基于多表频繁项投票和桶映射链的快速检索方法

高毫林*① 彭天强② 李弼程① 郭志刚①
①(信息工程大学信息工程学院 郑州 450002)
②(河南工程学院计算机工程与科学系 郑州 451191)

摘 要：为解决基于随机映射的高维向量快速检索方法位置敏感哈希存在的随机性强和内存消耗大两个问题，在E2LSH(Exact Euclidean Locality Sensitive Hashing)的基础上提出了基于多表频繁项投票和桶映射链的快速检索方法。该方法用检索结果构造基准索引矩阵，并对基准索引矩阵进行频繁项投票和校正得出最终索引来降低检索的随机性；桶映射链利用 E2LSH 的数据划分特性减少检索时读入内存的数据点的数目，以此来降低内存消耗。实验证明该方法能减弱检索的随机性，并有效地降低检索的内存消耗。这对于提高大规模信息检索尤其是图像检索的可行性有着较大的作用。

关键词：信息检索；位置敏感哈希；随机性；内存消耗；频繁项投票；桶映射链
中图分类号：TN391 **文献标识码**： A **文章编号**：1009-5896(2012)11-2574-08
DOI: 10.3724/SP.J.1146.2012.00548

A Fast Retrieval Method Based on Frequent Items Voting of Multi Table and Bucket Map Chain

Gao Hao-lin① Peng Tian-qiang② Li Bi-cheng① Guo Zhi-gang①
①(*Institute of Information Engineering*, *Information Engineering University*, *Zhengzhou* 450002, *China*)
②(*Department of Computer Science and Engineering*, *Henan Institute of Engineering*, *Zhengzhou* 451191, *China*)

Abstract: To solve the problem of strong randomicity and high memory cost of fast retrieval method Locality Sensitive Hashing (LSH) based on random projection, a fast retrieval method is presented based on multi table frequent items voting and bucket map chain on the basis of Exact Euclidean Locality Sensitive Hashing (E2LSH). The method constructs an index matrix with retrieval vectors, and performs frequent items voting and calibration on this matrix to decrease the randomocity. It also reduces the number of points loaded into memory by making use of the data partition property of E2LSH to decrease the memory cost. The experiments show that this method can decrease the randomicity and efficiently reduce the memory cost of retrieval. This is very important for increasing the feasibility of large scale information retrieval especially image retrieval.

Key words: Information retrieval; Locality Sensitive Hashing (LSH); Randomicity; Memory cost; Frequent items voting; Bucket map chain

1 引言

相似性搜索在数据压缩、数据库和数据挖掘、信息检索、图像和视频数据库、机器学习、模式识别、统计和数据分析等方面有着重要的作用。它们所研究的对象一般能用相关特征的集合或高维空间中的点表示，并根据给出的查询点找出距其最近的点。这些点的维数范围很大，基于树的方法虽然在一定程度上解决了高维检索问题。但当维数较高时，它们和线性穷尽搜索相比几乎没有什么进展甚至会退化到线性搜索[1]。

位置敏感哈希[1,2](Locality Sensitive Hashing, LSH)是当前高维空间中近似近邻(Approximate Near Neighbor, ANN)搜索速度最快的解决方法，LSH 在汉明空间进行搜索，E2LSH[3,4]是对 LSH 的改进之一，在欧氏空间进行搜索。与基于树的索引方法相比，它们不但复杂度低、支持维数高，而且检索时间大大缩短。目前 LSH 在图像检索[5−8]、复制检测[9]和化学混合物搜索[10]中都有应用。文献[11]还将 LSH 作为与主成分分析(PCA)和 K-Means 类似的视觉特征到视觉单词的映射方法。

LSH 和 E2LSH 作为 ANN 解决方案的基础在于相似性搜索并不一定要得出精确的最近邻。在许多情况下，近似最近邻提供的结果已经比较让人满意

2012-05-09 收到，2012-07-27 改回
国家自然科学基金(60872142)资助课题
*通信作者：高毫林 holygao@126.com

了，关键在于它能以更小的代价完成目标。但正是这个基础使得 LSH 不可避免地存在一定的随机性，即不同的哈希表得出的检索结果有所不同。这样的随机性如果得不到好的控制被进一步传播，就会影响算法的性能。对于随机性的减弱，本文对多表检索的结果进行融合，根据检索向量构造基准索引向量，然后由基准索引向量构造基准索引矩阵，再对基准索引矩阵进行频繁项投票得出准最终索引，最后对准最终索引进行校正得出接近真实情况的最终索引。

E2LSH 还存在另外一个缺点是内存消耗大，虽然和基于树的方法相比已经有所降低，但对于大规模数据集而言它消耗的内存仍然很大，甚至会超过系统内存上限。文献[12−15]已指出了这一点，但他们并没有给出改进方法。这是因为它是基于主存的方法，在每次执行检索时，需要先建立索引，而建立索引前又需要把所有数据点读入内存，这样内存消耗就很大。实际上，E2LSH 在进行数据点映射时实现了数据划分，在特征空间相近的点被映射到同一个桶中。这样，利用桶映射信息只读入和查询点相关的点就可以减少内存消耗。而且，静态数据集数据点相对位置不变，不需要重复建立索引。对于数据的增删，也可以只处理相关的桶。这就是采用桶映射链降低内存消耗的基本思想。它将每个桶用一个映射表示，整个数据集用一个桶映射链表示。搜索时根据查询点的桶标志在链中先确定桶映射的位置，然后读入该桶映射数据，再计算查询点与读入桶映射中的数据点的相似度。这样可以显著降低检索程序的内存消耗。

