匹配;反之,$\boldsymbol{q}$ 无匹配结果.其中,匹配门限 δ 对于不同的实际数据集可通过实验取得,一般情况下,在文献[2]中推荐取 0.6,在文献[12]中推荐取 0.7.

定义 2(特征向量的匹配度). 令 $\boldsymbol{Q}$ 为检索样

例 S 的特征向量集合，$\boldsymbol{V}$ 为数据库中一个多媒体对象 G 的特征向量集合. 定义匹配度：

$$match(S,G)=\frac{\|\{\boldsymbol{q}\}\|}{\|\boldsymbol{Q}\|},\ \boldsymbol{q}\in\boldsymbol{Q},\text{且}\ \boldsymbol{q}\ \text{在}\ \boldsymbol{V}\ \text{中成功匹配} \tag{2}$$

根据上述定义，检索样例与数据库中一个多媒体对象的匹配度由两者特征向量的匹配个数计算得到，特征点匹配的过程实际上就是一个特征向量空间中最近邻与次近邻的查询过程.

因此，高维向量空间中的 2-NN 查询是实现快速匹配的实际关键问题，定义如下.

定义 3(2-NN 查询). 令 $\boldsymbol{q}=(q_1,q_2,\cdots,q_D)$ 为一个待匹配高维向量，$\boldsymbol{V}=\{\boldsymbol{v}\}=\{(v_1,v_2,\cdots,v_D)\}$ 为检索库中一个多媒体对象的特征向量集合.

求取 $\boldsymbol{o}_1=\{o_{11},o_{12},\cdots,o_{1D}\}$，$\boldsymbol{o}_2=\{o_{21},o_{22},\cdots,o_{2D}\}$，使得

(1) $\forall\boldsymbol{v}=(v_1,v_2,\cdots,v_D)\in\boldsymbol{V}$ 且 $\boldsymbol{v}\neq\boldsymbol{o}_1$

$$\Big(\sum_{d=1}^{D}(o_{1d}-q_d)^2\Big)^{\frac{1}{2}}\le\Big(\sum_{d=1}^{D}(v_d-q_d)^2\Big)^{\frac{1}{2}} \tag{3}$$

(2) $\forall\boldsymbol{v}=(v_1,v_2,\cdots,v_D)\in\boldsymbol{V}$ 且 $\boldsymbol{v}\neq\boldsymbol{o}_1,\boldsymbol{o}_2$

$$\Big(\sum_{d=1}^{D}(o_{2d}-q_d)^2\Big)^{\frac{1}{2}}\le\Big(\sum_{d=1}^{D}(v_d-q_d)^2\Big)^{\frac{1}{2}} \tag{4}$$

这里，使用常用的欧式距离(Euclidean Metric)作为特征向量间的距离度量，实际计算时，为了节省计算开销，有时不进行开方操作.

3　并行压缩优先过滤索引 PCPF

PCPF 是一种面向多媒体数据库中高维向量的快速匹配设计的静态索引，其主体流程包括索引的构建以及查询匹配两大部分.

构建 PCPF 索引时，首先对原始数据集进行 KL 变换、量化压缩等数据压缩预处理步骤，然后根据 PCPF 索引结构对预处理后的近似向量集合构建高维向量近邻 PCPF 索引，同时保留 KL 变换后的数据集作为实际向量集合(步骤如图 1 中深色箭头所示).

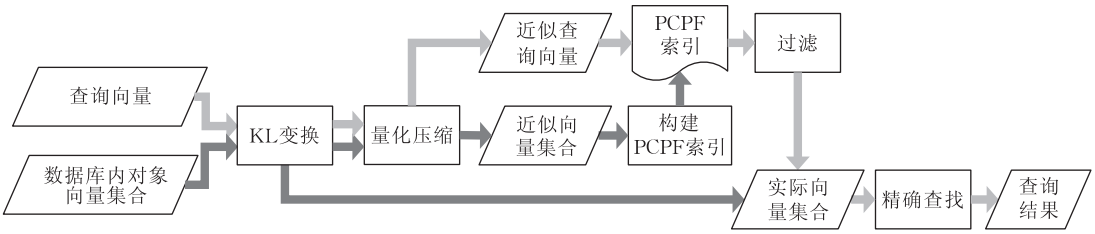


图 1　PCPF 总体框图示意

对特征向量 $\boldsymbol{q}$ 进行查询匹配时，首先对 $\boldsymbol{q}$ 进行 KL 变换并量化压缩；量化后的近似查询向量运用 PCPF 查询匹配算法通过索引文件过滤得到候选的近似特征向量集合；然后在此候选集中进行精确计算与排序，得到最近邻与次近邻，以此计算是否匹配成功(步骤如图 1 中浅色箭头所示).

3.1　数据压缩预处理

在构建索引及查询之前，为了消除数据各维间的相关性、得到更为平均的量化划分，从而提高查询效率. 依据文献[4]中的方法对数据集进行压缩预处理：KL 变换，位分配及量化.

KL 变换将数据集转换到 KLT 域，以此消除各维之间的互相关，使各维的数据间相互正交.

