Towards Optimal Indexing for Relevance Feedback in Large Image Databases+

Sharadh Ramaswamy and Kenneth Rose, *Fellow, IEEE*

***Abstract*—Motivated by the need to efficiently leverage user relevance feedback in content-based retrieval from image databases, we propose a fast, clustering-based indexing technique for exact nearest-neighbor search that adapts to the Mahalanobis distance with a varying weight matrix. We derive a basic property of point-to-hyperplane Mahalanobis distance, which enables efficient recalculation of such distances as the Mahalanobis weight matrix is varied. This property is exploited to recalculate bounds on query-cluster distances via projection on known separating hyperplanes (available from the underlying clustering procedure), to effectively eliminate noncompetitive clusters from the search and to retrieve clusters in increasing order of (the appropriate) distance from the query. We compare performance with an existing variant of VA-File indexing designed for relevance feedback, and observe considerable gains.**

***Index Terms*—CBIR, image database, index, relevance feedback, similarity search.**

I. Introduction

ADVANCEMENTS in semiconductor technology, magnetic storage hardware, and the growth of the Internet has spawned new database applications for multimedia data, such as multimediainformation systems, CAD/CAM, geographical information systems (GIS), medical imaging where large amounts of data are stored in databases for future retrieval. On the other hand, the proliferation of personal digital multimedia devices, such as digital cameras and video recorders, has resulted in the need for new applications that can handle these data, such as image search engines, personal digital libraries and albums, etc. While searching based on keywords is the current paradigm in many search engines, keywords are not necessarily the most effective representation of multimedia information. For example, it would be ineffective to mine databases of medical images based on keywords or "metadata" if the goal is to discover hidden correlations that are unknown and, hence, have not been quantified through metadata. Clearly, content-based image search and retrieval is a more appropriate paradigm. Given that collections of digital multimedia are ever growing, organization and management of such image/multimedia repositories is crucial.

Typically, images are represented by fixed-length feature vectors and the measure of similarity between two images is assumed to be proportional to the Euclidean distance between their feature vectors. Popular descriptors are the color histogram [1], the color layout descriptor [2], scale-invariant feature transforms [3], shape descriptors [4], [5], etc. Recently, a combination of texture features (extracted through Gabor filters) and color features (histograms) have been found to be efficient descriptors for a broad class of images and form a part of the MPEG-7 multimedia standard (see [6]). Useful feature vectors are often high dimensional, such as the 60-D texture descriptors of [6]. Last, we note that a "bag of features" representation can also be followed, where a variable number of feature vectors are used to represent each image. Similarity is now measured by the single/multiple-histogram intersection [7]–[9], Earth Mover's Distance (EMD) [10], or the χ^2 distance [11]. However, we focus on fixed-length features and omit the "bag of features" representation from discussion in subsequent sections.

II. Multidimensional Indexing

Similarity search is the search for elements in the image database most similar to the query image. A popular query model is the k-nearest neighbor (kNN) query, where given a query image, the k most similar images are extracted from the database. We distinguish between exact kNN search, where the kNNs are guaranteed to be returned and approximate kNN search, where the accuracy of search is sacrificed for increased search speed. In either setup, since the feature vectors themselves are large in number and of high dimensionality, it is more cost effective to store them on a hard-storage device, typically a hard disk. While computing speeds have largely followed Moore's law, speeds of hard disk devices have lagged behind and are limited by the presence of mechanical components [12]. Hence, the search for nearest neighbors in large, high-dimensional data-sets is challenging, with the search time being overwhelmingly dominated by IO operations (e.g., hard disk access times).

A. Hard Disk Access and Performance Metric

Data stored on a hard disk device can be accessed serially or through random accesses, which have different access costs. The IO time is proportional to the amount of data accessed and is determined by the number of sequential and random hard disk accesses caused. Irrespective of the access strategy, data are always stored and retrieved from the disk in units of *disk blocks* or *pages* (see [12] for a more detailed description). Typically, the time spent in accessing one page with a random IO is significantly more (by at least an order of magnitude) when

Manuscript received January 23, 2009; revised July 04, 2009. First published July 31, 2009; current version published November 13, 2009. This work was supported in part by the National Science Foundation under Grants IIS-0329267 and CCF-0728986; in part by the University of California MICRO program; in part by Applied Signal Technology, Inc.; in part by CISCO Systems, Inc.; in part by Dolby Laboratories, Inc.; in part by Qualcomm, Inc.; and in part by Sony Ericcson, Inc. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Kiyoharu Aizawa.

The authors are with the Signal Compression Lab, Electrical and Computer Engineering, University of California, Santa Barbara CA 93106 USA (e-mail: rsharadh@ece.ucsb.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2009.2028929

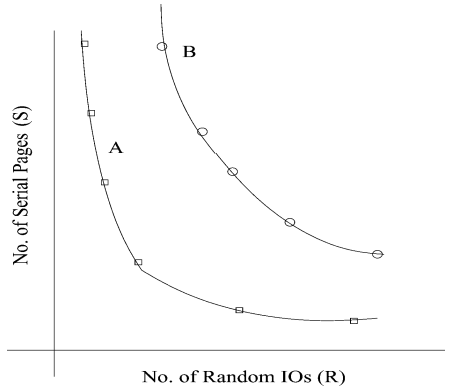

Fig. 1. Proposed performance metric to compare hypothetical indexing schemes A, B.

compared with sequential access. But, random IOs have the advantage of reading only the required pages. Hence, random IOs would be faster in retrieving pages that are spaced far apart while less costly sequential access of pages would be optimal if the required pages are spaced close together (even if not contiguously). The optimal search strategy would need to optimize between the number of sequential accesses and the number of random accesses caused.

To index a database is to organize the database so as to enable faster search for required nearest neighbors. The common performance metrics for nearest neighbor search have been to count page accesses or the response time. However, page accesses may involve both serial disk accesses and random IOs, which have different costs. On the other hand, response time (*seek times* and *latencies*) is tied to the hardware being used, and, therefore, the performance gain/loss would be platform dependent. Additionally, in any search strategy, by varying parameters, a sequence of indexes with different sequential and random IO costs are possible. To compare two indexing approaches, one would need take into account the full range of solutions possible.

Consider Fig. 1, where the IO performance of two hypothetical schemes A, B are plotted. Note that in our performance metric, we count separately the number of sequential accesses S and number of random disk accesses R. Given the performance in terms of the proposed metric, i.e., the (S,R) pair, it is possible to estimate total number of disk accesses or response times on different hardware models. The number of page accesses is evaluated as $S + R$. If T_{seq} represented the average time required for one sequential IO and T_{rand} for one random IOs, the average response time would be $T_{\text{res}} = T_{\text{seq}}S + T_{\text{rand}}R$. Now, if (as in Fig. 1) given the same number of sequential accesses, the random accesses of one scheme (A) is less than that of the other (B), then clearly the first scheme (A) also incurs lower access times on *all* hardware and lower page access costs and, hence, is the better indexing approach. Given these favorable properties, in all our experiments, we use this performance metric to compare different indexing schemes.

B. Early Multidimensional Indexing

In the general database search literature, several index structures exist that facilitate search and retrieval of multidimensional data. The typical approach is to recursively partition the data-set and represent each partition by a bounding geometric structure, such as rectangles (see R-tree [13]), spheres (see SS-tree [14]), etc. These regular geometric data-structures are used to prune the search space and return the nearest neighbors. In low-dimensional spaces, such methods outperform sequential scan.