2　欧式空间LSH实现方案

早期 LSH 的哈希函数是针对二进制汉明空间 $\{0,1\}^d$ 中的点的。对于欧式空间而言，虽然通过空间嵌入能把算法扩展到 l_2 范数，但这增加了算法的查询时间和错误率，以及算法的复杂度。而 E2LSH 不需要嵌入就可以直接工作在欧式空间中的点上，它还可以工作在任何 $p\in(0,2]$ 的 l_p 范数上。该方案继承了原始 LSH 的两个特点。其一是它很适合于维数很高但稀疏的数据点。尤其是当 d 是向量中非零元素的最大数目时，算法的运行时间限会保持几乎不变。该特点是其它数据结构所不具有的。其二是如果数据满足一定的有界增长特性(bounded growth property)，它可以很快地找到精确的近邻。对于点 q，$c>1$，$N(q,c)$ 代表 S 中 q 的 c-近似近邻的数目。如果 $N(q,c)$ 以 c 的函数按次指数(sub-exponentially)增长，并且给定常量因子去近似 q 到它的最近邻居的距离，那么，算法能够以固定概率在 $O(d\log n)$ 时间内找到最近的邻居 v。特别地，如果 $N(q,c)=O(c^b)$，运行时间是 $O(\log n+2^{O(b)})$ [4]。

E2LSH 是基于 p-定分布函数的，并且对于 $p\in(0,2]$ 的所有值都适用。稳定分布被定义为归一化独立同分布变量和的极限。它的定义如下：如果存在 $p\geq 0$ 对于 n 个向量 $\boldsymbol{v}_1,\cdots,\boldsymbol{v}_n$ 和分布 $\mathcal{D}$ 的独立同分布随机变量 $\boldsymbol{X}_1,\cdots,\boldsymbol{X}_n$，变量的线性组合 $\sum_i \boldsymbol{v}_i\boldsymbol{X}_i$ 和变量 $\left(\sum_i|\boldsymbol{v}_i|^p\right)^{1/p}\boldsymbol{X}$ 分布相同，其中 $\boldsymbol{X}$ 是分布 $\mathcal{D}$ 的随机变量，那么 $\Re$ 上的分布 $\mathcal{D}$ 被称为 p-稳定分布。这时，两个向量 $\boldsymbol{v}_1,\boldsymbol{v}_2$ 投影距离 $(\boldsymbol{a}\cdot\boldsymbol{v}_1-\boldsymbol{a}\cdot\boldsymbol{v}_2)$ 的分布与 $||\boldsymbol{v}_1-\boldsymbol{v}_2||_p\ \boldsymbol{X}$ 的分布相同，$\boldsymbol{a}$ 是从 p-稳定分布独立选择的 d 维向量。哈希函数的形式为 $h(\boldsymbol{v})=\lfloor(\boldsymbol{a}\cdot\boldsymbol{v}+b)/w\rfloor$，它把 d 维向量 $\boldsymbol{v}$ 映射到整数集上，b 是一个在 $[0,w]$ 上均匀选取的实数。并定义 $g(\boldsymbol{v})=(h_1(\boldsymbol{v}),\cdots,h_k(\boldsymbol{v}))$，其中 $h_i\in\mathcal{H}$，进一步选择 L 个函数 $g_1,\cdots,g_L$，来增加算法的查全率。

3　E2LSH检索随机性分析

E2LSH 哈希函数族由多重函数构成，函数的个数与点的冲突概率密切相关。因此，算法对参数非常敏感，不好的参数会使检索随机性较强，偏离正确的结果。好的参数则会降低随机性，增加检索准确性。

3.1 随机性与分段长度的关系

由于 LSH 采用的是随机映射的方法，所以其检索结果不可避免地存在随机性。它的随机性由算法发现真实最近邻的概率决定，也就是这两个点冲突的概率，而这个概率又与 p 稳定分布函数 $\mathcal{D}$ 有关。设 $f_p(t)$ 代表 p 稳定分布绝对值的概率密度函数，对于每个 h，两个点 v 和 h 冲突的概率即

$$P\left[\left\lfloor\frac{(\boldsymbol{a}\cdot v)+b}{w}\right\rfloor=\left\lfloor\frac{(\boldsymbol{a}\cdot q)+b}{w}\right\rfloor\right] \tag{1}$$