位分配算法根据数据的分布情况为各维分配量化位数，令第 i 维上的数据方差为 σ_i，总共的可分配位数为 b，分配给第 i 维的位(bit)数为 b_i，则有 $b=\sum b_i$. 位分配主要步骤如下：

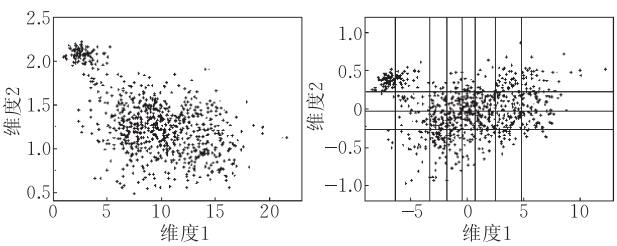
1. 令 $k=0$. 对于第 i 维，令 $s_i=\sigma_i$，$b_i=0$.

2. 求 $j=\max^{-1}s_i$，则

$$b_j\leftarrow b_j+1,\quad s_j\leftarrow s_j/4.$$

3. $k\leftarrow k+1$. 若 $k>b$，则终止；反之，重复步 2.

图 2 给出了一个二维实际数据集的 KL 变换和量化位数分配示例.



(a) 原始数据集　　(b) 处理后的数据集

图 2　KL 变换及位分配示例

量化根据位分配算法得到的可分配位数划分各维，在向量空间形成多个胞腔(*cell*)，以二进制串唯一地表示单个胞腔，并作为胞腔内原始特征向量的近似表示，图 3 给出了一组量化后的二维向量示例. 量化可以通过均匀地将原始向量分配给不同的 ps 个线程并行完成.

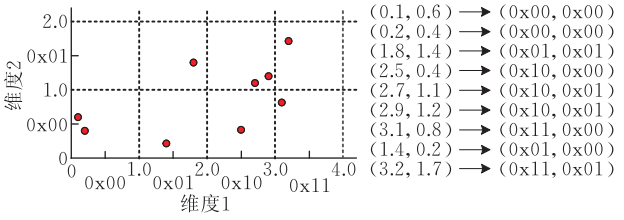


图 3 数据量化示例

3.2 PCPF 索引结构

PCPF 索引结构使用一种类 k-D 树森林的结构来组织量化压缩后的近似特征向量. PCPF 结构不具有一般树形结构唯一的根节点,它的第一层是一个链表结构,由 KL 变换后的第一维按数据均匀划分而成. 每一个链表的元素可以看作一棵子树的"根节点",称为"首层节点",每个首层节点下采用k-D 树结构形成一棵子树分支. 所有节点均包含一个指针序列,序列内元素指向相同近似向量对应的一个或多个原始特征向量在磁盘文件中的位置.

PCPF 的节点可以分为三类:首层节点*top_node*、中间节点 *mid_node* 和叶结点 *leaf_node*,节点的定义和主要包含元素如图 4 所示.

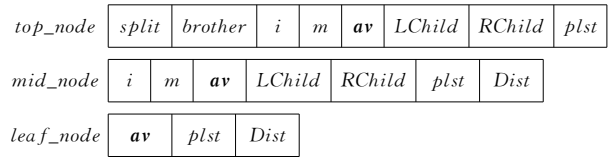


图 4 PCPF 索引节点

其中,*split* 为首层节点的分割点,*brother* 为指向下一个首层节点的指针,i 为子树当前分割的维数,m 为子树当前分割点,***av*** 为当前节点对应的近似向量,*LChild* 为左子树指针,*RChild* 为右子树指针,*plst* 为指向实际向量的指针序列,*Dist* 记录了查询遍历时节点至查询点的距离.

图 5 给出了二维的图 3 数据 PCPF 索引结构示意.

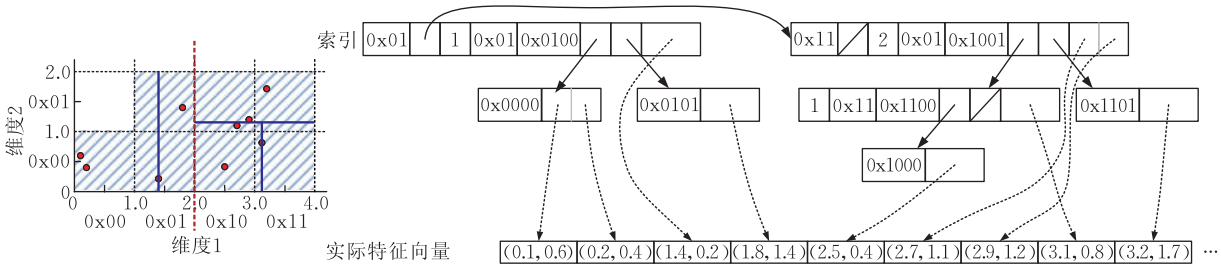


图 5 索引结构示意

3.3 PCPF 构建算法

基于 PCPF 的索引结构,提出一种可并行的 PCPF 构建算法,共分为 3 个步骤:压缩预处理、数据子集划分和并行索引构建. 先对预处理后的数据集进行一定的划分,在此基础上再并行地对划分后的数据集进行索引的构建.