However, it has been observed that the performance of many multidimensional index structures degrades as the feature space dimensionality increases and they eventually *underperform* sequential scan (see [15]). This is seen as a consequence of "the curse of dimensionality" [16]. Due to the exponential growth of hypervolume with dimensionality, a very large portion of the space is actually empty and recursive feature space representation by regular bounding geometric structures, such as spheres and rectangles, is ineffective. In fact, it has been shown that in some cases almost no part of the search space can be excluded by using such hierarchical data-structures [15]. Hence, searching on such naive index structures leads to a large number of needless and costly random disk accesses, making it slower than the simple sequential scan. In a famous result, Weber *et al.* [15] have shown that whenever the dimensionality is above 10, these methods are outperformed by a simple sequential scan.

C. Modern Approaches to Multidimensional Indexing

Recent approaches to exact kNN indexing have been inspired by compression. Here, a compressed representation of the database is maintained and the search is performed over this compressed representation. This includes the famous vector approximation (VA)-file approaches [15], [17], where scalar quantization of feature dimensions is utilized to prune the search space, and clustering/vector quantization (VQ) approaches [18]–[20]. There are also a few schemes which attempt transformations of the feature space to a 1-D representation [21], [22] to speed up search. Finally, several researchers have opined that feature vectors are poor approximations of user perceptions. Therefore, even a exact nearest neighbor search is inevitably approximate. Conversely, by returning only approximate nearest neighbors significant time in IO-processing can be saved. Notable approaches include VA-LOW [23], locality-sensitive hashing [24], and clustering [25], [26].

III. Relevance Feedback in Image Retrieval

While the Euclidean distance metric is popular within the multimedia indexing community, it is by no means the perceptually "correct" distance measure. Hence, significant research activity (in content-based image retrieval) has been directed toward learning Mahalanobis (or generalized Euclidean) distances (see [27]–[30]) and nonlinear transforms (see [31]).

A. Mahalanobis Weight Adaptation

In subsequent sections, we assume that the data-set $\mathcal{X}$ consists of d-dimensional feature vectors drawn from $\mathbb{R}^d$, i.e., $\mathcal{X} \subset \mathbb{R}^d$. Given a positive semi-definite weight matrix W, we define the Mahalanobis distance parameterized by W as the distance function

$$d_W(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y}))^T W^{-1}(\mathbf{x}-\mathbf{y})}. \quad (1)$$

We clarify that the Mahalanobis distance here is a general matrix weighted distance and not in the narrow sense where the weight is restricted to be the covariance. Note that $W \in \mathbb{R}^{d^2}$. Also, note that we require $W > \mathbf{0}$, which implies that $d_W(,)$ is a metric. Clearly, the Mahalanobis distance measure has more degrees of freedom than the Euclidean distance and by proper updation (or *relevance feedback*), has been found to be a much better estimator of user perception of similarity (see [32], [33], [27]). The goal in relevance feedback is to adapt the distance measure to match user expectations, by making the search an interactive process. Here, in each iteration a set of results is retrieved and the user provides feedback on the relevance of each result. If Mahalanobis distance is employed, this is used to update the weight matrix for the next iteration. Two popular methods for weight adaptation are MARS [34] and MindReader [32].

Let $\mathcal{Q}^+ = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_l\}$ be the set of positive samples returned from the earlier iteration, where $\mathbf{q}_k = [q_{k1}, q_{k2}, \ldots q_{kj} \ldots q_{kd}]^T, \forall 1 \leq k \leq l$. Similarly, we define $Q_i^+ = \{q_{ki}\}_{k=1}^{l}, 1 \leq i \leq d$.

In MARS, the weight matrix is constrained to be a diagonal matrix. If A is a finite set of real scalars, we use $VAR(A)$ to denote the variance of A. If W_{ij} is the (i,j)th diagonal entry of W, in the next iteration

$$W_{ii} = \frac{1}{\sqrt{VAR[Q_i^+]}} \tag{2}$$

and $W_{ij} = 0, \forall i \neq j$.

The MindReader approach attempts to estimate the full weight matrix W by minimizing the average distance from the query $\mathbf{q}$ to the elements of $\mathcal{Q}^+$, with the determinant $|W|$ constrained to be unity. At the same time, the query vector is also modified. The "optimal" W is found to be

$$W = \alpha C^{-1} \tag{3}$$

where C is the sample correlation matrix of $\mathcal{Q}^+$ and $\alpha \in \mathbb{R}$ is a constant of proportionality.

However, the MindReader approach can be affected by the singularity of C because of the relatively high dimensionality of the space involved and the low cardinality of $\mathcal{Q}^+$. A solution proposed in [32] employs the Moore-Penrose inverse instead of C^{-1}. Nevertheless, perceptual quality in experiments reported in [33] is significant degraded. Hence, the authors of [33], attempt to combine the MARS and MindReader approaches within a unifying framework, where the weight matrix W constrained to be diagonal if C is singular. If the weights are to be updated in batch fashion (after accumulating several queries) rather than the sequential online fashion, various optimization or information theoretic approaches to learn the new weight matrix have been proposed [28]–[30], [35].

B. Efficiency of Indexing With Relevance Feedback

Multidimensional search indexes are typically designed assuming a fixed Mahalanobis distance measure that is known in advance. The weight matrix is diagonalized and the data are correspondingly rotated and scaled into a new set of feature dimension s prior to indexing. However, in relevance feedback applications, the weight matrix changes with time and renders most standard indexes ineffective and very slow. Clearly, a truly effective relevance feedback application requires a new indexing approach.

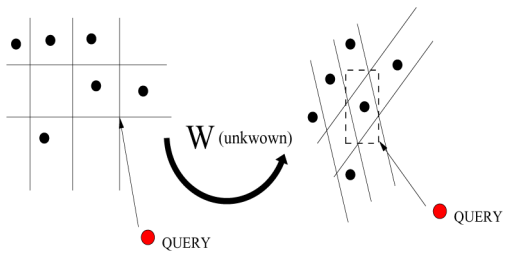

Fig. 2. VA-file under an unknown linear transformation.

A very popular and effective technique employed to overcome the curse of dimensionality is the vector approximation file (VA-File) [15]. VA-File partitions the space into hyper-rectangular cells, aligned with the co-ordinate axes. Each dimension is quantized uniformly and the quantization indices are stored of each feature vector in the so called *approximation file*, on the hard-disk. Upper and lower bounds on the distance to the query from each cell are estimated and these are used to prune the data-set of those vectors that are not likely to be good candidates. The final set of candidate vectors are read from the hard-disk and the nearest neighbor are determined.

A change in the Mahalanobis weight matrix is equivalent to rotating and skewing the bounding rectangles into uniform hyper-parallelograms. The method of [36] fits minimum bounding rectangles that contain these parallelograms (see Fig. 2). Note that these hyper-rectangles are larger and overlapping. A new set of distance bounds to these rectangles are evaluated and used to prune the search space.

In this paper, we consider a clustering approach towards similarity search. Clustering or vector quantization (VQ) is often the method of choice for signal compression, as it can exploit correlations across dimensions [37]. When adapted to database search, what is required is an efficient way of cluster ranking in order of relevance to the query. We expand on and subsume our early results on the design of clustering based indexes for similarity search [20], [38]. The data-set is clustered using a standard clustering/VQ technique (such as k-means) and only relevant ("nearest") clusters are retrieved during query processing. Clusters are retrieved until the kth nearest neighbor discovered so far is closer to the query than all remaining clusters, which guarantees that the k nearest neighbors have been discovered. We further note that with only one cluster, the indexing technique degenerates to the sequential scan, i.e., sequential scan is an extreme *special case*.

