Fast Query Point Movement Techniques for Large CBIR Systems

Danzhou Liu, *Student Member, IEEE,* Kien A. Hua, *Senior Member, IEEE,* Khanh Vu, and Ning Yu, *Student Member, IEEE*

***Abstract*—Target search in *content-based image retrieval* (CBIR) systems refers to finding a specific (target) image such as a particular registered logo or a specific historical photograph. Existing techniques, designed around query refinement based on relevance feedback, suffer from slow convergence, and do not guarantee to find intended targets. To address these limitations, we propose several efficient query point movement methods. We prove that our approach is able to reach any given target image with fewer iterations in the worst and average cases. We propose a new index structure and query processing technique to improve retrieval effectiveness and efficiency. We also consider strategies to minimize the effects of users' inaccurate relevance feedback. Extensive experiments in simulated and realistic environments show that our approach significantly reduces the number of required iterations and improves overall retrieval performance. The experimental results also confirm that our approach can always retrieve intended targets even with poor selection of initial query points.**

***Index Terms*—Content-based image retrieval, relevance feedback, target search, index structures.**

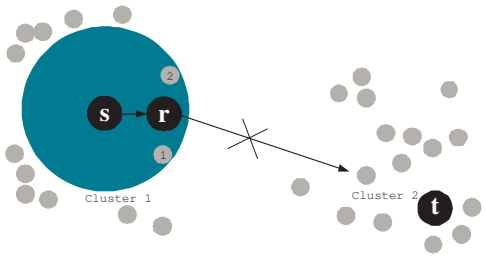

Fig. 1. Local maximum trap in existing approaches

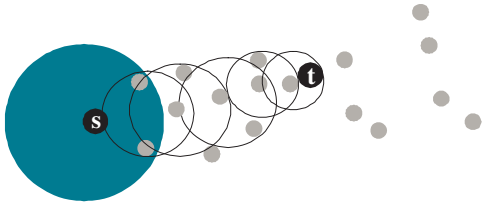

Fig. 2. Slow convergence in existing approaches

I. Introduction

CONTENT-based image retrieval (CBIR) has received much attention in the last decade, which is motivated by the need to efficiently handle the immensely growing amount of multimedia data. Many CBIR systems have been developed, including QBIC [12], Photobook [27], MARS [26], [30], NeTra [22], PicHunter [10], Blobworld [7], VisualSEEK [34], SIMPLIcity [39] and others [2], [6], [9], [14], [17], [24], [32], [38]. In a typical CBIR system, low-level visual image features (e.g., color, texture and shape) are automatically extracted for image descriptions and indexing purposes. To search for desirable images, a user presents an image as an example of similarity, and the system returns a set of similar images based on the extracted features. In CBIR systems with *relevance feedback* (RF), a user can mark returned images as positive or negative, which are then fed back into the systems as a new, refined query for the next round of retrieval. The process is repeated until the user is satisfied with the query result. Such systems are effective for many practical CBIR applications [13].

There are two general types of image search: *target search* and *category search* [10], [13]. The goal of target search is to find a specific (target) image, such as a registered logo, a historical photograph, or a particular painting. The goal of category search is to retrieve a given semantic class or genre of images, such as scenery images or skyscrapers. In other words, a user uses target search to find a known image. In contrast, category search is used to find relevant images the user might not be aware ahead of time. We focus on target search in this paper.

Two orthogonal issues in CBIR research are efficiency and accuracy. For instance, indexing techniques, such as R*-tree [3], may improve the efficiency of the search process. Their retrieval accuracy, however, depends on the effectiveness of the visual features used to characterize the database images. An effective CBIR system, therefore, needs to have both an efficient search mechanism and an accurate set of visual features. Addressing the effectiveness of the visual features is beyond the scope of this paper. We assume that the Euclidean distances between the images reflect their semantic similarity, and focus on investigating new search techniques to improve the efficiency of target search.