数据子集划分时,对于预处理后 k 维近似向量空间中的数据集 $\boldsymbol{A}$, $\|\boldsymbol{A}\|=N$,将第一维均匀划分为 s 个区间,形成大小分别为 $N_1, N_2, \cdots, N_s$ 的 s 个子集:$\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_s$. 这里,由于经过 KL 变换,预处理后的近似向量空间在第一维上包含了最大的能量,方差为最大(见图 2(b)示意),同时近似向量点具有相对均匀的分布,因此本文选择第一维作为划分的依据. 划分的区间个数 s 代表了可并行的程度,作为并行执行时划分线程的数目,一般与处理器核数相匹配.

并行索引构建时,对于每个子集,并行构建一个类 k-D 树结构:在方差最大的第 i 维上进行分割,分割点为该维数据的中间数 m,则两侧各有相同数量的点,生成一个节点以存储 i 与 m. 小于 m 的点划分为左子树,否则划分为右子树,在两棵子树上递归进行上述划分步骤,直至子树仅包含一个点. 子集上的第一个分割点作为该子树的根节点,即首层节点. 无子树的点作为叶结点,其它作为中间节点. 在每个子集 $\boldsymbol{A}_j$ 上构建完成一棵深度为 $\lceil \log_2 N_j \rceil$ 的二叉平衡树后,将首层节点以链表形式按子集划分区间的顺序连接起来,从而完成 PCPF 索引构建.

PCPF 构建算法的详细步骤描述如下,其中,步 1 预处理原始数据集;步 2~4 划分数据子集;步 5~9 构建索引.

算法 1. PCPF_Construct.

Input: $\boldsymbol{V}$: Set of Real Vectors, s: Number of Subsets;

Output: *rootLst*: List of Top-Level Nodes;

1. $\boldsymbol{A} \leftarrow Preprocess(\boldsymbol{V})$;
2. let dimension range $DR \leftarrow FirstDR(\boldsymbol{A})$;
3. $split_j \leftarrow (DR \times j)/s$, $j=1,2,\cdots,s$, $split_j \in \{split\}$;
4. $\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_s \leftarrow Divide(\boldsymbol{A}, \{split\})$
5. for each $\boldsymbol{A}_j$ do
6. $root_j \leftarrow BuildTree(\boldsymbol{A}_j)$;
7. $root_j.split \leftarrow split_j$;
8. $rootLst.Insert(root_j)$;

9. end for;
10. return *rootLst*;

Function *BuildTree* (set of vectors $\boldsymbol{A}$)

1. $i \leftarrow \mathrm{Max}^{-1}(Variance(d))$, $d=1,2,\cdots,D$;
2. $Node.\boldsymbol{av} \leftarrow Median\_i(\boldsymbol{A})$;
3. $Node.m \leftarrow Node.\boldsymbol{av}_i$;
4. $Node.i \leftarrow i$;
5. $\boldsymbol{AL} \leftarrow \varnothing, \boldsymbol{AR} \leftarrow \varnothing$;
6. for each $\boldsymbol{a}=(a_1,a_2,\cdots,a_D) \in \boldsymbol{A}$ do
7. if $a_i < Node.m$ then $\boldsymbol{AL} \leftarrow \boldsymbol{AL} \cup \boldsymbol{a}$;
8. else $\boldsymbol{AR} \leftarrow \boldsymbol{AR} \cup \boldsymbol{a}$; end if;
9. end for;
10. $Node.LChild \leftarrow BuildTree(\boldsymbol{AL})$;
11. $Node.RChild \leftarrow BuildTree(\boldsymbol{AR})$;
12. $Node.plst \leftarrow RealAddrs(Node.\boldsymbol{av})$;
13. return *Node*;

3.4 PCPF 查询匹配算法

进一步提出并行压缩优先过滤(PCPF)查询匹配算法. 根据索引结构，查询匹配通过查询向量集在各子树上并行查询完成. 对于一个查询向量，其匹配过程分为 3 个阶段：预处理、量化过滤与精确查找阶段. 预处理根据构建索引时的参数进行 KL 变换与量化近似操作，得到量化的近似查询向量，并根据划分的 $\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_s$ 子集将查询向量分配到距第一维分量最近的分割点相邻的二棵查询子树 $\boldsymbol{A}_j, \boldsymbol{A}_{j+1}$ 上；在量化过滤阶段，采用基于 BBF 算法思想的优先队列递归分别在 $\boldsymbol{A}_j, \boldsymbol{A}_{j+1}$ 上并行查找获得量化查询向量的近似近邻候选集，候选集大小设定为 C，$C \geq 2$，最大搜索次数分别为 T_j 和 T_{j+1}，满足 $T_j \times N_{j+1} = T_{j+1} \times N_j$ 且 $T_j + T_{j+1} = T$. 在精确查找阶段，读取候选集对应的实际向量并分别计算到查询向量的精确距离，排序得到距离最小的最近邻与次近邻，根据式(1)计算是否匹配. 对所有查询点匹配完毕之后，根据式(2)计算匹配度. PCPF 查询匹配的并行分为两个层面：第 1 层面是单个向量查询匹配时，在两棵相邻子树上并行查找近似候选集；第 2 层面是对于查询样例的多个特征向量，同时在 s 棵子树上执行第一层面的操作.