Central to such a search technique is the ability to tightly bound the distance to a cluster, without accessing the elements of the cluster [20]. We show how effective estimates of query-cluster distances can be performed while adapting to a *changing* weight matrix and how this filters out irrelevant regions of the database, thus providing significant speed-ups over known techniques. Consequently, the proposed clustering based approach is effective for relevance feedback in image databases.

IV. Point-To-Hyperplane Distance

Let $d_W(\mathbf{x},\mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T W(\mathbf{x}-\mathbf{y})}$ be the distance between any two feature vectors $\mathbf{x}$ and $\mathbf{y}$. Without loss of generality, we assume W is symmetric and positive definite, i.e., $d_W(\cdot,\cdot)$ is a metric. Let $H(\mathbf{a},b) = \{\mathbf{x} : \mathbf{a}^T\mathbf{x} + b = 0\}$ be a hyperplane and $\mathbf{y}$ a point in the space outside of it. Then

$$d_W(\mathbf{y},H) = \min_{\mathbf{x}\in H} d_W(\mathbf{x},\mathbf{y}) = \sqrt{\min_{\mathbf{x}\in H} d_W(\mathbf{x},\mathbf{y})^2}. \quad (4)$$

Using Lagrange multiplier λ, let $J = (\mathbf{x}-\mathbf{y})^T W(\mathbf{x}-\mathbf{y}) + \lambda(\mathbf{a}^T\mathbf{x}+b)$

$$\begin{aligned}
\frac{\partial J}{\partial \mathbf{x}} = 0 \Rightarrow \mathbf{x}^* - \mathbf{y} &= -\frac{1}{2}\lambda W^{-1}\mathbf{a} \\
\frac{\partial J}{\partial \lambda} = 0 \Rightarrow \mathbf{a}^T\mathbf{x}^* + b &= 0 \\
\Rightarrow \lambda = \frac{2(\mathbf{a}^T\mathbf{y}+b)}{\mathbf{a}^T W^{-1}\mathbf{a}}, \mathbf{x}^* - \mathbf{y} &= \frac{-(\mathbf{a}^T\mathbf{y}+b)W^{-1}\mathbf{a}}{\mathbf{a}^T W^{-1}\mathbf{a}} \\
\Rightarrow (\mathbf{x}^*-\mathbf{y})^T W(\mathbf{x}^*-\mathbf{y}) &= \frac{(\mathbf{a}^T\mathbf{y}+b)^2}{\mathbf{a}^T W^{-1}\mathbf{a}} \\
\Rightarrow d_W(\mathbf{y},H) = d_W(\mathbf{x}^*,\mathbf{y}) &= \frac{|\mathbf{a}^T\mathbf{y}+b|}{\sqrt{\mathbf{a}^T W^{-1}\mathbf{a}}}.
\end{aligned}$$

We note that if W were the identity matrix, then the formula reduces to the known version for Euclidean distance. Next consider two weight matrices W_1 and W_2, it is easy to note that

$$\frac{d_{W_1}(\mathbf{y},H)}{d_{W_2}(\mathbf{y},H)} = \sqrt{\frac{\mathbf{a}^T W_2^{-1}\mathbf{a}}{\mathbf{a}^T W_1^{-1}\mathbf{a}}}. \quad (5)$$

In other words, the ratio of point-to-hyperplane distances under differing weight matrices is *independent* of the point $\mathbf{y}$ (as well as the fixed translation b).

V. Adaptive Cluster Distance Bounding

It is easy to show that for any positive definite W, the shortest path between two points is along the straight line passing through the two points. Now, given a cluster $\mathcal{X}_m$, the query $\mathbf{q}$ and a hyperplane H that lies between the cluster and the query (a "*separating hyperplane*," see Fig. 3), by simple geometry it is easy to see that for any $\mathbf{x} \in \mathcal{X}_m$

$$\begin{aligned}
d_W(\mathbf{q},\mathbf{x}) &\geq d_W(\mathbf{q},H) + d_W(\mathbf{x},H) \\
&\geq d_W(\mathbf{q},H) + \min_{\mathbf{x}\in\mathcal{X}_m} d_W(\mathbf{x},H) \\
&= d_W(\mathbf{q},H) + d_W(\mathcal{X}_m,H) \\
\Rightarrow d_W(\mathbf{q},\mathcal{X}_m) &\geq d_W(\mathbf{q},H) + d_W(\mathcal{X}_m,H). \quad (6)
\end{aligned}$$

We focus on the second term, $d_W(\mathcal{X}_m,H)$, the "*support*". Had W been known in advance, this could have been evaluated offline and stored. Instead, let us denote the weight matrix used during clustering as W_0. Then, (5) implies

$$d_W(\mathcal{X}_m,H) = \sqrt{\frac{\mathbf{a}^T W_0^{-1}\mathbf{a}}{\mathbf{a}^T W^{-1}\mathbf{a}}}\, d_{W_0}(\mathcal{X}_m,H) \quad (7)$$

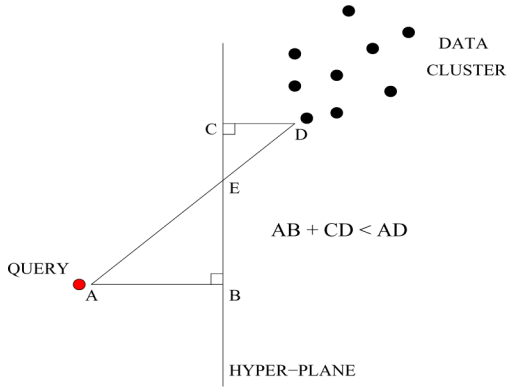

Fig. 3. Hyperplane bound.

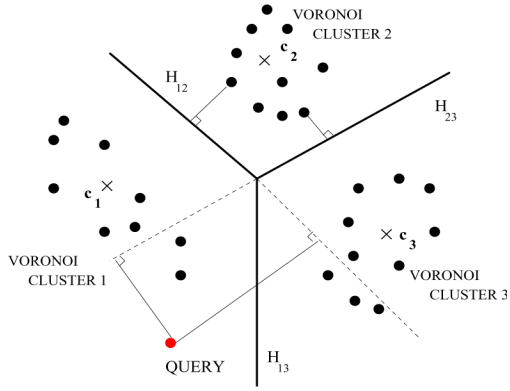

Fig. 4. Cluster distance bounding.

which demonstrates that it is *unnecessary* to reevaluate the support due to change in weight matrix after the clustering phase. Without loss of generality, in subsequent discussion, we will assume that $d_{W0}(\cdot,\cdot)$ is the Euclidean distance, and drop the suffix W_0.