Existing target search techniques re-retrieve previously examined images (i.e., those retrieved in the previous iterations) when they again fall within the search range of the current iteration. This strategy leads to the following disadvantages:

- No guarantee that the target can be found. The search operation generally takes several iterations of relevance feedback to examine a number of regions in the feature space, before it reaches the target image. During this iter-

Manuscript received April 30, 2007.

Danzhou Liu, Kien A. Hua, Khanh Vu, and Ning Yu are with the School of EECS, University of Central Florida, Orlando, Florida 32816, USA. E-mail: {dzliu, kienhua, khanh, nyu}@cs.ucf.edu.

Digital Object Indentifier 10.1109/TKDE.2008.188

ative process, the search advancement might get trapped in a region as illustrated in Figure 1. It shows s and t as the starting point p_s and the target point p_t, respectively. Initially, the 3-NN search with p_s as the query point yields three points p_s, p_1, and p_2 as the query result. Let us say, the user marks points p_1 and p_2 as relevant. This results in point p_r, their centroid, as the new query point. With p_r as the refined query, the next 3-NN computation again retrieves points p_1, p_2, and p_s as the result. In this scenario, the search process is trapped in this local region, and can never reach the target point p_t. Although, the system can escape the local maximum trap with a larger k, it is difficult to guess a proper threshold ($k = 14$ in this example). Consequently, the user might not even know a local maximum trap is occurring.

- Slow convergence. Including previously examined images in the computation of the current centroid results in repeat retrieval of some of the images. This prevents a more aggressive movement of the search in the feature space. This drawback is illustrated in Figure 2, where $k = 3$. It shows that it takes six iterations for the search operation to reach the target point p_t. This slow convergence incurs longer search time, and significant computation and disk access overhead.

To address the aforementioned limitations, we propose four target search methods: naïve random scan (NRS), local neighboring movement (LNM), neighboring divide-and-conquer (NDC), and global divide-and-conquer (GDC) methods. All these schemes are built around a common strategy: they do not reexamine previously checked images. Furthermore, NDC and GDC exploit Voronoi diagrams to aggressively prune the search space in order to move faster towards the target image. We formally prove that GDC and NDC converge much faster than NRS and other methods can. Our extensive experimental results confirm our complexity analysis, and show the advantage of the proposed techniques in both simulated and realistic environments. A preliminary version of this study was presented in [21]. In the current paper, we extend the original technique to consider inaccurate user relevance feedback. Furthermore, we introduce a new index structure and the corresponding query processing technique to further improve performance. More experimental results are also provided to make the study more complete.

The remainder of this paper is organized as follows. In Section II, we survey related work on target search. The proposed methods are presented in Section III in detail. Handling inaccurate user relevance feedback is discussed in Section IV. We introduce a new index structure and query processing technique for target search in Section V. Our experimental results are given in Section VI. Finally, we conclude the paper in Section VII.

II. Related Work

In this section, we survey existing techniques for target search and category search. Category search techniques can be used for target search if the desired category has only one target image.