PCPF 查询算法使用曼哈顿距离定义近似向量空间上的距离关系：令 $\boldsymbol{aq}=(aq_1, aq_2, \cdots, aq_D)$ 为查询向量 $\boldsymbol{q}$ 的近似向量，$\boldsymbol{a}=(a_1, a_2, \cdots, a_D)$ 为一个索引数据集的近似向量，则距离：

$$DIST(\boldsymbol{a}, \boldsymbol{aq}) = \sum_{i=1}^{D} |a_i - aq_i| \qquad (5)$$

PCPF 查询匹配算法的详细步骤描述如下，其中，步 1～9 预处理查询向量集；步 11～25 量化过滤候选近邻；步 26～32 精确查找匹配.

算法 2. PCPF_Match.

Input: $\boldsymbol{Q}$: Set of Query Vectors, *rootLst*: Top-Level Nodes List of Search Index, C: Size of Candidates, T: Max Search Times, δ: Threshold of Match

Output: *match*: Matched Ratio;

1. $\boldsymbol{AQ} \leftarrow Preprocess(\boldsymbol{Q})$;
2. let matched number $mn \leftarrow 0$;
3. let search tree $sLst \leftarrow \varnothing$;
4. for each $\boldsymbol{aq}=(aq_1, aq_2, \cdots, aq_D) \in \boldsymbol{AQ}$ do
5. for each $root_j \in rootLst$ do
6. if $IsMin(|aq_1 - root_j.split|)$ then
7. $sLst.Insert(root_j, root_{j+1})$;
8. end if;
9. end for;
10. let set of candidates $\boldsymbol{CAND} \leftarrow \varnothing$;
11. for each $root_j \in sLst$ do
12. let current node $Node \leftarrow root_j$;
13. let priority queue $\boldsymbol{PQ} \leftarrow \varnothing$;
14. while *Node* is not *leaf_node* do
15. $\boldsymbol{PQ}.Push(Node)$;
16. if $\boldsymbol{aq}[Node.i] < Node.\boldsymbol{av}[Node.i]$ then
17. $Node \leftarrow Node.Lchild$;
18. else
19. $Node \leftarrow Node.Rchild$;
20. end if;
21. end while;
22. let *Cth* nearest distance $Dist_c \leftarrow \infty$;
23. let current search time $t \leftarrow 0$;
24. $CPF\_Search(Node)$;
25. end for;
26. for each $pt \in \boldsymbol{CAND}[c].plst$, $c=1 \cdots C$ do
27. $min_1 \leftarrow \min \| (^*pt) - \boldsymbol{q} \|$;
28. $min_2 \leftarrow \text{second\_min} \| (^*pt) - \boldsymbol{q} \|$;
29. end for;
30. if $min_1 / min_2 < \delta$ then
31. $mn \leftarrow mn + 1$;
32. end if;
33. end for;
34. $match \leftarrow mn / \| \boldsymbol{Q} \|$;
35. return *match*;

Function *CPF_Search*(current node *Node*)

1. while $t \leq T_j$ do
2. if DIST($Node.\boldsymbol{av}, \boldsymbol{aq}$) $\leq Dist_c$ then
3. $\boldsymbol{CAND}.Insert(Node)$;
4. if $\boldsymbol{CAND}.size \geq C$ then
5. $Dist_c \leftarrow$ DIST($\boldsymbol{CAND}[C-1].\boldsymbol{av}, \boldsymbol{aq}$);
6. $\boldsymbol{CAND}.$ Delete($\boldsymbol{CAND}[C]$);
7. end if;
8. end if;
9. $t \leftarrow t+1$;
10. if *Node* is *leaf_node* then
11. $Node \leftarrow \boldsymbol{PQ}.Pop()$;

```
12.       CPF_Query(Node);
13.     end if;
14. end while;
```

4 算法复杂度分析

本节对提出的 PCPF 构建算法与查询匹配算法的时间复杂度进行理论分析. 这里,定义一个 D 维向量在一个维度上的基本运算为一次基本运算,则一个 D 维向量上的相应运算的运算量为 D.

4.1 构建算法时间复杂度

在构建 PCPF 索引前,首先需进行预处理操作,其中 KL 变换复杂度为 $O(DN^3)$;量化位数分配的复杂度为 $O(D^2N)$;量化复杂度为 $O(DN)$,量化操作可以完全均匀地划分给 ps 个线程执行,因此其理论复杂度可由其中一个线程上的复杂度代替,为 $O(DN/ps)$. 故总的量化预处理复杂度为 $O(DN^3)+O(D^2N)+O(DN/ps)$.

