Adaptive Nearest Neighbor Search for Relevance Feedback in Large Image Databases

P. Wu and B.S. Manjunath
Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106-9560
{peng, manj}@ece.ucsb.edu

Abstract

Relevance feedback is often used in refining similarity retrievals in image and video databases. Typically this involves modification to the similarity metrics based on the user feedback and recomputing a set of nearest neighbors using the modified similarity values. Such nearest neighbor computations are expensive given that typical image features, such as color and texture, are represented in high dimensional spaces. Search complexity is a critical issue while dealing with large databases and this issue has not received much attention in relevance feedback research. Most of the current methods report results on very small data sets, of the order of few thousand items, where a sequential (and hence exhaustive search) is practical. The main contribution of this paper is a novel algorithm for adaptive nearest neighbor computations for high dimensional feature vectors and when the number of items in the database is large. The proposed method exploits the correlations between two consecutive nearest neighbor searches when the underlying similarity metric is changing, and filters out a significant number of candidates in a two stage search and retrieval process, thus reducing the number of I/O accesses to the database. Detailed experimental results are provided using a set of about 700,000 images. Comparison to the existing method shows an order of magnitude overall improvement.

Keywords

relevance feedback, nearest neighbor search, similarity retrieval

1 Introduction

In recent years, there has been considerable work on the use of low level visual features for content based image and video retrieval. Much of this work focus on feature extraction and on combining multiple image features, such as color and texture, for effective retrieval. Classic examples are the IBM's QBIC [13] and the Virage image search engine [1]. While the low level features are quite effective in "similarity" retrieval, the similarity is not necessarily at a semantic level. This has been generally acknowledged in the image retrieval literature.

As a step towards semantic retrieval, many researchers have proposed the use of relevance feedback to improve the retrieval effectiveness. In relevance feedback, the user is given an opportunity to provide feedback to the system regarding the set of retrievals computed. This feedback is then used to compute a next set of retrievals that better match the user's expectations [9, 10, 14, 15, 16]. Typically, this learning involves either modifying the feature space and/or the similarity metrics used in computing the distances between the query and the database items. For example, in [10, 14] the database objects are adaptively clustered to match the user's responses. In [9, 15, 16], the weights associated with the various feature vector components are modified to achieve a similar objective.

Relevance feedback is an iterative process wherein the retrievals are updated at each stage based on the user's feedback. The updated set of retrievals are supposed to be more similar to the query image. Assuming that one can modify the feature space and the distance metrics in an efficient way to provide a better set of retrieval examples, one still needs to recompute a set of nearest neighbors for the given query vector iteratively. This nearest neighbor computation is needed during each update. When large datasets of image/video objects are involved, this iterative process of finding nearest neighbors is an expensive computation, and can be a limiting factor on the overall effectiveness of using relevance feedback for similarity retrieval.

One important factor contributing to this computational complexity is that nearest neighbor search in a high dimensional feature space is expensive when the search is over a large number of objects. Typically, the dimensionality of image/video descriptors tends to be in the range of few tens to few hundreds. Similarity search in such high-dimensions is an active research area. Some well-known indexing schemes are presented in [2, 3, 4, 5, 7, 8, 18, 19], but a recent study of these index structures demonstrates that their indexing performance degrades to a linear search of the whole database when the dimensionality is high [18]. For smaller databases (containing a few thousand items), a linear search is a feasible option. In fact, most of the current work on relevance feedback do not address this indexing issue and assume an exhaustive search of the database during each iteration. Such computations can become prohibitive as the database size increases.

The changing nature of the underlying feature space and the similarity metrics in relevance feedback further add to the complexity of this nearest neighbor search. For example, the distance between two feature vectors is typically calculated as a weighted Euclidean distance and the weights are updated based on the rele-

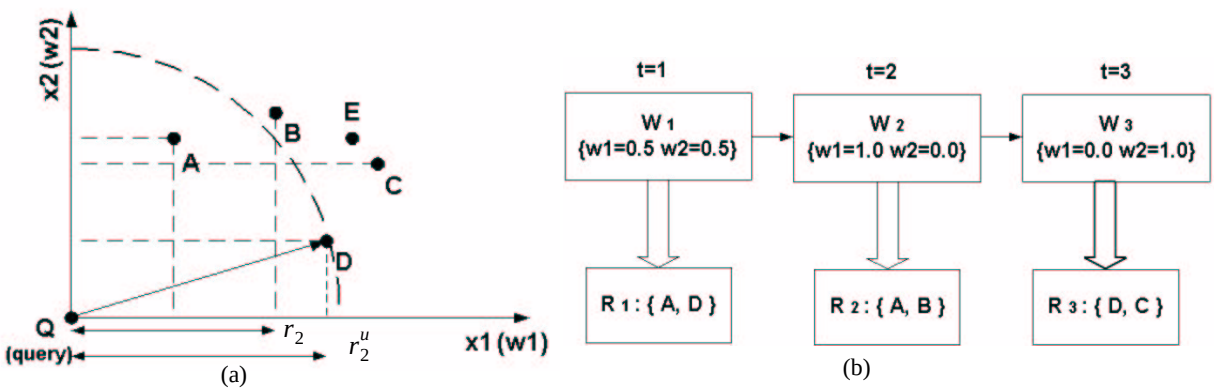


Fig 1. Weight updates during the relevance feedback. (a) A, B, C, D and E are feature vectors and Q is the query; (b) The two nearest neighbors to Q during each iteration.

vance feedback. Given the updated weight matrix, the next set of nearest neighbors is then computed. The focus of this paper is on efficient computation of these nearest neighbors when the underlying distance metric is changing during each iteration. To the best of our knowledge, this issue has not been addressed in the database research.