这两个点冲突需要满足两个条件：$|(a\cdot v)-(a\cdot q)|<w$，各分段边界不能落在 $(a\cdot v)$ 和 $(a\cdot q)$ 之间。第 1 个条件等价于 $|(v-q)\cdot a|<w$，该式进一步等价为 $\|\|v-q\|Z|\equiv|cZ|<w$，其中 $Z\sim\mathcal{D},c=||v-q||_p$。对于第 2 个条件，分段边界落在两点投影之间的概率为 cz/w，那么两点冲突的概率即为：$p(c)=\int_0^{w/c}f_p(r)(1-cr/w)\mathrm{d}r$，进行变量替换令 $t=rc$，可得

$$p(c)=\int_0^w\frac{1}{c}f_p\left(\frac{t}{c}\right)\left(1-\frac{t}{w}\right)\mathrm{d}t \tag{2}$$

对于固定的参数 w，$p(c)$ 随 c 单调递减，当 $r_1=R$, $r_2=cR$ 时，它的两个重要的值为 $p_1=p(R)$ 和

$p_2 = p(cR)$。p_1 表示 $v \in B(q, r_1)$，两点哈希值相等的概率的最小值，它与 w 的关系如图 1 所示，可见，分段长度越大，两点映射后哈希值相等即映射到同一个桶中的概率越大。p_2 表示 $v \notin B(q, r_2)$ 时，两点哈希值相等的概率的最大值，它与 w 和 c 的关系如图 2 所示，可见，距离越大的两点映射到同一桶中的概率越小。

3.2 随机性与 k 和 L 的关系

上节所述LSH的随机性主要与分段长度和两点间距离有关。而分段长度在初始化时就已经确定，点间距离又是不可调整的，所以减弱检索的随机性需要调整 LSH 函数族的参数。这是因为 LSH 虽然能使距离较近的点哈希后冲突概率较大，但是，即使两个点距离较近，它们冲突的概率并不能保证足够大。而只能保证这个概率大于 1/2。即假设存在一个点 $v^* \in X$ 使得 $\|q - v^*\| \le R$，对于某个 $i \in \{1, \cdots, L\}$，$g_i(v^*) = g_i(q)$ 成立的概率大于 1/2。其证明如下：固定 i，$P[g_i(v^*) = g_i(q)] \ge {p_1}^k$，其中 $k = \log_{1/p_2} n$，则有 ${p_1}^k = {p_1}^{\log_{1/p_2} n} = n^{-\frac{\log_2(1/p_1)}{\log_2(1/p_2)}}$，令 $\rho = \log_2(1/p_1) / \log_2(1/p_2)$，那么对某个 i 而言，冲突的概率是 $P[\exists i, g_i(v^*) = g_i(q)] \ge 1 - (1 - n^{-\rho})^L$。当 $L = n^\rho$，该概率为 $1 - (1 - n^{-\rho})^{n^{-\rho}} \ge 1 - 1/e > 1/2$。这说明，当 $L = 1$ 时，即仅用一个哈希表进行检索是不够的，如果对于准确率有较高的要求，需要增加表的个数。因此我们用实验说明 k 和 L 对算法随机性的影响。

考虑一个查询点 q 和一个近邻 $v \in B(q, R)$，令 $p_1 = p(R)$，q 和点 v 冲突的概率是 $P[g(q) = g(v)] \ge p_1^k$，那么，对于所有 L 个函数 q 和 v 不冲突的概率是 $(1 - p_1^k)^L$，这样它们在某个函数 g_i 上冲突即 $g_j(q) = g_j(v)$ 的概率就可表示为 $p_r = 1 - (1 - p_1^k)^L$。该概率与 k 和 L 变化关系如图 3 和图 4 所示。

可见，不管对于固定的点间距离还是固定的分段间隔，冲突概率都随 L 的增大而增大，随 k 的增大而减小。这是因为 L 越大，就意味着表的个数越多，冲突的概率增大。而对一个表而言，k 增大反而增强了随机性，使数据集进行哈希后得到更多的桶，导致冲突概率下降。比如，如果所有 k 次哈希运算都与查询点落入同一个桶认为该点是最近邻，那么它的概率是 p_1^k，增加 k 值会使这个概率降低。增加 w 虽然会增加落入每个桶中的点数，从而增加冲突概率。但是，为了得到最近邻结果需要搜索所有与查询点落入同一个桶中的点，这样就会增加查询时间。如果 $1 - (1 - p_1^k)^L \ge 1 - \delta$，则 $L \ge \log_{1-p_1^k} \delta$，$k$ 的取值要能使完成一次查询的时间最短。

4 基于多表频繁项投票的弱随机检索方法

对于 E2LSH 具有的随机性的改善，文献[16]提出的 LSH Forest 减少了需要确定的参数的个数，部分地解决了这个问题。文献[17]设计了自适应的 LSH 搜索算法模型，动态的为每次查询确定参数，文献[18]指出可以使用 ERC-Forest[19]降低表哈希结果的随机性。但这些方法没有综合利用多个表的检索结果，难以达到较好的效果。针对以上对 E2LSH 随机性的分析，本文采用多哈希表投票法减弱算法的随机性，它的核心是计算矩阵的频繁项，也就是在多个表中出现频次较高的检索结果。这样使得检索信息得以综合利用。