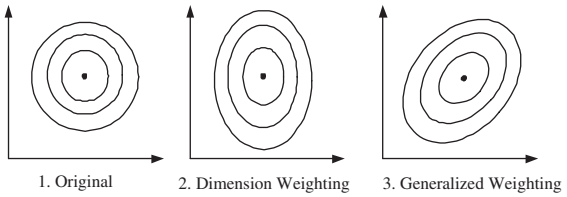

Fig. 3. Single-point movement query shapes

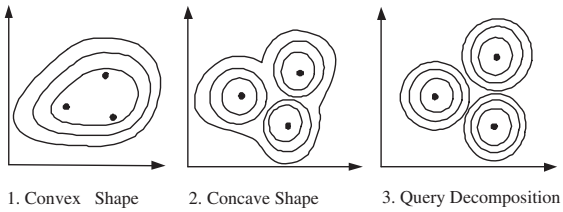

Fig. 4. Multiple-point movement query shapes

Two well-known techniques for target search were proposed in QBIC [12] and PicHunter [10]. IBM's QBIC system allows users to compose queries based on visual image features such as color percentage, color layout, and texture present in the target image, and ranks retrieved images according to those criteria. QBIC, however, is not an RF technique, so that it is difficult for users to define the ideal queries in the first try (because this system does not allow them to refine their queries as in recent RF systems). To lessen the burden on users, PicHunter proposes to predict query's intents by using a Bayesian-based RF technique to guide query refinement and target search. PicHunter's performance, however, depends on the consistency of users' behavior and the accuracy of the prediction algorithm. More importantly, both QBIC and PicHunter do not guarantee to find target images and suffer local maximum traps.

Techniques for category search can be divided into two groups: single-point and multiple-point movement techniques. A technique is classified as a single-point movement technique if the refined query Q_r at each iteration consists of only one query point. Otherwise, it is a multiple-point movement technique. Typical query shapes of single-point movement and multiple-point movement techniques are shown in Figures 3 and 4, where the contours represent equi-similarity surfaces. Single-point movement techniques, such as MARS [26], [30] and MindReader [18], construct a single query point close to relevant images and away from irrelevant ones. MARS uses a weighted distance (producing shapes shown in Figure 3.2), where each dimension weight is inversely proportional to the standard deviation of the relevant images' feature values in that dimension. The rationale is that a small variation among the values is more likely to express restrictions on the feature, and thereby should carry a higher weight. On the other hand, a large variation indicates this dimension is not significant in the query, thus should assume a low weight. MindReader achieves better results by using a generalized weighted distance, see Figure 3.3 for its shape. Ostensive relevance feedback [5] can be used to adjust the weights based on the checked images, while the length of time since an image was checked is used

in a decay function to modulate the impact of those already checked images.

In multiple-point movement techniques such as Query Expansion [8], Qcluster [20], and Query Decomposition [17], multiple query points are used to define the ideal space that is most likely to contain relevant results. Query Expansion groups query points into clusters and chooses their centroids as Q_r's representatives (see Figure 4.1). The distance of a point to Q_r is defined as a weighted sum of individual distances to those representatives. The weights are proportional to the number of relevant objects in the clusters. Thus, Query Expansion treats local clusters differently, as opposed to the equal treatment in single-point movement techniques.

In some queries, clusters are too far apart for a unified, all-encompassing contour to be effective; separate contours can yield more selective retrieval. This observation motivated Qcluster to employ an adaptive classification and cluster-merging method to determine optimal contour shapes for complex queries. Qcluster supports disjunctive queries, where similarity to any of the query points is considered as good, see Figure 4.2. To handle disjunctive queries both in vector space and in arbitrary metric space, a technique was proposed in FALCON [40]. It uses an aggregate distance function to estimate the (dis)similarity of an object to a set of desirable images. To bridge the semantic gap more effectively, we recently proposed Query Decomposition [17]. Based on the user's relevance feedback, this scheme automatically decomposes a given query into localized subqueries, which more accurately capture images with similar semantics but in very different appearance (e.g., the front view and side view of a car), see Figure 4.3. Other techniques [13], [36], [39] are also available to address the semantic gap. This issue is beyond the scope of this paper. In general, the above category search techniques do not guarantee to find target images and still suffer slow convergence, local maximum traps and high computation overhead.

To avoid local maximum traps and their associated problems, our methods will ignore all checked images. They will be discussed in the order of their sophistication in the next section. The most complex GDC is based on the single-point movement method, which proves to converge faster than multiple-point movement methods. GDC employs Voronoi diagrams to prune irrelevant images, assisting users in query refinement and speeding up convergence.