The paper is organized as follows: Section 2 provides a brief overview of relevance feedback and in Section 3 we discuss an approximation based indexing structure for high dimensional feature spaces. In Section 4 we propose a modification to the filtering process in the approximation based search and indexing to facilitate iterative nearest neighbor retrievals. Experimental results are provided in Section 5 and we conclude with a brief discussion in Section 6.

2 Relevance feedback

Low level descriptors often fail to capture the underlying semantics of the data. Relevance feedback has attracted considerable attention in the context of computing a better set of retrievals for a given query. For example, given a set of retrievals for a query, the user may identify some relevant examples and some non-relevant examples. Based on the user's feedback so collected, the similarity metric is modified to recompute the next set of retrievals. The hope is that this modification to the similarity metric would help provide better matches to a given query and meet the user's expectations.

Consider a database of N, M-dimensional feature vectors $F_i = [f_{i1}, f_{i2}, \ldots, f_{iM}]$, $i = 1, \ldots, N$. For example, these feature vectors may correspond to the color histograms of images in the database. Let $Q = [q_1, q_2, \ldots, q_M]$ be the query object. A popular dis-similarity metric - the weighted Euclidian distance - is used for retrieving the nearest neighbors. The distance between a database object F_i and the query Q is computed as:

$$d(Q, F_i, W) = (Q - F_i)W(Q - F_i)^T \qquad (1)$$

where W is a real symmetric matrix [9, 14, 15, 16]. In our following discussion, we assume W to be a diagonal matrix, and the results can be extended to a more general symmetric matrix.

The weight matrix W is updated iteratively in relevance feedback. Let t be the iteration index, $t = 1, 2, \ldots, T$. T is the total number of iterations. Let W_t be the weight matrix and R_t be the set of K nearest neighbors for the query Q during iteration t. The weight matrix is initialized by assigning the value $1/M$ to each of the diagonal elements. User's feedback is in the form of identifying those objects in R_t that are relevant to the query object. These objects are considered as positive examples. These positive examples are used to modify the weight matrix as follows [16],

$$w_{jj}^{t+1} = \frac{1}{\sigma_j} \qquad (2)$$

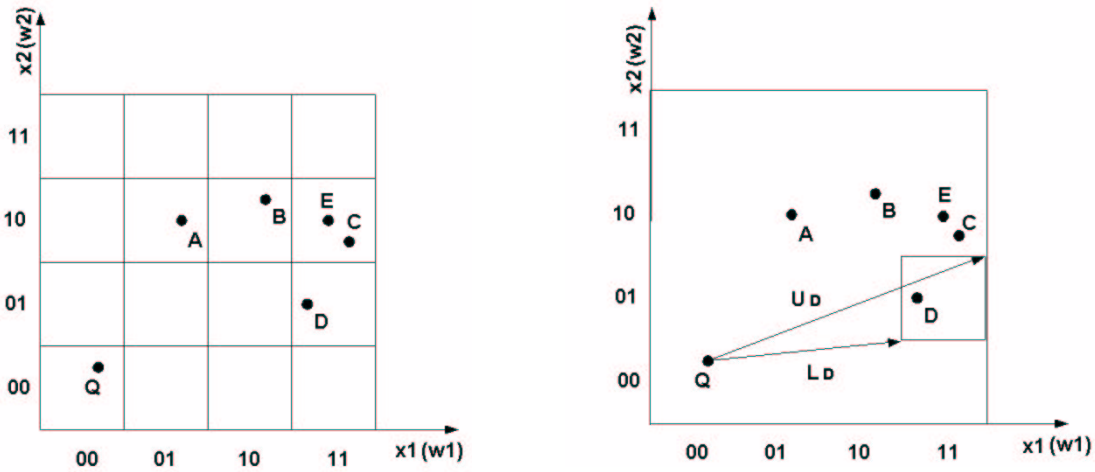
where σ_j is the standard deviation of f_{ij} from all the positive examples. These weights are further normalized.

$$w_{jj}^{t+1} = \frac{w_{jj}^{t+1}}{\sum_{j = 1, \ldots, M} w_{jj}^{t+1}} \qquad (3)$$

This modified weight matrix W_{t+1} will be used in iteration $t+1$ to recomputed the next set of nearest neighbors.

This iterative process is illustrated in Figure 1. Consider the case when $N = 4$, $M = 2$, $K = 2$ and $T = 3$. Note that the query Q is located at the origin of the coordinates in Figure 1(a). The two weights associated with the two axes x_1 and x_2 are denoted as w_1 and w_2, respectively, which correspond to the diagonal elements of the weight matrix. $R_1 \rightarrow R_2 \rightarrow R_3$ correspond to the changes in the nearest neighbor set as the weight matrix is updated from $W_1 \rightarrow W_2 \rightarrow W_3$.

During the relevance feedback, we face the task of finding K nearest neighbors R_t for a query under a weight matrix W_t at each iteration. As discussed in the introduction, the iterative process is very expensive if we consider each iteration of K nearest



(a) (b)

Fig 2. (a) Construction of the approximations; (b) Computation of the lower bound and the upper bound.

neighbor search as independent. In this paper, we focus on the issue of how to find R_{t+1} more efficiently given R_t, W_t and W_{t+1}.