4.1 基于频次矩阵列元素投票

从映射的观点来看，每个 g_i 可以对数据点在一个方向上进行映射。真实的近邻可能在某个方向上映射后的点距离查询点映射后的点较远，但不可能在各个方向上都“不幸”[20]。非真实的近邻可能在某个方向上距离映射后距离查询点较近，但不可能在各个方向上都“幸运”。这样，可以通过增加 L 来提高发现真实近邻的概率。但由于每个表都对应于一个数据集映射的结果，所以如果使用多个表进行检索需要对检索结果进行融合。我们采用频繁项投票的方法，选取 L 个表对应的检索结果中出现次数最多的图像。对 L 个表进行检索的结果记为 $I_{p,q}(x_n)$，

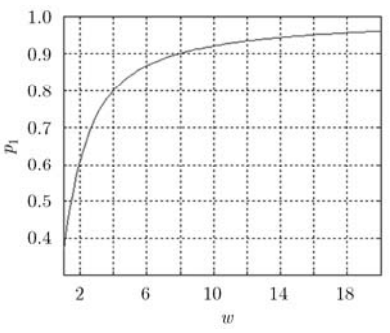

图 1 冲突概率与分段长度的关系

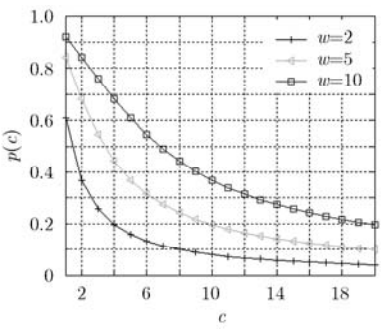

图 2 冲突概率最小值与两点距离的关系

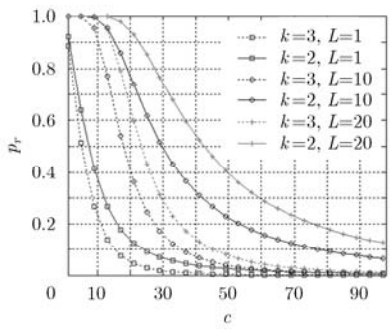

图 3 冲突概率与 k, L 和两点距离的关系

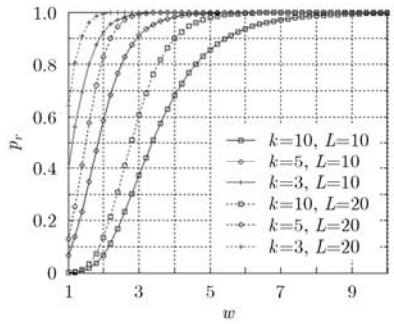

图 4 冲突概率与 k, L 和分段长度的关系

其中 $p=1,\cdots,L$, $q=1,\cdots,k$, x_n 表示图像在图像集的初始序号， $x_n\in[1,N]$, $n=1,\cdots,k$, N 表示图像集的图像数目。用矩阵表示为 $\boldsymbol{I}=\begin{pmatrix} I_{11} & \cdots & I_{1k} \\ \vdots & \ddots & \vdots \\ I_{L1} & \cdots & I_{Lk} \end{pmatrix}$。矩阵每一行对应于一个表的检索结果，最终结果可通过对矩阵每一列进行投票的方式得出，也就是统计矩阵每列各元素频次，将出现次数最多的元素(频繁项)作为最终结果，其结果记为 I_r , $r=1,\cdots,k$ ， 其中 $I_r=\arg(\max \mathrm{Freq}(I_{1r},\cdots,I_{Lr}))$ ， $\mathrm{maxFreq}(*)$ 表示取频次最高元素。

4.2 多表频繁项投票方法

多表频繁项投票法减弱随机性主要分为以下几步：

(1)通过 AP(Average Precision)值及检出率从多个表中选取基准索引向量。由于检出个数及排序情况不同，仅仅利用 AP 值的高低难于直接反映检索结果的优劣。可以选取 AP 较高的结果，再在此基础上选取检出率较高的结果作为最终索引的基准索引向量。

(2)构造基准索引矩阵。由于上述基准索引向量的长度可能不同，所以需要对部分基准索引向量进行补零。补零的位置通过计算修正最小编辑距离(Modified Minimum Edit Distance, MMED)确定。MMED 是指将一个向量在不同位置插入某元素补成与另外一个向量等长后两者编辑距离的最小值。如向量 $\boldsymbol{X}=(x_1,\cdots,x_m)$ 和向量 $\boldsymbol{Y}=(y_1,\cdots,y_n)$ ， 其中 $m\neq n$ 。若 $n=\max(m,n)$ ， 则在 $\boldsymbol{X}$ 中插入元素 x_p ，出 $\boldsymbol{X}'=(x_1,\cdots,x_p,\cdots,x_n)$ ， 使其与 $\boldsymbol{Y}$ 等长，其中 p 表示插入位置。然后计算两者的编辑距离，该距离为两个向量对应位置不同元素个数之和，插入位置距离强制为 1，即 $d_p(\boldsymbol{X}',\boldsymbol{Y})=\sum_{i\neq p}\mathrm{sim}(x_i,y_i)+1$ ， 其中 $\mathrm{sim}(x_i,y_i)=\begin{cases}1, & x_i\neq y_i\\ 0, & x_i=y_i\end{cases}$。不同的插入位置 p 得出的编辑距离 ED 不同，使得 ED 最小的 p 就是需要补零的位置， $p=\arg(\min(d_p(\boldsymbol{X}',\boldsymbol{Y})))$ 。得出 p 后对 $\boldsymbol{X}$ 补零，就可进一步构造基准索引矩阵。对第(1)步得出的基准索引向量，计算它们之间的 MMED，并构造基准索引矩阵。