If $\mathcal{H}_{\text{sep}}$ represents a countably finite set of separating hyperplanes (that lie-between the query $\mathbf{q}$ and the cluster $\mathcal{X}_m$)

$$d_W(\mathbf{q},\mathcal{X}_m) \geq \max_{H\in\mathcal{H}_{\text{sep}}} \{d_W(\mathbf{q},H) + d_W(\mathcal{X}_m,H)\}. \quad (8)$$

The second lower bound presented in (8) can be used to tighten the lower bound on $d_W(\mathbf{q},\mathcal{X}_m)$. Next, we note that the *boundaries* between clusters generated by the K-means algorithm are *linear hyperplanes*. If $\mathbf{c}_1$ and $\mathbf{c}_2$ are centroids of two clusters $\mathcal{X}_1$ and $\mathcal{X}_2$, and H_{12} the boundary between them, then $\forall \mathbf{y} \in H_{12}$

$$\begin{aligned}
d(\mathbf{c}_1,\mathbf{y}) &= d(\mathbf{c}_2,\mathbf{y}) \\
\Rightarrow \|\mathbf{c}_1\|_2^2 - \|\mathbf{c}_2\|_2^2 - 2(\mathbf{c}_1-\mathbf{c}_2)^T\mathbf{y} &= 0.
\end{aligned}$$

Therefore, the hyperplane $H_{12} = H(-2(\mathbf{c}_1-\mathbf{c}_2), \|\mathbf{c}_1\|_2^2 - \|\mathbf{c}_2\|_2^2)$ is the boundary between the clusters $\mathcal{X}_1$ and $\mathcal{X}_2$. We emphasize that these hyperplane boundaries *need not be stored*, as they can be generated during run-time from the centroids. It is straightforward to show that: Given a query $\mathbf{q}$ and a hyperplane H_{mn} that separates clusters $\mathcal{X}_m$ and $\mathcal{X}_n$, it lies between the query and cluster $\mathbf{X}_m$ ***if and only if*** $d(\mathbf{q},\mathbf{c}_m) \geq d(\mathbf{q},\mathbf{c}_n)$ (see Fig. 4).

A. Reduced Complexity Hyperplane Bound

For evaluation of the lower-bound of (6) and (8), we would need to precalculate and store $d(H_{mn}, \mathcal{X}_m)$ for all cluster pairs (m, n). With K clusters, there are $K(K-1)$ distances that need to be precalculated and stored, in addition to the cluster centroids themselves. The total storage for all clusters would be $O(K^2 + Kd)$. This heavy storage overhead makes the hyperplane bound, in this form, impractical for a very large number of clusters. However, we can loosen the bound in (8) as follows:

$$\begin{aligned} d_W(H, \mathcal{X}_m) &= \sqrt{\frac{\|\mathbf{a}\|_2^2}{\mathbf{a}^T W^{-1} \mathbf{a}}} d(\mathcal{X}_m, H)\} \\ &\geq \sqrt{\frac{\|\mathbf{a}\|_2^2}{\mathbf{a}^T W^{-1} \mathbf{a}}} \min_{H \in \mathcal{H}_{\text{sep}}} d(\mathcal{X}_m, H)\} \\ \Rightarrow d_W(\mathbf{q}, \mathcal{X}_m) &\geq \max_{\mathcal{H}_{\text{sep}}} \left\{ d_W(\mathbf{q}, H) + \sqrt{\frac{\|\mathbf{a}\|_2^2}{\mathbf{a}^T W^{-1} \mathbf{a}}} d_{\text{sep}} \right\} \end{aligned}$$

where $d_{\text{sep}} = \min_{H \in \mathcal{H}_{\text{sep}}} d(\mathcal{X}_m, H)$. This means that for every cluster $\mathcal{X}_m$ we would only need to store one distance term d_{sep}, thus reducing the total storage to $O(K(d+1))$. For the special case when $d_W(\cdot, \cdot)$ is itself Euclidean, i.e., no weight adaptation, see [20]. For small K, even $\|\mathbf{a}\|_2$, for all cluster boundaries $\mathbf{a}$, can be calculated offline and stored. Even otherwise, we note that it is IO time (and not processor time) which is the bottleneck in query processing.

VI. Clustering and Index Generation

The first step in index construction is the creation of Voronoi clusters. Voronoi clusters are created through nearest neighbor partitioning of the feature space and have linear hyperplane boundaries. There exist several techniques of clustering the data-set, from the fast K-means algorithm [39] (which requires multiple scans of the data-set) and generalized Lloyd algorithm (GLA) [37] to methods such as BIRCH [40], which require only a single scan of the data-set. The output of any of these algorithms can be a starting point. From each of the K clusters detected by a generic clustering algorithm, a pivot is chosen, i.e., K pivot points in all. Then the entire data-set is canned and each data-element is mapped to the nearest pivot. Last, data mapping to the same pivot are grouped together to form Voronoi clusters (see Algorithm 1). This would lead to slight re-arrangement of clusters, but this is necessary to retain piecewise linear hyperplane boundaries between clusters. We believe the centroid is a good choice as a pivot. Thus, quick Voronoi clustering, with possibly only a single scan of the entire data-set, can be achieved using any generic clustering algorithm.

We note that the K-means, GLA and BIRCH algorithms are fast and can generate reliable estimates of cluster centroids, from sub-samples of the data-set. Typically, for K clusters, even a sub-sample of size 100 K is sufficient. As we shall see, for the range of clusters we are considering, this would be overwhelmingly smaller than the data-set. Faster index construction would be possible by allowing for hierarchical and multistage clustering. However, only the clusters at the leaf level are returned.

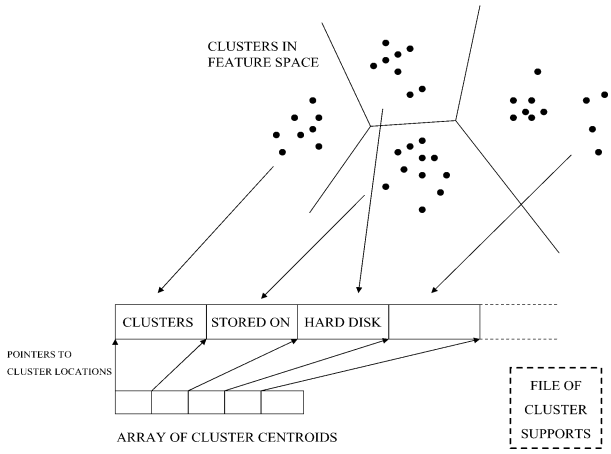

Fig. 5. Proposed index structure.

Algorithm 1 Voronoi-Clusters$\mathcal{X}$, K)

1: //Generic clustering algorithm returns
   //K cluster centroids
   $\{\mathbf{c}_m\}_{=1}^K \leftarrow$GenericCluster$(\mathcal{X}, K)$
2: set $l = 0, \mathcal{X}_1 = \varphi, \mathcal{X}_2 = \varphi, \ldots, \mathcal{X}_K = \varphi$
3: **While** $l < |\mathcal{X}|$ **do**
4: $l = l + 1$
5: //Find the centroid nearest to data element $\mathbf{x}_l$
   $k = \arg\min_m d(\mathbf{x}_l, \mathbf{c}_m)$
6: //Move $\mathbf{x}_l$ to the corresponding Voronoi partition
   $\mathcal{X}_k = \mathcal{X}_k \bigcup \{\mathbf{x}_l\}$
7: **end while**
8: return $\{\mathcal{X}_m\}_{=1}^K, \{\mathbf{c}_m\}_{=1}^K$

We tested several clustering techniques including GLA and BIRCH, and the results were largely similar. While it is possible to also optimize the clustering itself, that is not our goal in these experiments.

A. Storage and Retrieval Strategy

Elements within the same cluster are stored together (contiguously). We retain the cluster centroids $\mathbf{c}_m$ and maintain pointers from each centroid to the location of the corresponding cluster on the hard-disk. We also maintain in a separate file the distance (bounds) of each cluster from its bounding hyperplanes. We note that the total storage is $O(K.d+K^2)$ and $O(K.(d+1))$ real numbers, for the full and reduced complexity hyperplane bounds respectively, where K is the number of clusters.