3 An approximation based index scheme to support relevance feedback

Typical audio-visual descriptors are high dimensional vectors [1, 11]. Their dimensionality range from few tens to a few hundreds. To index such high dimensional feature vectors, various index structures have been proposed [2, 3, 4, 7, 8, 18, 19]. One can broadly classify them into feature based and distance based methods, as discussed in [4]. In feature based methods, the feature values in each of the feature dimensions are used to partition the space. This partitioning is independent of the distance metric used for comparing objects. In distance based methods, the distance of the objects to a set of pivot points is used to guide the partitioning.

In image/video retrieval, feature based methods [2, 3, 18] are preferred over the distance based methods [19] due to the variations in the similarity metrics and the need for interactivity and user feedback. Further, it is argued in [18] that typical index methods are outperformed by a linear search when the search dimensions exceed 10. This has motivated the introduction of approximation methods [7, 18] to speed up the linear search. Approximation based methods have certain advantages. First, they support different distance measures. This is a very important property especially for learning and concept mining related applications. Secondly, the construction of approximation can be made adaptive to the dimensionality of data. A brief description of the construction of such an approximation is given below.

3.1 The construction of approximations

Given a high dimensional feature space, the corresponding approximation based index structure is constructed as follows.

First, each of the feature vector dimensions is partitioned into a certain number of non overlapping segments. Typically, the number of segments is assigned to be 2^{B_j}, $j = 1, \ldots, M$, where B_j is the number of bits allocated to dimension j. Denote the boundary points that determine the 2^{B_j} partitions to be $b_{l,j}$, $l = 0, 1, \ldots, 2^{B_j} - 1$. A given partition $G^j(l)$ is determined by $[b_{l,j}, b_{l+1,j}]$. Thus, each high dimensional cell is represented by a string of bits of length B ($B = \sum_{j=1,\ldots,M} B_j$). For a feature vector F_i, its approximation $P(F_i)$ is an index to the cell containing F_i. Consider Figure 2(a), where $B_1 = B_2 = 2$. The approximation for $P(A)$ is "0110", where "01" and "10" are the indices on dimension x_1 and x_2 respectively. Note that each approximation cell contains a number of feature vectors. For example, in Figure 2(a), $P(C)$ and $P(E)$ have the same approximation "1110".

Note that if F_i is in partition l along the jth dimension, then

$$b_{l,j} \le f_{ij} \le b_{l+1,j} \tag{4}$$

Given a query feature vector Q and a weight matrix W_t, it is easy to verify that [18]

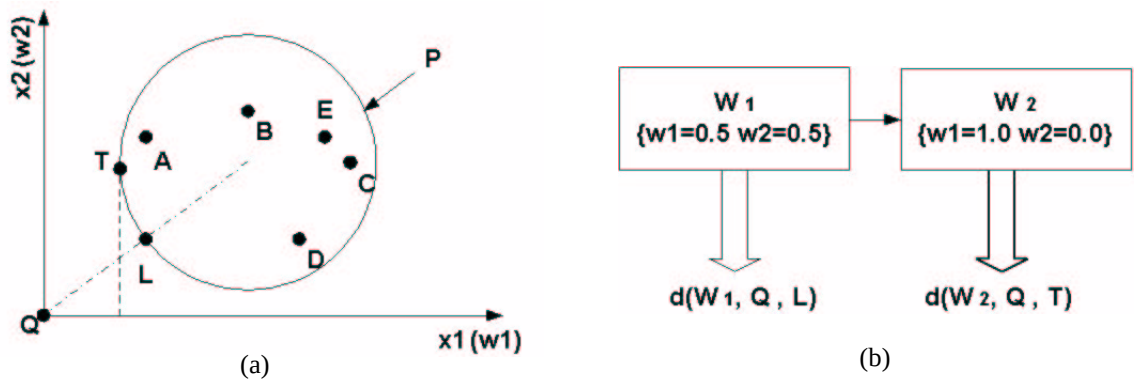
$$L_i(Q, W_t) \le d(Q, F_i, W_t) \le U_i(Q, W_t) \tag{5}$$

where the lower bound $L_i(Q, W_t)$ and upper bound $U_i(Q, W_t)$ are computed as

$$L_i(Q, W_t) = [l_{i1}\ \ l_{i2}\ \ \ldots l_{iM}] W_t [l_{i1}\ \ l_{i2}\ \ \ldots l_{iM}]^T \tag{6}$$

$$U_i(Q, W_t) = [u_{i1}\ \ u_{i2}\ \ \ldots u_{iM}] W_t [u_{i1}\ \ u_{i2}\ \ \ldots u_{iM}]^T \tag{7}$$

where



(a) (b)

Fig 3. (a) Using hyper-sphere to bound feature vectors. For illustrative purpose, we assume feature vectors A, B, C, D and E are bounded by sphere P. (b) The lowers bound computed from different weight matrixes when using hyper-sphere as the approximation.

$$l_{ij} = \begin{cases} q_j - b_{l+1,j} & q_j > b_{l+1,j} \\ 0 & q_j \in [b_{l,j}, b_{l+1,j}] \\ b_{l,j} - q_j & q_j < b_{l,j} \end{cases} \quad (8)$$

$$u_{ij} = \begin{cases} q_j - b_{l,j} & q_j > b_{l+1,j} \\ max(q_j - b_{l,j}, b_{l+1,j} - q_j) & q_j \in [b_{l,j}, b_{l+1,j}] \\ b_{l+1,j} - q_j & q_j < b_{l,j} \end{cases}$$

Figure 2(b) illustrates these bounds for the 2-D examples in Figure 2(a).