(3)计算基准索引矩阵频繁项。得出基准索引矩阵后利用 4.1 节的频繁项投票方法可得频繁项及其频次，频繁项即为准最终索引。

(4)用非基准索引向量对准最终索引进行索引校正，也就是寻找这些频繁项在非基准索引向量中的相对位置，来确定最终索引，这样就充分利用了距离保持信息。

5 基于桶映射链的低内存消耗检索方法

虽然 LSH 算法能使检索速度大大提高，但内存消耗大是它的主要局限。虽然存在一些改进算法，如文献[21, 23]在一定程度上降低了内存消耗值，但这些算法当图像规模较大时内存消耗会迅速增加，并可能超过内存上限，从而造成系统崩溃。这就难以满足大规模图像库检索的要求，需要对内存消耗进一步降低。

5.1 E2LSH 空间消耗分析

标准的基于 LSH 的算法确保 $n^{1+\rho}$ 的空间消耗，参数 $\rho<1$ 并依赖于 c 。文献[1,2]的算法 $\rho=1/c$ ，文献[4]的算法的空间消耗为 $O(dn+n^{1+\rho(c)})$ ， 其中 ρ 低于 $1/c$ 。文献[22]算法的空间消耗为 $O(dn^{1+\rho(c)})$ ，其中 $\rho(c)$ 任意接近 $1/c^2$ ， 明显高于文献[4]，该文献的另外一种空间有效的算法可达到 $O(dn+n\lg O(1))$ 的空间消耗。当 $c=2$ 时，该算法指数趋近于 0.25，而文献[4]中大约为 0.45。文献[23]的空间消耗虽然与文献[22]相似，但其 ρ 值偏大。文献[24]说明基于 LSH 的算法很难达到 $\rho<0.462/c^2$ 的水平，这些基于 LSH 算法的空间消耗的上界见表 1 所示，为便于对比，其中 $\lg n$, $1/c$ 和 dn 等因子已忽略。

ρ 对空间和时间消耗的大小起着重要的作用，它随分段长度 w 和两点间距离 c 的变化规律分别如图 5 和图 6 所示。可见，当 c 增大时， ρ 的值不断减小。而 w 只在接近于 1 的较小区间内对 ρ 有明显的影响，这个区间与数据集中点的距离有关。所以通过调整分段长度的值而改变内存消耗是不可行的。由于算法所使用的数据结构已经固定，我们从数据结构的存储方式进行改进，以降低内存消耗。也就是根据算法在检索时只需要对部分数据点进行运算的特点将数据集哈希结果用外存保存，仅将需要运算的点调入内存。这样就可以大大降低内存消耗。

表1 LSH相关算法空间消耗的上界

算法	度量	空间消耗	备注
文献[1,2]算法	Hamming	$n^{1+\rho}$	$\rho = 1/c$
文献[4]算法	Euclid	$n^{1+\rho}$	$\rho < 1/c$
文献[23]算法	Euclid	n	$\rho = O(1/c)$
文献[22]算法	Euclid/ Euclid	$n^{1+\rho}/n$	$\rho = 1/c^2 + O(1) / \rho = O(1/c^2)$

5.2 基于桶映射链的低内存消耗检索方法

由于 E2LSH 在进行哈希运算时间数据点映射到了桶中，而且距离相近的点映射到同一个桶中。所以如果检索时只读入查询点所在桶包含的数据点而不是读取全部数据点，就会显著降低内存消耗。这就需要把全体数据点用文件映射的方式组织起来。具体表示方式如下：每个桶用一个映射表示，多个桶映射组成一个链，即每个表由一个桶映射链表示，一个桶映射链就包含了全体数据点，它对应于一个哈希表其数据结构如表 2 所示。

采用这样的数据结构可使一个数据集由多个桶映射表征，进一步将每个桶映射 buckMap 由一个文件 indexFile 记录，在查询时就不需要读入所有桶映射文件，实现低内存消耗的检索。基于桶链映射低内存消耗检索方法主要过程如下：先计算查询点的 buckIndex，然后根据 buckIndex 搜索桶映射链，找出查询点所在的桶映射，再读入该桶映射包含的数据点，最后计算精确的相似度，返回结果。具体过程如表 3 所示。

表 2 桶映射链数据结构构建

准备：
创建包含所有点的桶映射链 buckMapChain
buckMapChain =<buckMap_1, $\text{buckMap}_2, \cdots, \text{buckMap}_j$>
一个链包含所有的桶映射 buckMap，每个桶映射
buckMap_j ={buckIndex, pointVector}
桶映射由 buckIndex 区分，其中的 pointVector 包含映射到该桶中的所有点
pointVector =[pointID_1, $\text{pointID}_2, \cdots, \text{pointID}_k$]
pointID_k 指明序号为 k 的点