Unlike queries in traditional database systems, users in most cases cannot specify an ideal query to retrieve the desired result in multimedia database systems, and have to rely on iterative feedback to refine their queries. Target search may involve four typical types of queries: sampling queries [8], [10], [12], [17], [18], [20], [21], [26], constrained sampling queries [21], k-NN queries [18], [26] and constrained k-NN queries [21] as discussed in Section III. Among all the aforementioned techniques, only Chakrabarti *et al.* discussed how to efficiently evaluated k-NN queries in the Query Expansion model [8] for category search. They observed that the refined queries in the Query Expansion model are not modified dramatically from one iteration to another. Instead of evaluating subsequent queries from scratch, they proposed several techniques to save most of the disk I/O cost and CPU cost by appropriately exploiting the information generated during the previous iterations.

In related work, there have been many research efforts in sampling for selectivity estimation [41], [42], real-time CPU scheduling for mobile multimedia systems [43], and efficiently evaluating k-NN queries [16], [19], [29], [37] and constrained k-NN queries [11] without relevance feedback, but much less has been reported on efficiently answering sampling queries, constrained sampling queries and constrained k-NN queries involved in target search, where we need to consider iterative feedback and users' inaccurate relevance feedback. Most existing hierarchical index structures (e.g., R-tree [15], R*-tree [3], and A-tree [31]) were not designed specifically for target search, which typically cannot be answered in one iteration and may require auxiliary information (e.g., sampling points) to answer sampling queries and constrained sampling queries. Collecting auxiliary information on the fly during each feedback iteration causes overheads on CPU and disk I/O. In summary, a new index structure and efficient query processing technique for target search are highly demanded.

III. Target Search Methods

In this section, we present the four proposed target search methods. Again, the goals of our target search methods are avoiding local maximum traps, achieving fast convergence, reducing resource requirements, and guaranteeing to find target images.

Reconsidering already checked images is one of the several shortcomings of existing techniques that leads to the local maximum trap problem and slow convergence; the idea of leaving out checked images is our chief motivation for a new design principle. To simplify discussion, we assume that users are able to accurately identify the most relevant image from the returned images, and this most relevant image is the closest to the target image among the returned ones. Table I summarizes the notations we use throughout this paper.

In target search, the ultimate goal is to locate the target images, and if none is found, the final precision and recall of the search is zero. In CBIR with RF, the traditional recall and precision can be computed for individual iterations. For target search, we will use the so-called 'aggregate' recall and precision: if after several, say i, iterations the target image is found, the average precision and recall are $1/(i \cdot k)$ and $1/i$, where k is the fixed number of images retrieved at each iteration. In short, the number of iterations to find a target image is not only the most significant measure of efficiency, but also the most significant indicator of precision and recall. Therefore, we use the number of iterations as the major measure for theoretical analysis and experimental evaluation of the four proposed target search methods.

A query for target search is defined as $Q = \langle n_Q, P_Q, W_Q, D_Q, \mathbb{S}', k \rangle$, where n_Q denotes the number of query points in Q, P_Q the set of n_Q query points in the current search space $\mathbb{S}'$, W_Q the set of weights associated with P_Q, D_Q the distance function, and k the number of points to be retrieved in each iteration (see Figure 5). As

TABLE I
NOTATIONS

Notation	Description
Q	a query
k	the number of data points to be retrieved with Q
n_Q	the number of query points in Q
P_Q	a set of n_Q query points in Q
W_Q	a set of weights associated with P_Q
D_Q	the distance function for Q
Q_s	the starting query
Q_r	a refined query at a feedback iteration
$\mathbb{S}_k$	the query result set
$\mathbb{S}$	the whole space (i.e., the whole image database)
$\|\mathbb{S}\|$	the cardinality of $\mathbb{S}$
$\mathbb{S}'$	the current search space, where $\mathbb{S}' \subseteq \mathbb{S}$
p_s	the starting query point
p_t	the target point (i.e., the target image)
M	the node capacity (i.e., fanout)
m	the minimum number of entries in a node
MBB	the minimum bounding box
CBIR	content-based image retrieval
RF	relevance feedback
NRS	the naïve random scan method
LNM	the local neighboring movement method
NDC	the neighboring divide-and-conquer method
GDC	the global divide-and-conquer method