Fig. 5 is representative of our index. We estimate the query cluster distances through the hyperplane bounds, and thereafter retrieve clusters in order of distance from the query. After each cluster is read, the list of kNNs so far is evaluated/updated. If the distance of the kth best candidate so far is less than the distance to the next closest cluster, the search stops since there already exist at least k better candidates, i.e., the kNNs have been found. Otherwise, the next closest cluster is read till all clusters have been read or the kNNs have been found.

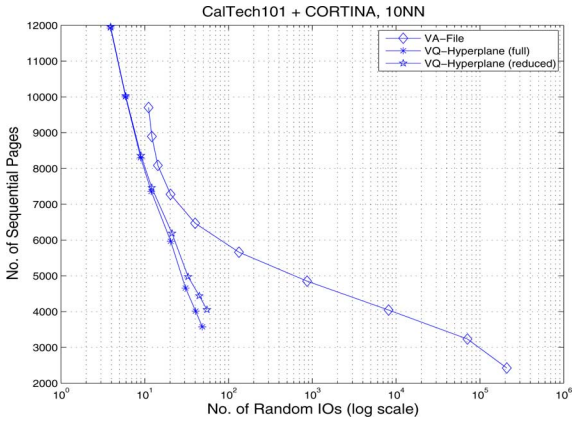

Fig. 6. IO performance with MARS (diagonal weight matrix).

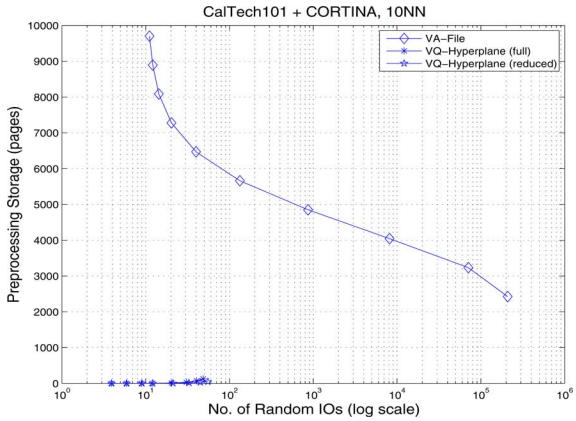

Fig. 7. Preprocessing storage with MARS.

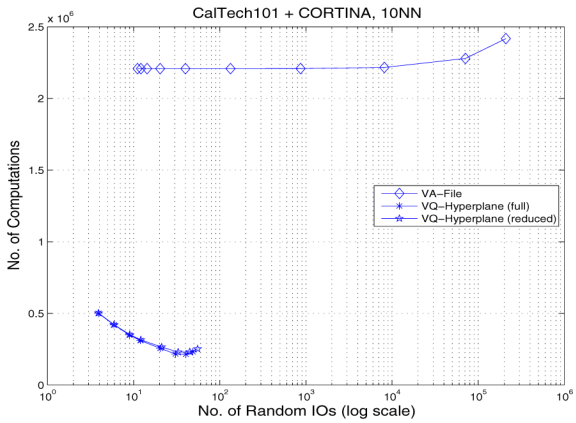

Fig. 8. Computational costs with MARS.

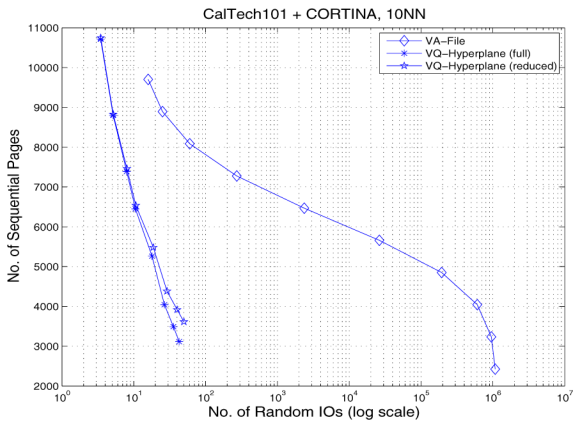

Fig. 9. IO performance with random weight matrix.

VII. Experimental Results

We compared the performance of our index (henceforth referred to as "VQ-Hyperplane (full)" for full complexity and "VQ-Hyperplane (reduced)" for reduced storage complexity bounds resp.) with a well-known variant of VA-File [36] that is adapted to leverage relevance feedback.

A. Data-Set CORTINA-Caltech101

This data-set consists of a 1 103 271 element sub-sample of the CORTINA image data-set[1] and consists of 48-D MPEG-7 texture features.[2] Within this we embed 733 images extracted from 11 classes of the CalTech 101 data-set [41] which have strong texture signatures. The classes considered are "accordion," "barrel," "bass," "brain," "beaver," "cougar body," "Leopards," "wild cat," "hedgehog," "platypus," and "soccer ball." These images classes are *a priori* unknown to the system and the images are used as queries for performance evaluation. For each query, the ten nearest neighbors (10NN) were mined. We also assumed a *page size* of 8 kB.

In one set of experiments, we evaluated the (diagonal) weight matrix for each query-class according to principles and heuristics established in MARS [34]. In another set of experiments, the weight matrix was modeled as $W = U^T \Lambda U$. The orthonormal matrix U was generated randomly and the eigenvalues were uniformly distributed between 0 and 10. We present results from one such realization of W, that is representative of general performance.

We evaluated the performance of VA-File at various quantization levels (3–12 bits per dimension) and the VQ method for varying numbers of clusters (10–400 clusters). We note that with the MARS weights there is *no rotation* of the feature space. In MARS, we notice moderate gains for the VQ-hyperplane methods in the IO performance (Fig. 6). For the same number of sequential accesses, random disk IOs are reduced by factors ranging from 1.5X to 200X. But we note that the VA-File requires $\approx$100 X more preprocessing storage (Fig. 7) and also $\approx$ 10X higher computational costs (Fig. 8). This is because the VA-File maintains a separate compressed representation for *each element* of the database, as a result of which $O(N)$ distance computations and storage of $.15N - 0.3N$ are needed, where N is the size of the database. In order to reduce the number of costly random access reads in the VA-File, the quantization resolution in each dimension needs to be increased, which again results in larger approximation files. In contradistinction, the VQ method reduces random IO reads by *reducing* the number of clusters.

On the other hand, for the random weight matrix, our indexes are able to consistently reduce the number of random IO reads as compared with VA-File, when allowed (roughly) the same number of sequential disk accesses (see Fig. 9). This is because of the rotation and scaling of the space, which significantly loosens the distance bounds of the VA-File. At 5-bit quantization for the VA-File and 300 clusters for our indexes, we note an $\approx 10^5$X reduction in costly random IO reads. Clearly, at

[1]See http://cortina.ece.ucsb.edu/.

[2]http://scl.ece.ucsb.edu/datasets/CORTINA_HTD_1Million.bin

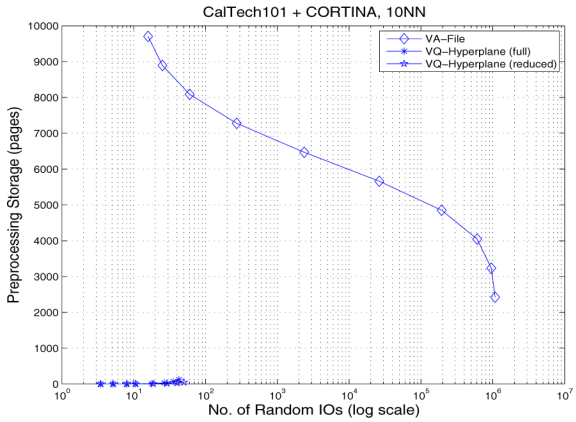

Fig. 10. Preprocessing storage with random weight matrix.