表 3 桶映射链搜索流程

(1)查询：
(2)为查询点 q 计算 buckIndex；
(3)在 buckMapChain 中找到标志为 buckIndex 的 buckMap；
(4)读取 buckMap 中的所有点 pointVector;
(5)计算 q 与 pointVector 中各点 pointID_k 的相似度；
找到相似点并返回结果。

该方法可以使内存消耗从 $O(dn)$ 降到 $O(d\tilde{n})$，$\tilde{n}$ 表示包含查询点的桶映射中点的数目。可见，内存消耗降低程度与 $\tilde{n}$ 有关，$\tilde{n}$ 的值越小，内存消耗降低的程度越低。

6 弱随机低内存消耗快速检索方法

6.1 弱随机检索实验

我们选取了 TRECVID 部分图像进行实验，检索的目标为“歌手”，如图 7(a)所示第 1 幅图像，共有 24 幅相关图像。每次实验需要先用 E2LSH 对数据集建立索引，由于哈希函数的确定需要产生随机数，所以桶分配的结果会有不同，即使相关桶包含的点数几乎相同，检索结果也会有所差异。图 7 给出了两次实验对查询图像的检索结果，分别如图 7(a)和图 7(b)所示。可见，检索结果的排序是不同的。

同一组相关图像的不同相似度排序体现了 E2LSH 方法的随机性，为了获取真实的索引，需要对多次检索结果进行融合。而多次检索的结果是存在一定规律性的。虽然每次哈希运算桶分配情况不同，检索结果中同一图像的序号可能不同，但综合

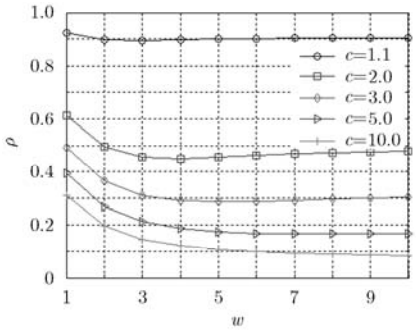

图 5 ρ 与 w 的关系

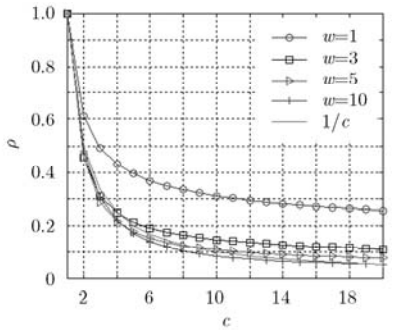

图 6 ρ 与 c 的关系

(a)实验1检索结果

(b)实验2检索结果

图 7 不同哈希表在同一数据集上对同一幅图像检索结果

多次结果来看，图像间相似度基本不变。这说明哈希函数对图像有距离保持作用，同时，检索结果是互相独立的。另外，虽然有些检索结果会遗漏部分相关图像，但不影响其它检索结果。因此，对多个表的检索结果采用投票的方法进行融合，可以减弱随机性，得出接近真实情况的最终结果。我们采用多表投票法确立最终索引。

弱随机检索实验主要分为以下几步：

(1)将检索结果 1–5 对应的向量分别记为 $\boldsymbol{X}_1$ – $\boldsymbol{X}_5$，其中 AP 较高的结果有 $\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \boldsymbol{X}_5$，在此基础上选取检出率较高的 $\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3$ 结果作为最终索引的基准索引向量。

(2)计算修正最小编辑编辑 MMED 并求最小值，可知 $\boldsymbol{X}_1$ 和 $\boldsymbol{X}_3$ 在 $p = 3$ 时 MMED 最小，所以将

$\boldsymbol{X}_3$ 补成(18 19 0 25 28 26 20 22 29 27 32 34 24 23 37 39 31 33 40 35 41 30 36 38)，进一步构造基准索引矩阵

$$\boldsymbol{I}=\begin{pmatrix}18 & 19 & 21 & \cdots & 36 & 38\\ 19 & 18 & 21 & \cdots & 36 & 38\\ 18 & 19 & 0 & \cdots & 36 & 38\end{pmatrix}$$

(3)得出基准索引矩阵后计算频繁项，结果为 $\begin{pmatrix}18 & 19 & 21 & 25 & 28 & 26 & 20 & 22 & \cdots & 38\\ 2 & 2 & 2 & 3 & 3 & 2 & 2 & 3 & \cdots & 3\end{pmatrix}$，该矩阵第 1 行表示频繁项，第 2 行表示频繁项频次，得准最终索引为(18 19 21 25 28 26 20 22 ⋯ 36 38)。

(4)用 $\boldsymbol{X}_4,\boldsymbol{X}_5$ 对(18 19 21 25 28 26 20 22 ⋯ 36 38)进行索引验证，不需要对准最终索引进行调整，准最终索引即为最终索引。