构建 PCPF 索引时,划分数据子集的复杂度为 $O(N)$. 然后在每个数据子集上建立索引分支,每个字分支根据 k-D 树的构建复杂度为 $O(\log^2(DN_j))$. 并行子索引构建复杂度其实为最大子集合上的构建复杂度,即 $O(\log^2(D\mathrm{Max}(N_j)))$,当子集完全划分时,取得最佳值,为 $O(\log^2(D(N/s)))$,最差情况下,数据划分极度不均匀,可视作所有向量集中于一个子集的极端情况,等同于 $s=1$ 时,为 $O(\log^2(DN))$. 则构建 PCPF 索引复杂度为 $O(N)+O(\log^2(D\mathrm{Max}(N_j)))$,最优划分情况下为 $O(N)+O(\log^2(D(N/s)))$,最差划分情况下为 $O(N)+O(\log^2(DN))$.

总体的构建时间为数据预处理和索引构建两部分之和,依据上面的分析结果消减低次项及系数,为 $O(DN^3)+O(D^2N)$.

4.2 查询匹配算法时间复杂度

查询匹配单个向量时,首先进行预处理,此时预处理参数已确定,不需重新计算,因此单个 D 维向量的 KL 变换计算复杂度为 $O(D)$,量化复杂度为 $O(D)$,因此,总体预处理复杂度为 $O(D)$.

查询匹配分别在相邻的两个子树分支上进行,对于每棵子树而言,遍历阶段复杂度为 $O(\log(DN_j))$,回溯阶段复杂度为 $O(DT_j)$. 并行查询匹配复杂度为最深的一棵子树上的查询匹配复杂度,当子集完全均匀划分时,子树分支深度一致,取得最佳值,为 $O(\log(D(N/s)))+O(D(T/2))$,最差情况下,数据划分极度不均匀,可视作仅构建了一棵索引子树,即 $s=1$,此时复杂度为 $O(\log(DN))+O(DT)$.

总体的单个向量查询匹配时间为预处理和索引构建两部分之和,依据上面的分析结果消减低次项及系数,为 $O(\log(DN))+O(DT)$.

对 M 个向量进行查询匹配时,可分配到不同子树并行执行,最佳情况下,子集完全均匀划分且 M 个查询向量也完全均匀分布,此时复杂度为 $O(\log(D(M/s)(N/s)))+O(D(MT/s))$,最差情况下,等同于 $s=1$,为 $O(\log(DMN))+O(DMT)$. 消除低次项和系数后,M 个向量并行查询匹配的时间复杂度为 $O(\log(DMN))+O(DMT))$.

5 实验及结果分析

为了验证 PCPF 索引性能,本文基于多媒体数据库进行了并行效率实验、索引构建效率实验、查询匹配效率实验和查询匹配效果实验. 实验中,主要比较了 PCPF 与经典的 BBF 算法,同时为了更好地分析实验结果,本文也对单线程串行执行的 PCPF 算法进行了实验,作为参考对比.

本文采用 2 个图像数据库作为实验数据来源,一个包含从 benchmark 图片库 COREL 中精选的 10 个主题共 1000 张图像;另一个为实际工作中使用的月球空间数据影像库,共包含 3227 张月球表面图像. 图像高维特征向量采用 SURF[12] 特征,图像平均特征点数目为 10933, 90% 的图像特征点数目分布在 5000～15000 之间.

实验硬件环境:Intel Core i3 2.93 GHz 4 核 CPU,2.0 GB RAM,采用 Microsoft Windows XP Professional 操作系统. 实验软件开发环境:VC 2008+OpenCV 2.0. 实验相关参数如下:特征向量默认维数为 128,总量化位数为 2^{10},PCPF 与 BBF 算法搜索次数 T 按文献[2]推荐设置为 200,默认候选集大小 C 为 2,默认匹配门限 δ 为 0.7,预处理线程数目 $ps=s$.

5.1 并行效率实验

PCPF 索引面向单机多核架构采用多线程框架完成并行索引构建与查询匹配. 并行效率实验旨在测试线程数目对于效率的影响,这里,使用并行加速比和并行效率衡量,定义如下:

$$\text{并行加速比}=\frac{\text{并行执行时间}}{\text{串行执行时间}} \tag{6}$$

$$\text{并行效率}=\frac{\text{并行加速比}}{\text{实际利用核数}} \tag{7}$$

图 6 给出了实验结果,(a)为构建 PCPF 索引时不同线程并行执行下的加速比与效率,是选自 COREL 和月球库的 10 组平均大小为 10000 的向量集合的构建平均值,横坐标指示了线程数目 $s=2,3,\cdots,10$. (b)为查询匹配的结果,查询向量集合大小为 500～1000.

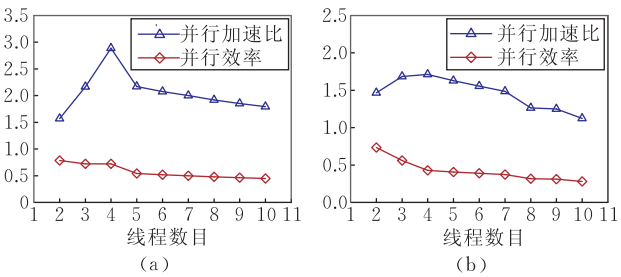


图 6　并行加速比与并行效率实验结果

从图 6 可以看出，在实验采用的四核处理器环境下，在不同线程下，构建加速比为 1.57～2.89，并行效率为 0.45～0.79；查询匹配的加速比为 1.12～1.71，并行效率为 0.28～0.73. 并行加速比在 $s=4$，即与处理器核数相匹配时取得最大值，线程数小于处理器核数时递增，超过时递减. 并行效率随着线程数目增加而递减.