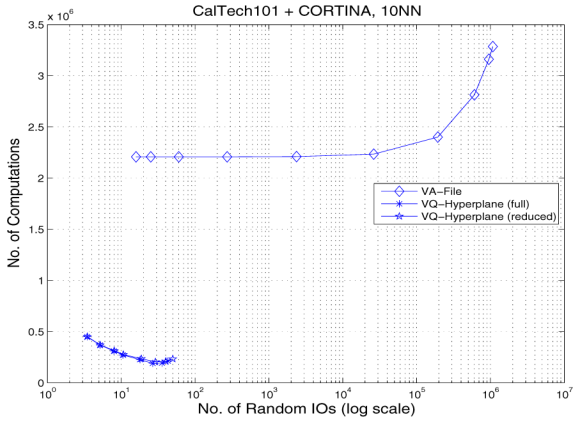

Fig. 11. Computational costs with random weight matrix.

this quantization level, efficiency of the VA-File in pruning the search space is almost nil, i.e., it almost underperforms the sequential scan. The performance degradation from the full complexity to the reduced storage complexity hyperplane bounds is also minimal. Large gains in storage and computational costs were also observed (Figs. 10 and 11).

B. Data-Set: BIO-RETINA

Our data-set BIO-RETINA[3] consists of MPEG-7 texture feature descriptors extracted from 64×64 blocks generated from images of tissue sections of feline retinas as a part of an ongoing project at the Center for Bio-Image Informatics, UCSB. It is 208 506 elements long and 62-D. We also assumed a *page size* of 8 kB. The query sets themselves were generated by randomly selecting 100 elements from the relevant data-sets. For each query, the 10 nearest neighbors (10NN) were mined.

The weight matrix, typically a correlation matrix [32], was modeled as $W = U^T \Lambda U$. The orthonormal matrix U was generated randomly and the eigenvalues were uniformly distributed between 0 and 10. We present results from one such realization of W, that is representative of general performance.

We evaluated the performance of VA-File at various quantization levels (5–12 bits per dimension) and the VQ method for varying numbers of clusters (10–600 clusters). We note that our indexes are able to consistently reduce the number of random IO reads as compared with VA-File, when allowed (roughly)

[3]Download from http://scl.ece.ucsb.edu/datasets/BIORETINA_features.txt

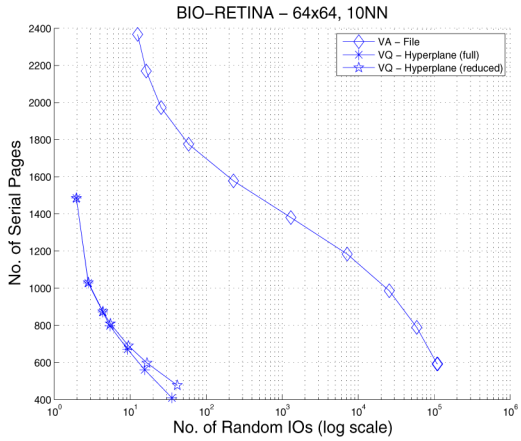

Fig. 12. IO performance.

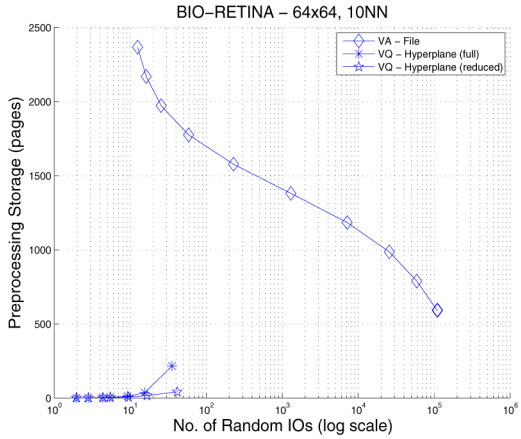

Fig. 13. Preprocessing storage.

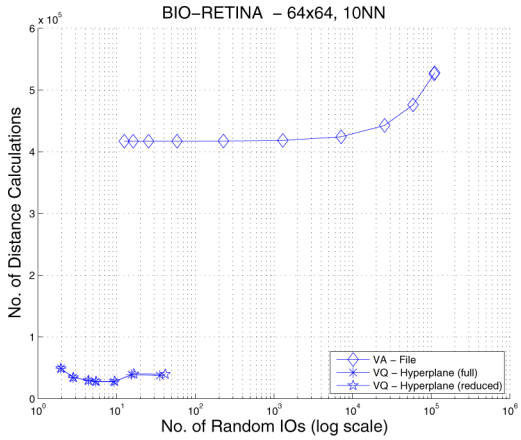

Fig. 14. Computational cost.

the same number of sequential disk accesses. For BIO-RETINA (Fig. 12), at 6 bit quantization for VA-File, a nearly $3000X$ reduction in costly random disk accesses is achieved by the vector quantization/clustering approach with 15 clusters. We note that at $K = 600$ clusters (the last/rightmost operating point), clustering was done in a multistage fashion, which explains the sudden "plateau" in storage/computation.

We also note that the VQ method has significantly ($\approx 10X - 100X$) lower storage and lower computational costs (Figs. 13 and 14).

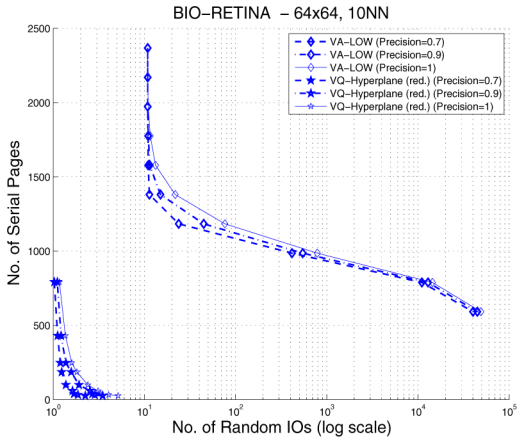

Fig. 15. IO performance under precision metric.

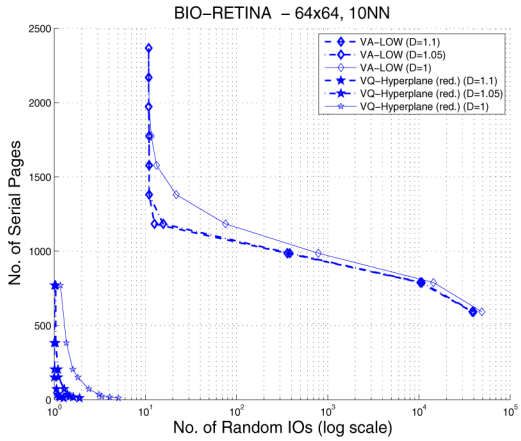

Fig. 16. IO performance under distance ratio (D) metric.