用直接基于特征计算相似度的方法进行验证，结果表明它与新方法方的检索结果几乎相同，说明新方法比较好地反映了真实的近似程度。

6.2 低内存消耗检索实验

该实验从 TRECVID 图像集中选取 4 组数目分别为 10k, 50k, 100k 和 150k 的图像进行实验。实验主要记录 E2LSH 和基于桶映射链的 E2LSH (Bucket Map Chain-Exact Euclidean Locality Sensitive Hashing, BMC-E2LSH)检索方法的内存消耗情况，主要包括两组实验。一是对采用两种方法的图像检索程序进行内存消耗监测，以 50 ms 为间隔进行记录。采用 E2LSH 方法的程序主要包括读入特征集、建立索引、选择查询点和进行搜索 4 部分，采用方法的程序主要对查询阶段进行检测，因为建立桶映射链的准备阶段对于静态数据集只需要运行 1 次，而且可以离线进行，对于数据增加只需要对新增数据进行映射。BMC-E2LSH 主要包括选择查询点和进行搜索两部分。图 8 给出了在数目为 10k 和 100k 的情况下两种方法的内存消耗，为了便于显示，该图主要给出了检索进程各阶段内存消耗曲线。可见，对于两组图像集来说，BMC-E2LSH 方法都能使内存消耗显著下降。

第 2 组实验主要记录 4 组图像集下两种方法的阶段性内存消耗值。这是为了便于对两种检索方法不同阶段内存消耗进行对比，E2LSH 方法记录建立索引前、建立索引后、进行搜索前和进行搜索后 4 个点的内存消耗，BMC-E2LSH 方法记录进行搜索前和进行搜索后两个点的内存消耗，其结果分别如图 9 和图 10 所示。由图 9 可见，采用 E2LSH 方法检索时，10k 幅图像建立索引后和完成搜索后内存消耗分别为 28.8 MB 和 32.2 MB，而 150k 幅图像的这两个数字达到了 160.3 MB 和 327.1 MB。在图 10 中，采用 BMC-E2LSH 方法检索时，10k 幅图像完成搜索后内存消耗为 11.6 MB, 150k 幅图像完成搜索后内存消耗为 24.3 MB，都明显小于采用 E2LSH 方法检索时的值。

为了对两种检索方法的内存消耗进行对比，图 11 给出了 E2LSH 方法从建立索引到完成搜索的内存消耗和 BMC-E2LSH 方法从开始搜索到完成搜索的内存消耗。可见，在图像数目为 10k 和 150k 时，E2LSH 方法的内存消耗为 13.9 MB 和 166.9 MB，而 BMC-E2LSH 方法的内存消耗为 0.62 MB 和 13.2 MB。后者能使检索的内存消耗明显降低。

7 结束语

当前高维向量快速搜索的方法主要是 E2LSH，这在多个研究方向都有所体现。但对减弱它的随机性的文献还很少，对内存消耗大的问题虽然有涉及但还没有好的解决方法。随机性的产生主要由哈希函数族的随机性和参数的不确定性引起，内存消耗大主要是因为它是基于主存的方法，在建立索引前把全部数据点读入内存。本文提出了基于多表频繁项投票和桶映链的方法解决这两个问题。多表频繁项投票可以降低 E2LSH 的随机性，桶映射链可以降低 E2LSH 的内存消耗，这些在实验结果中得到了证明。

应该说明的是，桶映射链降低内存消耗的方法主要与数据集中与查询点相关的点的数目有关。图 11 显示对于 150k 幅图像，桶映射链方法内存消耗仅为 E2LSH 方法的约 1/13，这个比例虽然不是固定的，但是新方法能达到明显降低内存消耗的目的。

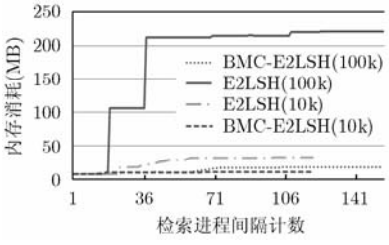

图 8 检索进程内存消耗情况监测

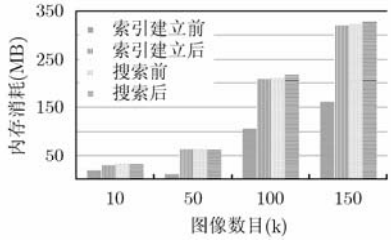

图 9 E2LSH 检索进程内存消耗

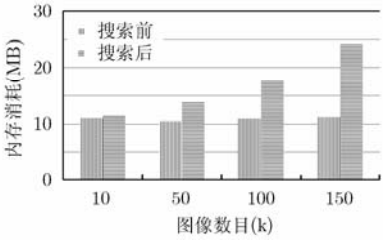

图 10 BMC-E2LSH 检索进程内存消耗

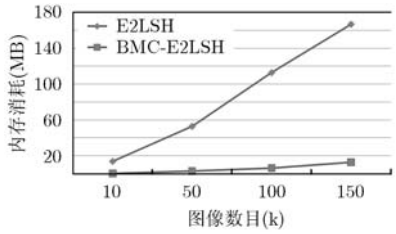

图 11 两种检索方法的内存消耗

参 考 文 献

[1] Indyk P and Motwani R. Approximate nearest neighbors: towards removing the curse of dimensionality[C]. Proceedings of the Symposium on Theory of Computing, Dallas, Texas, USA, ACM, 1998: 604–613.