In equation (6) and (7), $L_i(Q, W_t)$ and $U_i(Q, W_t)$ vary with weight matrix W_t. The computations of these bounds can become more involved if approximations other than hyper rectangles are used. In Figure 3, we illustrate this observation by using a hyper-sphere to bound the feature vectors [5]. When the weight matrix is updated from W_1 to W_2, l_{ij} also changes.

3.2 K nearest neighbor (NN) search in approximation based index structures

Approximation based nearest neighbor search can be considered as a two phase filtering process [18].

1. **Phase I** - Approximation level filtering: In this phase, the set of all vector approximations is scanned sequentially and lower and upper bounds on the distances of each object in the database to the query object are computed. During the scan, a buffer is used to keep track of the K-th largest upper bound found from the scanned approximations. If an approximation is encountered such that its lower bound is larger than the K-th largest upper bound found so far, the corresponding feature vector can be skipped since at least K better candidates exist. Otherwise, the approximation will be selected as a candidate and its upper bound will be used to update the buffer. Denote the set of candidate objects at this stage to be $N_1(Q, W_t)$ and the number of elements in the set to be $|N_1(Q, W_t)|$.

2. **Phase II** - Data level filtering: In the second phase filtering, the actual $|N_1(Q, W_t)|$ feature vectors are examined. The feature vectors are visited in increasing order of their lower bounds and the exact distances to the query vector are computed using equation (1). If a lower bound is reached that is larger than the K-th actual nearest neighbor distance encountered so far, there is no need to visit the remaining candidates. Let $N_2(Q, W_t)$ denote the set of objects visited before the lower bound threshold is encountered. Finally, the K nearest neighbors are found by sorting the $|N_2(Q, W_t)|$ distances.

The objective of Phase I filtering is to find a set of approximations so that K nearest neighbors can be found from the feature vectors contained by those approximations. Note that the efficiency of the two phase filtering process is mainly determined by the effectiveness of Phase I filtering. The reason is that the number of candidates from Phase I filtering determines the cost of disk access or page access. The disk/page access is considered the most expensive process in large databases. The smaller this candidate set is, the better is the indexing performance. For this reason, our main focus is on how the Phase I filtering can be improved in the relevance feedback context.

4 Adaptive Nearest Neighbor Search for Relevance Feedback

In investigating how to adapt the Phase I filtering within the context of relevance feedback, we begin by exploring the correlation between R_t and R_{t+1}, i.e., the K nearest neighbors in two consecutive iterations.

At iteration t, R_t contains the K nearest neighbors of query Q computed using the weight matrix W_t. Let $r_t(Q)$ denote the distance between Q and the K-th farthest object. Our objective

is to establish an upper bound on $r_{t+1}(Q)$ when W_t is updated to W_{t+1}. Denote this upper bound as $r^u_{t+1}(Q)$.

Theorem 1:(upper bound of $r_t(Q)$)

Let $R_t = \{F_i^t, i = 1, \ldots, K\}$ be the set of nearest neighbors of query Q at iteration t. Let

$$r^u_{t+1}(Q) = max\{d(Q, F_i^t, W_{t+1}), i = 1, \ldots, K\} \quad (9)$$

Then

$$r_{t+1}(Q) \le r^u_{t+1}(Q) \quad (10)$$

proof: see [20].

Consider Figure 1, we have $R_1 = \{A, D\}$. When W_1 is updated to W_2, $r^u_2(Q) = d(Q, D, W_2)$. The theorem states that the maximum of the distances between the query and objects in R_2 computed using W_2, $r_2(Q) = d(Q, B, W_2)$, can not be larger than $r^u_2(Q)$.

The purpose of Phase I filtering discussed in Section 3.2 is to determine a subset of approximations from which the K nearest neighbors can be retrieved. Let $N_1^{opt}(Q, W_t)$ denote the minimal set of approximations that contain all the K nearest neighbors. The best case scenario for Phase I filtering is to identify exactly this subset $N_1^{opt}(Q, W_t)$. In the following, we will introduce a set of necessary conditions for an approximation to be a qualified one in $N_1^{opt}(Q, W_t)$. If any of the necessary conditions can not be satisfied, the approximation will be safely excluded from the Phase I filtering.

Necessary condition I: *Denote the K-th largest upper bound among all the approximations $P(F_i)$, $i = 1, \ldots, N$, to be $\gamma(Q, W_t)$. If a hyper rectangle $P(F_i)$ belongs to $N_1^{opt}(Q, W_t)$, then $L_i(Q, W_t) < \gamma(Q, W_t)$.*

When an approximation $P(F_i)$ does not satisfy this condition, it can be excluded. In the standard approaches [3, 7, 18], the approximations are scanned sequentially. Only the K-th largest upper bound from the scanned approximations is available and used for filtering. Let ϕ be K-th largest upper bound found so far during a scan. Note that this is not necessarily the same as $\gamma(Q, W_t)$. An approximation will be chosen as a candidate if its lower bound is smaller than ϕ, and its upper bound will be used to update the buffer and the value of ϕ. This filtering process may introduce some false candidates, i.e., candidates with lower bound larger than $\gamma(Q, W_t)$. The number of such false candidates resulting from Phase I filtering depend on how soon the value of ϕ returned from the buffer can converge to $\gamma(Q, W_t)$. The following theorem establishes a bound on $\gamma(Q, W_t)$.