C. Approximate Nearest Neighbors: BIO-RETINA

Since feature vectors are approximations of the objects they represent, it could be argued that even exact search is unavoidably approximate. Since each disk IO retrieves a cluster, it would be sufficient to stop the search after the first few disk accesses to extract an approximate result, that could be further refined with user feedback. Since the results are not exact, a measure of quality is needed and typically *precision* or *recall* of the returned results is used. If $\mathcal{A}(\mathbf{q})$ and $\mathcal{G}(\mathbf{q})$ represent the approximate and golden (true) answer sets for query $\mathbf{q}$, we define

$$\text{Precision} = \frac{|\mathcal{A}(\mathbf{q}) \cap \mathcal{G}(\mathbf{q})|}{|\mathcal{A}(\mathbf{q})|}$$
$$\text{Recall} = \frac{|\mathcal{A}(\mathbf{q}) \cap \mathcal{G}(\mathbf{q})|}{|\mathcal{G}(\mathbf{q})|}.$$

For kNN queries, $|\mathcal{A}(\mathbf{q})| = |\mathcal{G}(\mathbf{q})|$ and, hence, precision equals recall.

It has also been argued that precision or recall are hard metrics that improperly measure the quality of results [25], [42] and that softer metrics such as the *distance ratio* metric proposed in [25], [26] would be more appropriate

$$D = \frac{\sum_{\mathbf{x} \in \mathcal{A}(\mathbf{q})} d(\mathbf{x}, \mathbf{q})}{\sum_{\mathbf{x} \in \mathcal{G}(\mathbf{q})} d(\mathbf{x}, \mathbf{q})}. \tag{9}$$

For brevity of presentation, we describe results for only the reduced complexity hyperplane bound with the BIO-RETINA data-set. Similar results are observed with the full complexity hyperplane bound and with the CORTINA-CalTech101 data-set.

1) Comparison With VA-LOW: Figs. 15 and 16 show the performance of our clustering + red. complexity hyperplane bound retrieval and VA-LOW [23], a variant of the VA-Files that can return approximate NNs. In the VA-LOW, the search in the second phase is stopped once sufficient vectors have been visited to assure a certain precision.

We first note that even the very first cluster returns high-precision results. However, in order to reduce the number of sequential IOs, it is necessary to allow 300–500 clusters, which results in a few additional random disk IOs. However, in VA-LOW several random and sequential disk access are necessary. Moreover, we observe that 100% precision results, i.e., the retrieval of *exact* nearest neighbors, is possible with just the first few disk IOs. However, in this approach, there is no guarantee that the exact NNs have been found, even though experimentally we notice 100% precision.

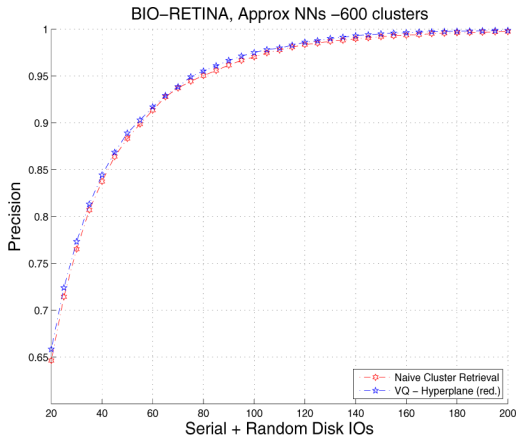

Fig. 17. Precision, 600 clusters.

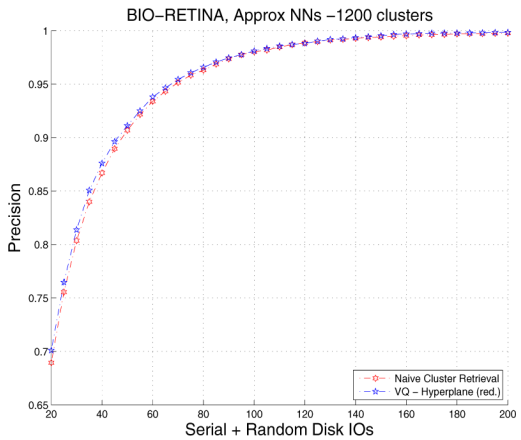

Fig. 18. Precision, 1200 clusters.

2) Comparison With Naive Cluster Retrieval: Next, we compare performance against naive cluster retrieval, i.e., retrieval

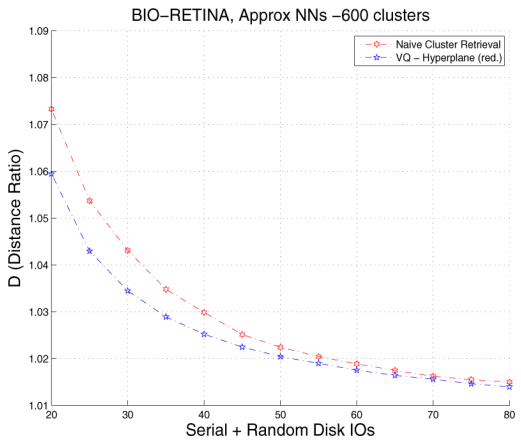

Fig. 19. Distance ratio (D) metric, 600 clusters.

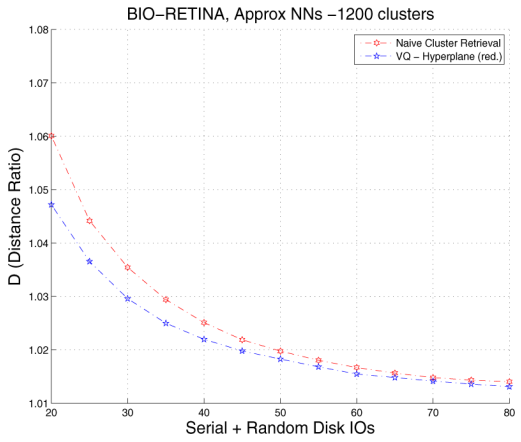

Fig. 20. Distance ratio (D) metric, 1200 clusters.

of clusters in order of distances to the centroids. The search is stopped at various stages and the precision (or distance ratio) is measured. This was performed at two levels of clustering with $K = 600$ clusters and with $K = 1200$ clusters.

We note that when the performance measure is precision, the retrieval performance of both schemes is very similar, with minor gains for hyperplane based cluster retrieval. However, if the performance measure is the distance ratio, we note that the naive cluster retrieval is worse with disk accesses increased by as much as 20%, when compared with hyperplane bound based retrieval for roughly same accuracy (D). This is because, with the hyperplane bound, better selection of clusters is possible.

VIII. Conclusion

The need to maintain large image repositories on secondary storage devices necessitates efficient high-dimensional indexes that are immune to the "curse of dimensionality." There is also the need to adapt to users' perceptions of similarity, which could be significantly different from that induced by the standard Euclidean norm. The Mahalanobis distance metric has more degrees of freedom and with proper relevance feedback can adapt to user perceptions. However, most indexing schemes typically require that the Mahalanobis weight matrix be known beforehand and are typically rendered useless if the weight matrix changes after index creation. We proposed a clustering-based indexing technique, where relevant clusters are retrieved till the exact nearest neighbors are found. Central to our indexing scheme is a cluster distance estimation technique that provides tight lower bounds on query-cluster distances, while adapting to changes in the distance metric. This enabled efficient cluster pruning with low preprocessing storage and computation costs. The IO access times of our index are significantly lower than variants of VA-Files adapted to relevance feedback or naive cluster retrieval and, thus, enabling effective application of relevance feedback techniques.

Acknowledgment

The authors would like to thank Dr. M. Verardo, Prof. S. Fisher, and Dr. G. Lewis, from the Neuroscience Research Institute, UCSB, for generously providing the image data; Dr. J. Biyun and R. Kumar, of the Center for Bio-Image Informatics, UCSB, who extracted the features for the BIO-RETINA data-set; and J. Xu of the Vision Research Lab, UCSB, who extracted the features for the CORTINA data-set.