根据实验结果可以得出，并行线程数目 s 等于处理器核数 $core$ 时，可以得到最优的性能提升；$s<core$ 时，由于没有充分利用每个处理器的计算资源，加速比较小，但此时一方面由于实际利用核数（即式(7)中的分母）也较少，另一方面存在空闲处理器可以处理操作系统等的计算请求，故并行效率较高；$s>core$ 时随着线程数目的增加，维护多个线程本身的开销增大，并行加速比与效率均有所下降. 此外，构建算法的性能提升优于查询匹配算法，这是由于对向量集合构建索引前经过了一系列预处理，使得数据集分布较为均匀，因此各线程负载较为均衡；而查询向量合集本身并非均匀分布，而其预处理采用的 KL 变换基以及量化维数等参数均是根据数据库中多媒体对象的向量集合的数据分布计算到，并不一定能很好地划分查询数据集而使各线程负载十分均衡，因此，数据分布的差异使并行性能提升也有所不同.

由此，PCPF 通过多处理器架构下的多线程并行执行得到了明显的性能优化，并行线程数目等于处理器核数时为最优，因此，在后续实验中，取$s=4$.

5.2　索引构建效率实验

索引构建效率实验旨在评估 PCPF 对特征向量数据集构建索引的时间消耗. 事实上，索引的构建在查询匹配之前完成并与多媒体对象一起存储于数据库中，其时间消耗并不影响检索效率. 但由于 PCPF 为静态索引，即当一个多媒体对象特征向量集合改变时，将重新建立索引，故我们也对此进行评估.

图 7 给出了实验结果，图(a)比较了从 5k～15k 的不同规模的向量集合上构建 PCPF 索引和 BBF 索引时的时间消耗，每个构建时间均为 10 个向量集合的构建平均值，来源为月球图像库；图(b)比较了不同维数($D=20,50,80,120$)的向量集合上构建 PCPF 索引和 BBF 索引时的时间消耗，同样为 10 个向量集合的构建平均，来源为 COREL 图片库.

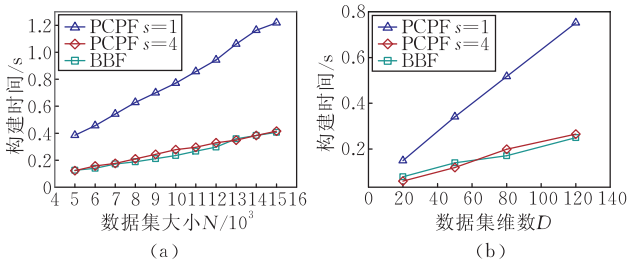


图 7　PCPF 与 BBF 构建时间对比

从图中可以看出，采用串行($s=1$)的方法构建 PCPF 索引耗时大于 BBF 索引构建，其时间增长随数据集增大和维数增加而增加. 并行 PCPF 构建与 BBF 构建耗时十分接近.

串行 PCPF 耗时较多的主要原因是为了提升查询性能进行了 KL 变换、位分配及量化等压缩预处理步骤，这些操作的计算复杂度较高（见 4.1 节分析）从而引起额外耗时，并且计算量随数据集增大和维数增加均有增加，这与 BBF 索引直接构建于原始数据集上有所不同. 而并行 PCPF 由于采用多线程方式执行，大大降低了构建时间，虽然有些情况下略高于 BBF，但总体来说，差别不大.

由此，PCPF 索引通过并行构建降低了索引结构优化引起的构建时间增加，其构建时间在不同数据集大小和维数下与 BBF 相接近.

5.3　查询匹配效率实验

查询匹配效率实验比较了 PCPF 算法对匹配查询向量集合的磁盘 I/O 与时间消耗.

图 8 和图 9 分别给出了查询匹配的磁盘 I/O 总字节数和平均单个向量查询匹配耗时比较试验结果，取 10 个大小为 500～1000 的查询向量集合匹配时的平均值. 图(a)比较了在 5k～15k 的不同规模的向量集合上匹配查询向量集合的 I/O 与时间，实验影像库为月球图像库，图(b)为不同维数向量集下的结果，采用 COREL 图片库.

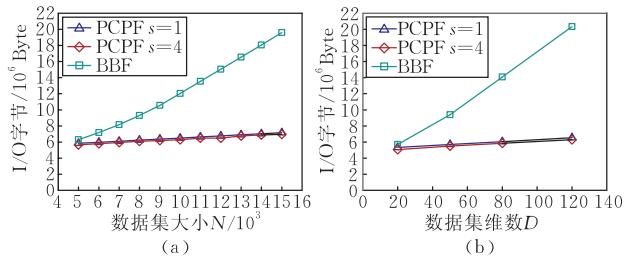


图 8　PCPF 与 BBF 查询匹配磁盘 I/O 对比

从图 8 中可以看出，随着索引向量集规模 N 的增加和维数 D 的增大，BBF 算法的 I/O 急剧增加，而

PCPF 算法增量平缓. $N=10000$ 时,串行执行的 PCPF 算法比 BBF 算法 I/O 减少 50%以上;$N=15000$ 时,可减少 60%以上;$D=50$ 时串行 PCPF 磁盘访问量为 BBF 的 60%不到,$D=120$ 时仅为 30%不到. 这是由于 N 和 D 增加时,通过 BBF 算法匹配的实际向量访问增长较快,而 PCPF 算法仅需多读取少量的近似向量,需访问的实际向量大小仍由候选集大小决定,数目很少,因而增量不大. 并行 PCPF 的匹配总 I/O 字节数与串行执行的相差不多,总体略偏少一些,其原因在于,一个向量在查询匹配时并行算法只遍历两棵邻近搜索子树,而非全部近似向量.