[2] Gionis A, Indyk P, and Motwani R. Similarity search in high dimensions via hashing[C]. Proceedings of the 25th International Conference on Very Large Data Bases. Edinburgh, Scotland, Morgan Kaufmann, 1999: 518–529.

[3] Andoni A and Indyk P. E2LSH 0.1 user manual[OL]. http://www.mit.edu/~andoni/LSH/manual.pdf, October 20, 2011.

[4] Datar M, Immorlica N, Indyk P, *et al.*. Locality sensitive hashing scheme based on p-stable distributions[C]. Proceedings of the ACM Symposium on Computational Geometry, New York, USA, ACM, 2004: 253–262.

[5] Jegou H, Douze M, and Schmid C. Improving bag-of-features for large scale image search[J]. *International Journal of Computer Vision*, 2010, 87(3): 316–336.

[6] Gruman K Efficiently searching for similar images[J]. *Communications of the ACM*, 2010, 53(6): 85–94.

[7] Avrithis Y, Tolias G, and Kalantidis Y. Feature map hashing: sub-linear indexing of appearance and global geometry[C]. Proceedings of the International Conference on Multimedia, New York, NY, USA, 2010: 231–240.

[8] Mu Ya-dong and Yan Shui-cheng. Non-metric locality-sensitive gashing[C]. Proceedings of the 24th AAAI Conference on Artificial Intelligence, Atlanta, Georgia, USA, 2010: 539–544.

[9] Liu Zhu, Liu Tao, and David G. Effective and scalable video copy detection[C]. Proceedings of the ACM SIGMM International Conference on Multimedia Information Retrieval, New York, USA, 2010: 119–128.

[10] Cao Yi-qun, Jiang Tao, and Thomas G. Accelerated similarity searching and clustering of large compound sets by geometric embedding and locality sensitive hashing[J]. *Bioinfomatics*, 2010, 26(7): 953–959.

[11] Wu Lei and Hua Xian-sheng. Elements of visual concept analysis[J]. *Studies in Computational Intelligence*, 2011, 346: 679–717.

[12] He Jun-feng, Radhakrishnan R, and Chang S F. Compact hashing with joint optimization of search accuracy and time[C]. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, USA, 2011: 753–759.

[13] Liu Wei, Wang Jun, Kumar S, *et al.*. Hashing with graphs[C]. Proceedings of the 28th International Conference on Machine Learning, Bellevue, WA, USA, 2011: 522–529.

[14] Jegou H, Douze M, and Schmid C. Hamming embedding and weak geometric consistency for large scale image search[C]. Proceedings of the 10th European Conference on Computer Vision, Berlin, Heidelberg, Germany, ACM, 2008: 304–317.

[15] Jegou H, Douze M, and Schmid C. Improving bag-of-features for large scale image search[J]. *International Journal Computer Vision*, 2010, 87(3): 316–336.

[16] Bawa M, Condie T, and Ganesan P. Lsh forest: self-tuning indexes for similarity search[C]. Proceedings of the 14th International Conference on World Wide Web, New York, NY, USA, 2005: 651–660.

[17] Dong Wei, Wang Zhe, Josephson W, *et al.*. Modeling LSH for performance tuning[C]. Proceedings of Association of Computing Machinery Conference on Information and Knowledge Management, Napa Valley California, USA, 2008: 669–678.

[18] Mu Ya-dong, Sun Ju, and Han Tony X. Randomized locality sensitive vocabularies for bag-of-features model[C]. Proceedings of Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision, Berlin, Heidelberg, Germany, 2010: 748–751.

[19] Moosmann F, Nowak E, and Jurie F. Randomized clustering forests for image classification[J]. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 2008, 30(9): 1632–1646.

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

[21] Lu Q, Josephson W, and Wang M. Multi-probe LSH: efficient indexing for high-dimensional similarity search[C]. Proceedings of the 24rd International Conference on Very Large Data Bases, Vienna, Austria, ACM, 2007: 950–961.

[22] Andoni A and Indyk P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions[J]. *Communications of the ACM*, 2008, 51(1): 117–122.

[23] Panigrahy R. Entropy-based nearest neighbor algorithm high dimensions[C]. Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, Miami, Florida, USA, ACM, 2006: 1185–1195.

[24] Motwani R, Naor A, and Panigrahy R. Lower bounds on locality sensitive hashing[C]. Proceedings of the ACM Symposium on Computational Geometry, Sedona, Arizona, USA, ACM, 2006: 253–262.

高毫林：　男，1979年生，博士生，研究方向为图像检索.

彭天强：　男，1978年生，讲师，研究方向为智能信息处理.

李弼程：　男，1970年生，教授，研究方向为智能信息处理.

郭志刚：　男，1975年生，讲师，研究方向为海量信息检索.