Theorem 2:(upper bound on $\gamma(Q, W_t)$)

For the $|N_1(Q, W_t)|$ candidates resulting from iteration t, denote the K-th largest upper bound of them under W_{t+1} to be $\theta_{t+1}(Q)$, then

$$\gamma(Q, W_{t+1}) < \theta_{t+1}(Q) \quad (11)$$

proof: see [20].

The above observation leads us to the following necessary condition for an approximation $P(F_i)$ to be chosen as a candidate of $N_1^{opt}(Q, W_t)$.

Necessary Condition II: For an approximation $P(F_i)$ to be chosen as a candidate of $N_1^{opt}(Q, W_{t+1})$, its lower bound must satisfy the inequality

$$L_i(Q, W_{t+1}) < \theta_{t+1}(Q) \quad (12)$$

Unlike the first condition, the bound $\theta_{t+1}(Q)$ can be computed in advance of Phase I filtering at iteration $t+1$. When the inequality (12) is not satisfied, we can skip the corresponding $P(F_i)$ and do not need to update the buffer even if $L_i(Q, W_{t+1})$ is smaller than ϕ.

Based on theorem 1, another necessary condition can be derived for $P(F_i)$ to be a qualified one in $N_1^{opt}(Q, W_t)$. At iteration t, $L_i(Q, W_t)$ and $U_i(Q, W_t)$ are computed according to equation (6) and (7) for $P(F_i)$. Given r^u_{t+1} from equation (9), we have the following condition.

Necessary Condition III: *For approximation $P(F_i)$ to be a qualified one in $N_1^{opt}(Q, W_{t+1})$, its lower bound $L_i(Q, W_{t+1})$ must satisfy*

$$L_i(Q, W_{t+1}) < r^u_{t+1}(Q) \quad (13)$$

Obviously, when $r^u_{t+1}(Q) < \gamma(Q, W_{t+1})$, fewer candidates need to be examined in Phase I filtering (see the proposed algorithm below). A simple example is illustrated in Figure 4. In Figure 4, when $K = 1$ and weight matrix W_1 is updated to W_2, $\gamma(Q, W_2)$ at iteration $t = 2$ is determined by the distance between Q and the lower right vertex of P_A, $r^u_2(Q) = d(Q, A, W_2)$. The inequality $r^u_2(Q) < \gamma(Q, W_2)$ enables us to search only in the rectangle where A stays.

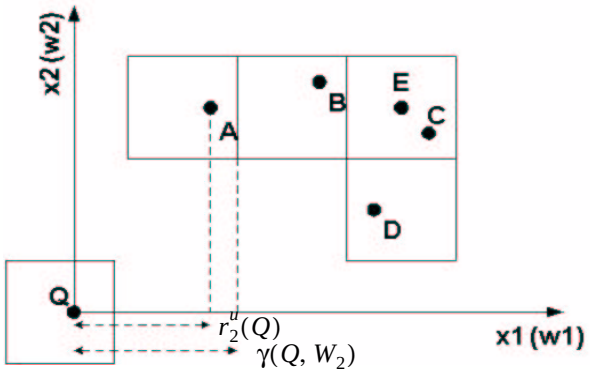


Fig 4. Illustration of using $r_{t+1}^u(Q)$ to exclude the approximations that need not to be searched. See Necessary Condition III in the text for more details.

The above conditions on the lower bounds lead us to the following proposed modifications to the standard Phase I filtering approach. In the following, we refer to the method described in Section 3.2 as the standard approach, and our modifications to it as given below, as the proposed approach.

4.1 An adaptive K-NN search algorithm

The following steps describe our proposed changes to the phase I filtering. Note that a buffer is used to keep track of the value of ϕ, the K-th largest upper bound found so far during a scan. In practice, this buffer can be implemented using a heap-based priority queue [6].

1. If $t = 1$. use the standard approach to Phase I filtering (Section 3.2) to find the candidates and the K-nearest neighbors to a query; else, go to Step 2;

2. Given W_t, R_{t-1} and $N_1(Q, W_{t-1})$, compute $r_t^u(Q)$ and $\theta_t(Q)$;

3. Insert $U_i(Q, W_t)$, $i = 1, \ldots, K$, into a buffer of size K and return ϕ. Let $N_1(Q, W_t) = \{P(F_i), i = 1, \ldots, K\}$ and $i = K$;

4. $i = i + 1$;

5. compute $L_i(Q, W_t)$ and $U_i(Q, W_t)$ of $P(F_i)$;

6. if $L_i(Q, W_t) \le min(r_t^u(Q), \theta_t(Q))$ and $L_i(Q, W_t) < \phi(Q, W_t)$, insert $U_i(Q, W_t)$ into the buffer and return ϕ, insert $P(F_i)$ into $N_1(Q, W_t)$ and $|N_1(Q, W_t)| = |N_1(Q, W_t)| + 1$; otherwise, go to next step;

7. if $i \le (N - 1)$, go to step 4; otherwise, continue;

8. return $N_1(Q, W_t)$ and $|N_1(Q, W_t)|$, end;

It is not difficult to notice that the main modification of the proposed approaches as compared with the standard approach is highlighted by steps 5 and 6. The essential difference between the two approaches is that in the proposed approach more constraints are enforced in filtering approximations. Because these constraints are necessary conditions for an approximation to be one in $N_1^{opt}(Q, W_t)$, more approximations can be excluded. Thus this proposed method results in a smaller set of $N_1(Q, W_t)$ compared with that of the standard approach.