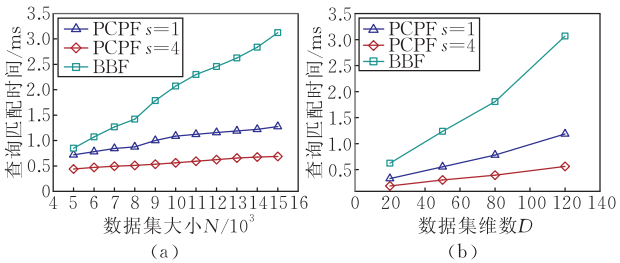


图 9 PCPF 与 BBF 查询匹配时间对比

图 9 显示,$N=10000$ 时,串行的 PCPF 查询匹配耗时仅不到 BBF 的 50%;$N=15000$ 时,为 40%左右;对于不同维数,$D=50$ 时耗时为 40%左右,$D=120$ 时为 35%左右. 这是由于 PCPF 查询匹配时通过访问近似向量减少了磁盘 I/O,且使用二进制运算求取近邻矢量减少了浮点运算次数,因此即使串行执行也明显优于 BBF 算法. PCPF 经过多线程并行执行,进一步提高了查询匹配效率,平均单个向量的匹配耗时与 BBF 相比仅为其 15%~50%.

由此,PCPF 查询匹配时磁盘 I/O 少于 BBF 算法,查询匹配耗时大幅减少,效率明显优于 BBF 算法.

5.4 查询匹配效果实验

查询匹配效果实验用于检验 PCPF 算法效率的提高对匹配效果的影响.

实验使用 10 幅包含光度变化、噪声污染、角度变换和尺度变换的样例图片与实验库中图像相匹配取平均值,采用查全率($Recall$)-查错率($1-Precision$)曲线,衡量不同算法的匹配效果,理想状态的查全率-查错率曲线应收缩于(0,1)点,即查错率为 0,查全率为 1. 其中,查全率($Recall$)和查错率($1-Precision$)定义如下,在实验中通过调整 δ 的设置计算取得.

$$Recall=\frac{\text{正确匹配特征点数}}{\text{目标总特征点数}} \tag{8}$$

$$1-Precision=\frac{\text{错配特征点数}}{\text{匹配特征点数}} \tag{9}$$

从图 10 中可以看到,通过串行执行的 PCPF、PCPF 和 BBF 算法得到的匹配效果十分接近,其中最好的是串行执行的 PCPF 查询匹配算法,其较好的效果一方面是由于其与 BBF 均为近似查找算法;另一方面有可能是由于通过近似向量来过滤匹配的方法一定程度上降低了第二近邻的查找准确率,因此增加了匹配成功率,从而提升了查全率,降低了查错率(参见式(1)). 并行 PCPF 查询匹配算法由于在查询过程中没有完全遍历所有的子树以求取第一近邻,所以其匹配效果较串行情况较差,有时接近于 BBF 算法,但总体匹配效果仍略好于 BBF.

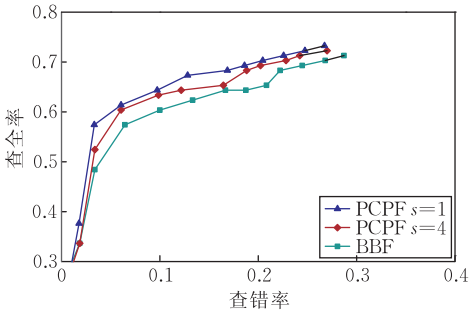


图 10 PCPF 与 BBF 查全率、查错率对比

由此,PCPF 算法效率的提升非但没有导致匹配精度的下降,反而能比 BBF 算法取得更好的匹配效果.

6 总结与展望

本文针对多媒体数据库对象检索中的高维特征向量匹配问题,基于多核处理器架构,提出一种新的并行压缩优先过滤索引(PCPF). PCPF 采用并行构建算法量化特征向量并构建多个索引子分支,并利用优先队列并行过滤多个子分支得到近似近邻候选集来提高查询匹配效率. 实验证明,PCPF 并行构建与查询匹配算法在多核处理器架构下取得了良好的并行效果,较经典的 BBF 算法在查询匹配性能上有大幅提升,同时提高了匹配效果,并且没有明显地增加构建时间.