References

[1] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system," *Computer*, vol. 28, no. 9, pp. 23–32, 1995.

[2] T. Sikora, "The MPEG-7 visual standard for content description-anoverview," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 6, pp. 696–702, Jun. 2001.

[3] D. Lowe, "Object recognition from local scale-invariant features," *ICCV*, vol. 2, pp. 1150–1157, 1999.

[4] M. Bober, "MPEG-7 visual shape descriptors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 6, pp. 716–719, Jun. 2001.

[5] G. Mori, S. Belongie, and J. Malik, "Efficient shape matching using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 11, pp. 1832–1837, Nov. 2005.

[6] B. Manjunath, J.-R. Ohm, V. Vasudevan, and A. Yamada, "Color and texture descriptors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 6, pp. 703–715, Jun. 2001.

[7] M. Swain and D. Ballard, "Indexing via color histograms," *ICCV*, pp. 390–393, 1990.

[8] A. Barla, F. Odone, and A. Verri, "Histogram intersection kernel forimage classification," in *Proc. Int. Conf. Image Processing*, 2003, vol. 3, pp. III-513–III-516.

[9] K. Grauman and T. Darrell, "The Pyramid Match kernel: Discriminativeclassification with sets of image features," *ICCV*, vol. 2, 2005.

[10] Y. Rubner, C. Tomasi, and L. Guibas, "The Earth mover's distance asa metric for image retrieval," *Int. J. Comput. Vis.*, vol. 40, no. 2, pp. 99–121, 2000.

[11] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.

[12] R. Ramakrishnan and J. Gerhke, *Database Management Systems*, 2nd ed. New York: McGraw-Hill, 2001.

[13] A. Guttman, "R-trees: A dynamic index structure for spatial searching," *SIGMOD*, pp. 47–57, 1984.

[14] D. A. White and R. Jain, "Similarity indexing with the SS-tree," *ICDE*, pp. 516–523, 1996.

[15] R. Weber, H. Schek, and S. Blott, "A quantitative analysis and performancestudy for similarity-search methods in high-dimensional spaces," *VLDB*, pp. 194–205, Aug. 1998.

[16] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton, NJ: Princeton Univ. Press, 1961.

[17] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. E. Abbadi, "Vectorapproximation based indexing for non-uniform high dimensional datasets," *CIKM*, pp. 202–209, 2000.

[18] J. Chen, C. Bouman, and J. Dalton, "Hierarchical browsing and searchof large image databases," *IEEE Trans. Image Process.*, vol. 9, no. 3, pp. 442–455, Mar. 2000.

[19] J. Chen, C. Bouman, and J. Allebach, "Fast image database search usingtree-structured VQ," in *Proc. Int. Conf. Image Processing*, 1997, vol. 2, pp. 827–8305.
[20] S. Ramaswamy and K. Rose, "Adaptive cluster-distance bounding forsimilarity search in image databases," in *Proc. Int. Conf. Image Processing*, 2007, vol. 6, pp. 381–384.
[21] C. Yu, B. C. Ooi, K. L. Tan, and H. V. Jagadish, "Indexing the distance: An efficient method to knn processing," *VLDB*, pp. 421–430, Sept. 2001.
[22] N. Koudas, B. C. Ooi, H. T. Shen, and A. K. H. Tung, "LDC: Enablingsearch by partial distance in a hyper-dimensional space," *ICDE*, pp. 6–17, 2004.
[23] R. Weber and K. Böhm, "Trading quality for time with nearest neighborsearch," *EDBT*, pp. 21–35, 2000.
[24] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in highdimensions via hashing," *VLDB*, pp. 518–529, Sep. 1999.
[25] E. Tuncel, H. Ferhatosmanoglu, and K. Rose, "VQ-Index: An index structure for similarity searching in multimedia databases," *ACM Multimedia*, pp. 543–552, 2002.
[26] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. E. Abbadi, "Approximatenearest neighbor searching in multimedia databases," *ICDE*, pp. 503–511, Apr. 2001.
[27] T. Huang and X. S. Zhou, "Image retrieval with relevance feedback: From heuristic weight adjustment to optimal learning methods," in *Proc. Int. Conf. Image Processing*, 2001, vol. 3, pp. 2–5.
[28] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon, "Information-theoreticmetric learning," *ICML*, pp. 209–216, 2007.
[29] E. Xing, A. Ng, M. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," *Adv. Neural Inf. Process. Syst.*, vol. 15, pp. 505–512, 2003.
[30] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall, "Learning amahalanobis metric from equivalence constraints," *J. Mach. Learn. Res.*, vol. 6, pp. 937–965, 2005.
[31] Y. Chen, X. S. Zhou, and T. Huang, "One-class SVM for learning in image retrieval," in *Proc. Int. Conf. Image Processing*, 2001, vol. 1, pp. 34–37.
[32] Y. Ishikawa, R. Subramanya, and C. Faloutsos, "Mindreader: Queryingdatabases through multiple examples," *VLDB*, pp. 218–227, Aug. 1998.
[33] Y. Rui and T. Huang, "Optimizing learning in image retrieval," *CVPR*, vol. 1, pp. 1236–1243, 2000.
[34] Y. Rui, T. Huang, and S. Mehrotra, "Content-based image retrieval withrelevance feedback in MARS," in *Proc. Int. Conf. Image Processing*, 1997, vol. 2, pp. 815–818.
[35] P. Jain, B. Kulis, and K. Grauman, "Fast image search for learnedmetrics," *CVPR*, pp. 1–8, 2008.
[36] Y. Sakurai, M. Yoshikawa, R. Kataoka, and S. Uemura, "Similaritysearch for adaptive ellipsoid queries using spatial transformation," *VLDB*, pp. 231–240, 2001.
[37] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer, 1992.
[38] S. Ramaswamy and K. Rose, "Fast adaptive Mahalanobis-distance basedsearch and retrieval in image databases," in *Proc. Int. Conf. Image Processing*, 2008, pp. 181–184.
[39] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1996.
[40] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient dataclustering method for very large databases," *SIGMOD*, pp. 103–114, 1996.
[41] CalTech 101 Data-Set [Online]. Available: www.vision.caltech.edu/Image Datasets/Caltech101/
[42] E. Tuncel and K. Rose, "Towards optimal clustering for approximatesimilarity searching," *ICME*, vol. 2, pp. 497–500, Aug. 2002.

Sharadh Ramaswamy received the B.Tech. and M.Tech. degrees in 2003 from the Indian Institute of Technology, Madras, and the Ph.D. degree in 2009 from the University of California, Santa Barbara.

He then joined MayaChitra, Inc., as a member of Research Staff. His research interests lie in compression, sensor networks, search and retrieval in databases, video processing, and machine learning.

Kenneth Rose (S'85–M'91–SM'01–F'03) received the Ph.D. degree in 1991 from the California Institute of Technology, Pasadena.

He joined the Department of Electrical and Computer Engineering, University of California, Santa Barbara, where he is currently a Professor. His main research activities are in the areas of information theory and signal processing and include rate-distortion theory, source and source-channel coding, audio and video coding and networking, pattern recognition, search and retrieval in databases, and nonconvex optimization. He is interested in the relations between information theory, estimation theory, and statistical physics, and their potential impact on fundamental and practical problems in diverse disciplines.

Dr. Rose was a corecipient of the 1990 William R. Bennett Prize Paper Award of the IEEE Communications Society, as well as the 2004 and 2007 IEEE Signal Processing Society Best Paper Awards.