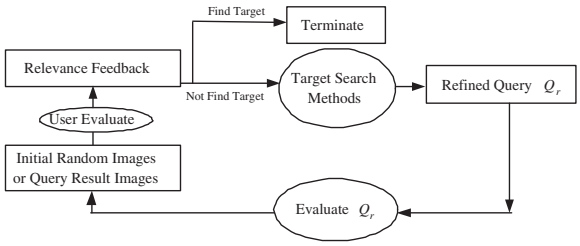

Fig. 5. Overview of the target search systems

discussed in Section II, various techniques have been proposed to automatically determine n_Q and P_Q as well as adjusting W_Q and D_Q for improved retrieval effectiveness. For single-point movement techniques, $n_Q = 1$; for multiple-point movement techniques, $n_Q > 1$. Now we illustrate below how to use this model to represent the four typical types of queries:

- For a sampling query, we set $n_Q = 0$ and $\mathbb{S}' = \mathbb{S}$, which signify that this query is to randomly retrieve k points in the whole image database $\mathbb{S}$.
- For a constrained sampling query, we set $n_Q = 0$.
- For a k-NN query with single-point movement techniques, we set $n_Q = 1$ and $\mathbb{S}' = \mathbb{S}$; For a k-NN query with multiple-point movement techniques, $n_Q > 1$ and $\mathbb{S}' = \mathbb{S}$.
- For a constrained k-NN query with single-point movement techniques, we set $n_Q = 1$ while for a constrained k-NN query with multiple-point movement techniques, $n_Q > 1$.

This definition is a generalized version of $Q = \langle n_Q, P_Q, W_Q, D_Q \rangle$ defined in [8], where the search space is assumed to be the whole database for every search. In our generalized definition, $\mathbb{S}'$ is included to account for the dynamic size of the search space, which shrinks gradually after each iteration. Let Q_s denote the starting query, Q_r a refined query at a feedback iteration, Q_t a target query which results in the retrieval of the intended target, and $\mathbb{S}_k$ the query result set.

NAIVERANDOMSCAN($\mathbb{S}$, k)

Input:
set of images $\mathbb{S}$
number of retrieved images at each iteration k
Output:
target image p_t

```
01  Q_s ← ⟨0, P_Q, W_Q, D_Q, S, k⟩
02  S_k ← EVALUATEQUERY(Q_s) /* randomly retrieve k points in S */
03  S' ← S − S_k
04  while user does not find p_t in S_k do
05      Q_r ← ⟨0, P_Q, W_Q, D_Q, S', k⟩
06      S_k ← EVALUATEQUERY(Q_r) /* randomly retrieve k points in S' */
07      S' ← S' − S_k
08  enddo
09  return p_t
```

Fig. 6. Naïve Random Scan Method

A. Naïve Random Scan Method

The NRS method randomly retrieves k different images at a time until the user finds the target image or the remaining set is exhausted, see Figure 6. Specifically, at each iteration, a set of k random images are retrieved from the candidate (i.e. unchecked) set $\mathbb{S}'$ for relevance feedback (lines 2 and 6), and $\mathbb{S}'$ is then reduced by k (lines 3 and 7). Clearly, the naïve scan algorithm does not suffer local maximum traps and is able to locate the target image after some finite number of iterations. In the best case, NRS takes one iteration, while the worst case requires $\left\lceil \frac{|\mathbb{S}|}{k} \right\rceil$. On average NRS can find the target in $\left\lceil \sum_{i=1}^{\lceil \frac{|\mathbb{S}|}{k} \rceil} i / \left\lceil \frac{|\mathbb{S}|}{k} \right\rceil \right\rceil = \left\lceil (\left\lceil \frac{|\mathbb{S}|}{k} \right\rceil + 1)/2 \right\rceil$ iterations. In other words, NRS takes $\mathcal{O}(|\mathbb{S}|)$ to reach the target point regardless of data distribution. Therefore, NRS is only suitable for a small database set.