5 Experiments

In this section, we demonstrate the effectiveness of the proposed approach over a large image database.

5.1 Overview of the experiment

Dataset: A total of 685, 900 images collected from the internet (under the "shopping" category) are used in our experiment.

Feature space: Color feature is used to annotate the images. The color feature descriptors are computed in the LUV color space [12]. Each color component is quantized into 4 bins. This results in a $M = 64$ dimensional color feature for each of the image objects. The value on each dimension is uniformly quantized into the range 0 to 255.

Relevance feedback: Since color is the feature describing the image data, the users are instructed to provide the feedback based on their perceived color similarity. To facilitate gathering the feedback information, an user interface was developed For a specific query, the user will select a certain number of relevant retrievals to update the distance measure. The distance metric is updated during each iteration as described in Section 2.

Index structure: The approximations are constructed using the method described in Section 3.1. In the implementation, we assign equal number of bits to all dimensions ($B_j = constant$, $j = 1, \ldots, M$). Each of the feature dimensions is uniformly partitioned. Let S denote length of these partitions. Experiments are carried out for four different values of S, $S = 4, 8, 16, 32$. A smaller value S corresponds to the approximation constructed at a finer resolution. We compare the standard approach to computing the K-NN (Section 3.2) to our proposed method (Section

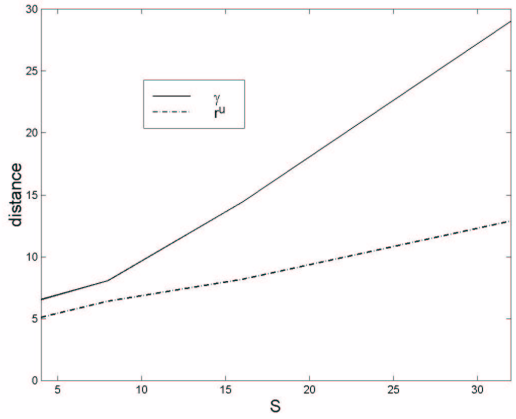


Fig 5. Overall measurements r_t^u and γ computed from approximations at different resolutions.

4.1) for these different resolutions of constructing the approximations.

Queries: 50 queries Q_m, $m = 1, \ldots, 50$, are selected randomly from the dataset. These queries are assigned to a group of L ($L = 3$) users to perform the relevance feedback. For each query, $K = 20$ nearest neighbors are retrieved during each iteration. As mentioned earlier, the feedback from the users is based on color relevance only. For each query Q_m, the user is asked to perform 6 iterations of relevance feedback ($T = 6$).

Measurements: To evaluate the effectiveness of the proposed indexing scheme, we are interested in the following issues:

1. How often the inequality $r_t^u(Q) < \gamma(Q, W_t)$ holds in practice? As discussed in Section 4, this may lead to a significant savings in $|N_1(Q, W_t)|$.

2. How close $\theta_t(Q)$ is to $\gamma(Q, W_t)$? The closer they are, the more is the number of approximations that can be excluded during the first phase of filtering.

3. The reduction in $|N_1(Q, W_t)|$ using the proposed approach as compared with that of the standard approach.

So, each time the K nearest neighbor search is performed using the proposed approach, we record the measurements $r_t^u(Q)$, $\theta_t(Q)$, $\gamma(Q, W_t)$ and $|N_1(Q, W_t)|$. For the standard approach, $|N_1(Q, W_t)|$ is recorded at each iteration.

In order to reduce the subjectivity associated with individual user feedbacks, we use the average value, averaged over all the users for any given measurement. For instance, the average value of $\theta_t(Q_m)$ for query Q_m is computed as

$$\theta_t(Q_m) = \frac{1}{L(T-1)} \sum_{l=1}^{L} \sum_{t=2}^{T} \theta_t^{(l)}(Q_m) \qquad (14)$$

where $\theta_t^{(l)}(Q_m)$ represents $\theta_t(Q_m)$ from user l, $l = 1, \ldots, L$. Note that we do not count the first round of search, since the first iteration of nearest neighbor search is the same for both of the standard and the proposed approaches.

5.2 Results

5.2.1 Upper bound $r_t^u(Q)$ vs. $\gamma(Q, W_t)$

As discussed in Section 4, when $r_t^u(Q) < \gamma(Q, W_t)$, it is guaranteed that $|N_1(Q, W_t)|$ of the proposed approach is smaller than that of the standard approach. So we first evaluate how often this inequality holds in experiments. Using the approximations constructed at a certain resolution S, the average values on $r_t^u(Q_m)$ and $\gamma(Q_m, W_t)$ are computed for each of the 50 queries. For overall performance evaluation, these are further averaged over all the queries Q_m, resulting in two values r_t^u and γ. In Figure 5, two curves are plotted. The solid one corresponds to value γ for all the four resolutions $S = 4, 8, 16, 32$, and the dashed curve represents value r_t^u for all four resolutions.