在实际多媒体检索应用中,特征向量分布具有一定的规律,合理的聚类十分重要,不但能有效提升检索效率,更能增加检索结果的准确性. PCPF 索引构建时仅考虑数据集的均匀划分,而没有加入聚类的考虑. 对此,后续工作将通过分析向量集合的分布特性,研究合理的聚类对于索引性能与效率的影响.

致 谢 感谢中国科学院国家天文台提供了嫦娥二号月球遥感影像数据作为本文的实验数据来源之一(月球空间数据影像库)!

参 考 文 献

[1] Datta R, Joshi D, Li J et al. Image retrieval: Ideas, influences, and trends of the new age. ACM Computing Surveys, 2008, 40(2): 5:0-5:60

[2] Lowe D G. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 2004, 60(2): 91-110

[3] Weber R, Schek H J, Blott S. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces//Proceedings of the VLDB. New York, 1998: 194-205

[4] Ferhatosmanoglu H, Tuncel E, Agrawal D et al. Vector approximation based indexing for non-uniform high dimensional data sets//Proceedings of the CIKM. McLean, 2000: 202-209

[5] Jagadish H V, Ooi B C, Tan K L et al. iDistance: An adaptive B^+-tree based indexing method for nearest neighbor search. ACM Transactions on Database Systems, 2005, 30(2): 364-398

[6] Slaney M, Casey M. Locality-sensitive hashing for finding nearest neighbors. IEEE Signal Processing Magazine, 2008, 25(2): 128-131

[7] Tao Y, Yi K, Sheng C et al. Quality and efficiency in high dimensional nearest neighbor search//Proceedings of SIGMOD. Providence, 2009: 563-576

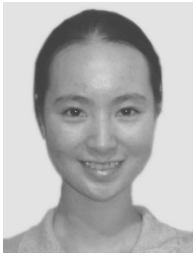
[8] Zhang Jun-Qi, Zhou Xiang-Dong, Wang Mei et al. Cluster splitting based high dimensional metric space index B^+-tree. Journal of Software, 2008, 19(6): 1401-1412(in Chinese)
(张军旗，周向东，王梅等. 基于聚类分解的高维度量空间索引 B^+-Tree. 软件学报，2008，19(6)：1401-1412)

[9] Wang Guo-Ren, Huang Jian-Mei, Wang Bin et al. A High dimensional data indexing technique based on max gap space mapping. Journal of Software, 2007, 48(6): 1419-1428(in Chinese)
(王国仁，黄健美，王斌等. 基于最大间隙空间映射的高维数据索引技术. 软件学报，2007，48(6)：1419-1428)

[10] Yang Heng, Wang Qing, He Zhou-Can. Multiple randomized sub-vectors quantization hashing for high-dimensional image feature matching. Journal of Computer-Aided Design & Computer Graphics, 2010, 22(3): 494-502(in Chinese)
(杨恒，王庆，何周灿. 面向高维图像特征匹配的多次随机子向量量化哈希算法. 计算机辅助设计与图形学学报，2010，22(3)：494-502)

[11] Beis J S, Lowe D G. Shape indexing using approximate nearest-neighbor search in high-dimensional spaces//Proceedings of the CVPR. San Juan, 1997: 1000-1006

[12] Bay S H, Tuytelaars T, Gool L V. SURF: Speeded up robust features//Proceedings of the European Conference on Computer Vision. Graz, 2006: 404-417



CHEN Hui-Zhong, born in 1982, Ph. D. candidate. Her current research interests include content-based image retrieval, intelligent information process.

CHEN Yong-Guang, born in 1962, Ph. D., professor, Ph. D. supervisor. His research interests include electromagnetic environment simulation and evaluation, electronic warfare simulation.

JING Ning, born in 1963, Ph. D., professor, Ph. D. supervisor. His research interests include GIS and spatial database.

CHEN Luo, born in 1973, Ph. D., professor. His research interests include GIS and database.

Background

Nowadays, local features are being widely used for content-based multimedia retrieval in multimedia databases because of their distinctiveness, stability and abundance. However, when applying the local features for retrieval, the high dimensionality and the large numbers always make the matching being a very time-consuming process. Thus, it is very necessary to research on the high-dimensional index which can speed up matching the features.

The key point of high-dimensional features matching can be considered as a 2-NN search problem. Existing high-dimensional indexes are mostly focused on kNN search and are not very suitable for the matching. Algorithms such as Best-Bin-First (BBF) are designed especially for the problem. In this paper, we propose a novel index PCPF which is based on the idea of the priority queue of BBF and also adopt the idea of vector approximation. To make the full use of modern multi-processor CPUs, it is designed as a parallel index structure. We propose the constructing and query algorithms for PCPF using multi-threading technologies, which can be run in parallel with a multi-processor CPU. The experimental results on the benchmark and real dataset show that PCPF works efficiently and outperforms the classical method with better effect.

This research was partially supported by the National Natural Science Foundation of China (No. 60902036); the National High Technology Research and Development Program (863 Program) of China (Nos. 2008AA12A211, 2011AA12A306). The main purpose of these research projects is constructing high-performance data management environments for large amount of earth and lunar remote sensing data. The work introduced in this paper belongs to an important part of them, which aims at retrieving managed images by the contents from the large scale database. It is proposed to speed up the retrieval.