B. Local Neighboring Movement Method

Existing techniques allow already checked images to be reconsidered, which leads to several major drawbacks as mentioned in Section I. We apply our non-re-retrieval strategy to one such method, such as MindReader [18], to produce the LNM method. LNM is similar to NRS except lines 5 and 6 as follows:

```
05  Q_r ← ⟨n_Q, P_Q, W_Q, D_Q, S', k⟩ based on the user's relevance feedback
06  S_k ← EVALUATEQUERY(Q_r)  /* perform a constrained k-NN query */
```

Specifically, Q_r is constructed such that it moves towards neighboring relevant points and away from irrelevant ones, and a k-NN query is now evaluated against $\mathbb{S}'$ instead of $\mathbb{S}$ (lines 5 and 6). When LNM encounters a local maximum trap, it enumerates neighboring points of the query, and selects the one closest to the target. Therefore, LNM can overcome local maximum traps, although it could take many iterations to do so.

Again, one iteration is required in the best case. To simplify the following worst-case and average-case complexity

analysis, we assume that $\mathbb{S}$ is uniformly distributed in the n-dimensional hypercube and the distance between two nearest points is a unit.

Theorem 1: For LNM, the worst and average cases are $\left\lceil \sqrt{n}\sqrt[n]{|\mathbb{S}|}/\lceil \log_{2^n} k \rceil \right\rceil$ and $\left\lceil (\frac{\sqrt{n}\sqrt[n]{|\mathbb{S}|}}{\lceil \log_{2^n} k \rceil} + 1)/2 \right\rceil$, respectively, assuming $\mathbb{S}$ is uniformly distributed.

Proof: The hypercube's edge length is $\sqrt[n]{|\mathbb{S}|} - 1$, and the diagonal's $\sqrt{n}(\sqrt[n]{|\mathbb{S}|} - 1)$. Let the distance between the initial query point and the target point be l, then $l \leq \sqrt{n}(\sqrt[n]{|\mathbb{S}|} - 1) < \sqrt{n}\sqrt[n]{|\mathbb{S}|}$. Note that the expected radius for k-NN search in $\mathbb{S}$ is $r = \lceil \log_{2^n} k \rceil$ because the distance between two nearest points is a unit as given above. Since $\mathbb{S}' \subset \mathbb{S}$, k-NN search in LNM requires a radius larger than r, but less than $2r$. In other words, at each iteration, LNM moves towards the target image at an average speed of cr where $1 \leq c < 2$. It follows that the number of iterations needed to reach the target is $\lceil l/(c\lceil \log_{2^n} k \rceil) \rceil$, which is bounded by $\left\lceil \sqrt{n}\sqrt[n]{|\mathbb{S}|}/\lceil \log_{2^n} k \rceil \right\rceil$. Then, the worst and average cases are $\left\lceil \sqrt{n}\sqrt[n]{|\mathbb{S}|}/\lceil \log_{2^n} k \rceil \right\rceil$ and $\left\lceil (\frac{\sqrt{n}\sqrt[n]{|\mathbb{S}|}}{\lceil \log_{2^n} k \rceil} + 1)/2 \right\rceil$, respectively. ■

If the data were arbitrarily distributed, then the worst case could be as high as NRS's, i.e. $\left\lceil \frac{|\mathbb{S}|}{k} \right\rceil$ iterations (e.g., when all points are on a line). In summary, in the worst case LNM could take anywhere from $\mathcal{O}(\sqrt[n]{|\mathbb{S}|})$ to $\mathcal{O}(|\mathbb{S}|)$.