From Figure 5, we observe that the inequality $r_t^u(Q) < \gamma(Q, W_t)$ holds in general. As such, a better performance can be expected using the proposed method. Note also that the difference between r_t^u and γ increases for larger values of S. This is caused by the rapid increase in γ due to the larger sizes of the hyper rectangles used in the corresponding approximations. A larger difference between r_t^u and γ should help in a significant improvement for the proposed method, and this is confirmed by the results in Section 5.2.3.

Note that Figure 5 provides an overall evaluation of measurements $r_t^u(Q)$ and $\gamma(Q, W_t)$. It is possible that for a certain

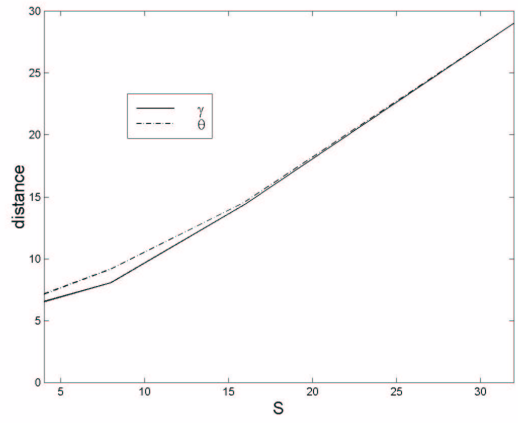


Fig 6. Overall measurements θ and γ computed from approximations at different resolutions

query Q_m and a certain resolution S, the inequality (10) does not hold. To be specific, we examine $r_t^u(Q_m)$ and $\gamma(Q_m, W_t)$ for all 50 queries at 4 resolutions, which provides 200 instances. Only for 2 instances we observe the values $r_t^u(Q_m)$ and $\gamma(Q_m, W_t)$ fail the inequality. Both happen at the finest resolution, $S = 4$.

5.2.2 Upper bound $\theta_t(Q)$ vs. $\gamma(Q, W_t)$

In Section 4, $\theta_t(Q)$ is introduced to remove some false approximations. The closer the value of $\theta_t(Q)$ is to the value of $\gamma(Q, W_t)$, the more is the number of approximations that are eliminated during the Phase I filtering. In Figure 6, the overall evaluation of the closeness of these two measurements are presented at different resolutions. The values are averaged over all users and over all quries, as discussed in Section 5.2.1.

We observed from the Figure 6 that $\theta_t(Q)$ does in fact approximate $\gamma(Q, W_t)$ quite well. Note that $\theta_t(Q)$ is computed from the Phase I candidates from a previous iteration, which typically is a very small fraction of the overall number of items in the database. This low computational complexity helps in screening out a large number of otherwise potential candidates for the current iteration of Phase I filtering.

5.2.3 Improvement on $|N_1(Q, W_t)|$

Here we examine the number of candidates found from Phase I filtering by using the standard approach, as introduced in Section 3.2, and the proposed approach from Section 4.

For a given resolution S, the average of $|N_1(Q_m, W_t)|$, $m = 1, \ldots, 50$, is computed for both approaches. We denote the averages for the standard approach and the proposed approach to be $stan(|N_1|)$ and $prop(|N_1|)$ respectively. Since we are concerned with the relative improvement of using the proposed algorithm, the ratio of the two averages, computed as

$$\alpha = stan(|N_1|)/prop(|N_1|) \quad (15)$$

is used to measure the effectiveness. Figure 7(a) gives the ratios for different resolutions. α ranges from 4 at finer resolutions ($S = 4$) to about 60 for coarser resolutions ($S = 32$), thus demonstrating a significant overall reduction in the number of disk or page accesses.

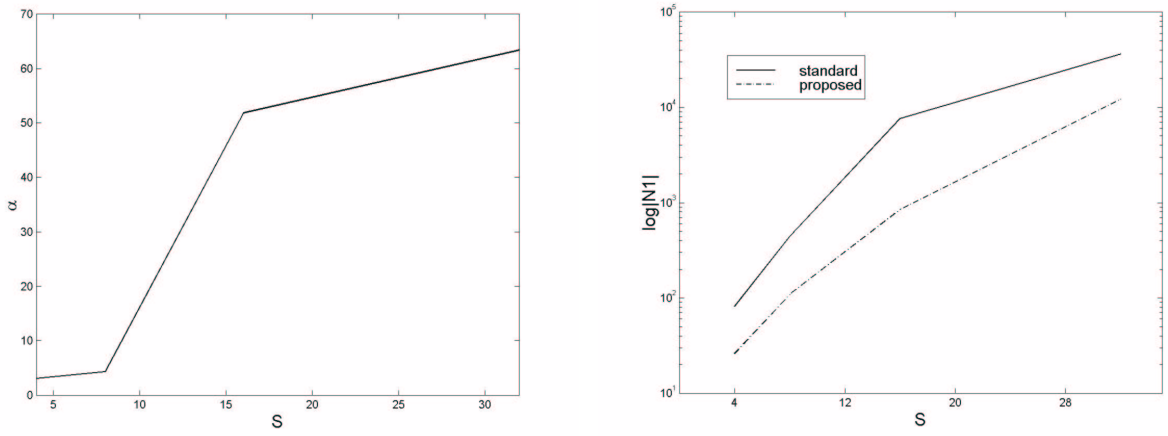
Figure 7(b) shows the actual values of $stan(|N_1|)$ and $prop(|N_1|)$ of all resolutions on a log scale. It is interesting to note that when $|N_1|$ is large, α is also large.

6 Discussion

In this work an adaptive search scheme for computing nearest neighbors of a given query in the context of relevance feedback is presented. During relevance feedback iterations, the underlying similarity metric is changing. The proposed method exploits the correlations between the nearest neighbors in successive iterations to filter out a large number of otherwise potential candidates in a two phase search process. The method is tested on a large image database and a significant performance improvement is demonstrated. The proposed method thus facilitates iterative learning in large databases so that the scalability of the learning algorithms can be better understood. While we discuss the proposed algorithm in the context of approximation based index structures, similar modifications can be made to tree based index structures as well. Also, it can be extended to cases where the weight matrix in the relevance feedback update is not diagonal.

Acknowledgments

This research was in part supported by the following grants/ awards: The Institute of Scientific Computing Research (ISCR) award under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No.



(a) (b)

Fig 7. (a) Ratio α for different resolutions. (b) The actual value of $|N_1|$ for both approaches

W-7405-ENG-48, NSF IRI#9704785, NSF #EIA-9986057, NSF #EIA-0080134, and by Samsung Electronics. The authors would like to thank Prof. S. Chandrasekaran and J. Tesic for many fruitful discussions, and S. Newsam and B. Sumengen for the datasets used in the experiments.

7 References

[1] J.R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Jain, and C.F. Shu, "The virage image search engine: An open framework for image management," In Proceedings of SPIE, Storage and Retrieval for Still Image and Video Databases IV, pages 76--87, San Jose, CA, USA, February 1996..

[2] S. Berchtold, D. A. Keim, and H. P. Kreigel, "The X-tree: An index structure for high-dimensional data," In Proc. of the 22nd Int'l Conference on Very Large Databases, Bombay, India, September 1996, pp. 28-39.

[3] S. Berchtold, C. Bohm, H.V. Jagadish, H. Kriegel, J. Sander. "Independent quantization: a index compression technique for high-dimensional data spaces," Proc. Int. Conf. on Data Engineering, San Diego, USA, 2000, pp. 577-588.

[4] K. Chakrabarti and S. Mehrotra, "The hybrid tree: An index structure for high dimensional feature spaces," In Proc. Int. Conf. Data Engineering, pp. 440-447, Sydney, Australia, 1999.

[5] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: an efficient access method for similarity search in metric spaces," In Proc. of the 23rd Conference on Very Large Databases (VLDB'97), pages 426--435, Athens, Greece, Aug. 1997.

[6] T. H. Cormen, C. E. Leiserson and Ronald L Rivest, "Introduction to algorithms," McGraw-Hill Book Company, 1992.

[7] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, A. E. Abbadi, "Vector approximation based indexing for non-uniform high dimensional data sets," Proceedings of the 9th ACM International Conference on Information and Knowledge Management (CIKM), pp. 202-209, Washington, DC, USA. November 2000.

[8] A. Guttman, "R-trees: a dynamic index structure for spatial searching," Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 47--57, Aug. 1984.

[9] Y. Ishikawa, R. Subramanya, and C. Faloutsos, "Mindreader: Query databases through multiple examples," In Proc. of the Int'l. Conf. on Very Large Data Bases, pages 218-227, New York, NY, USA, August 1998.

[10] C. Meilhac and C. Nastar, "Relevance feedback and category search in image databases," Proceedings of IEEE International Conference on Multimedia Computing and Systems, pp. 512-517, Florence, Italy, 7-11 June 1999.

[11] MPEG-7 Visual part of eXperimentation Model Version 8.0, ISO/IEC JTC1/SC29/WG11 #N3673, La Baule, October 2000.

[12] MPEG-7 Description of Color/Texture core experiments, ISO/IEC JTC1/SC29/WG11 #N2929, Melbourne, Australia, October 1999.

[13] W. Niblack, R. Barber, and et al., "The QBIC project: Querying images by content using color, texture and shape," Proceedings of the SPIE - The International Society for Optical Engineering, vol.1908, (Storage and Retrieval for Image and Video Databases, San Jose, CA, USA, 23 Feb. 1993.) 1993, p.173-87.

[14] A. L. Ratan, O. Maron, W. E. L. Grimson and T. Lozano-Perez, "A Framework for learning query concepts in image classification," Proc. IEEE Conf. Computer Vision and Pattern Recognition, pages 423-431, 1999.

[15] Y. Rui, T. Huang, "Optimizing Learning in Image Retrieval", Proc. Int. Conf. on Computer Vision, pp. 236-243, 2000.

[16] Y. Rui, T. S. Huang, and S. Mehrotra, "Content-based image retrieval with relevance feedback in MARS," in Proc. of IEEE Int. Conf. on Image Processing '97, pages 815--818, October 1997.

[17] H. D. Tagare, "Increasing retrieval efficiency by index tree adaptation," Proceedings IEEE Workshop on Content-Based Access of Image and Video Libraries, San Juan, Puerto Rico, 20 June 1997, pp. 28-35.

[18] R. Webber, J.-J. Schek and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional space," Proceedings of the Int. Conf. on Very Large Data bases, pp. 194-205, New York City, New York, August 1998.

[19] D. White and R. Jain, "Similarity indexing with the SS-tree," in Proc. 12th IEEE International Conference on Data Engineering, New Orleans, Louisiana, Feb. 1996, pp. 516-523.

[20] P. Wu, "Search and Indexing of Large Image/Video Databases," Ph. D Thesis, Dept. Electrical and Computer Engineering, University of California, Santa Barbara, 2001.