C. Neighboring Divide-and-Conquer Method

Although LNM can overcome local maximum traps, it does so inefficiently, taking many iterations and in the process returning numerous false hits. To speed up convergence, we propose to use Voronoi diagrams [1], [28] in NDC to reduce search space. The Voronoi diagram approach finds the nearest neighbors of a given query point by locating the Voronoi cell containing the query point. Specifically, NDC searches for the target as follows, see Figure 7. From the starting query Q_s, k points are randomly retrieved (line 2). Then the Voronoi region VR_i is initially set to the minimum bounding box of $\mathbb{S}$ (line 3). In the **while** loop, NDC first determines the Voronoi seed set $\mathbb{S}_{k+1}$ (lines 6 to 10) and p_i, the most relevant point in $\mathbb{S}_{k+1}$ according to the user's relevance feedback (line 11). Next, it constructs a Voronoi diagram VD inside VR_i using $\mathbb{S}_{k+1}$ (line 12). The Voronoi cell region containing p_i in VD is now the new VR_i (line 13). Because only VR_i can contain the target (as proved in Theorem 2), we can safely prune out the other Voronoi cell regions. To continue the search in VR_i, NDC constructs a k-NN query using p_i as the anchor point (line 15), and evaluates it (line 16). The procedure is repeated until the target p_t is found. When NDC encounters a local maximum trap, it employs Voronoi diagrams to aggressively prune the search space and move towards the target image, thus significantly speeding up the convergence. Therefore, NDC can overcome local maximum traps and achieve fast convergence. We prove the following invariant.

Theorem 2: The target point is always contained inside or on an edge (surface) of VR_i, the Voronoi cell region enclosing the most relevant point p_i.

NEIGHBORINGDIVIDECONQUER($\mathbb{S}$, k)

Input:
set of images $\mathbb{S}$
number of retrieved images at each iteration k
Output:
target image p_t

01 $Q_s \leftarrow \langle 0, P_Q, W_Q, D_Q, \mathbb{S}, k \rangle$
02 $\mathbb{S}_k \leftarrow$ EVALUATEQUERY(Q_s) /* randomly retrieve k points in $\mathbb{S}$ */
03 $VR_i \leftarrow$ the minimum bounding box of $\mathbb{S}$
04 $iter \leftarrow 1$
05 **while** user does not find p_t in $\mathbb{S}_k$ **do**
06 **if** $iter \neq 1$ **then**
07 $\mathbb{S}_{k+1} \leftarrow \mathbb{S}_k + \{p_i\}$
08 **else**
09 $\mathbb{S}_{k+1} \leftarrow \mathbb{S}_k$
10 **endif**
11 $p_i \leftarrow$ the most relevant point $\in \mathbb{S}_{k+1}$
12 construct a Voronoi diagram VD inside VR_i using points in $\mathbb{S}_{k+1}$ as Voronoi seeds
13 $VR_i \leftarrow$ the Voronoi cell region associated with the Voronoi seed p_i in VD
14 $\mathbb{S}' \leftarrow$ such points $\in \mathbb{S}$ that are inside VR_i except p_i
15 $Q_r \leftarrow \langle 1, \{p_i\}, W_Q, D_Q, \mathbb{S}', k \rangle$
16 $\mathbb{S}_k \leftarrow$ EVALUATEQUERY(Q_r) /* perform a constrained k-NN query */
17 $iter \leftarrow iter + 1$
18 **enddo**
19 **return** p_t

Fig. 7. Neighboring Divide-and-Conquer Method

Proof: Theorem 2 can be proved by contradiction. First, note that according to the properties of the Voronoi cell construction, if VR_i contains the most relevant point (i.e. the closest point) p_i to the target point p_t, its seed p_i is the nearest neighbor of p_t among $\mathbb{S}_{k+1}$. Suppose p_t is inside VR_j, $i \neq j$. Then there exits another point in $\mathbb{S}_{k+1}$ closer to p_t than p_i, a contradiction. ■

Figure 8 explains how NDC approaches the target. In the first iteration, $\mathbb{S}_k = \{p_1, p_2, p_s\}$ is randomly picked by the system, assuming $k = 3$. The user identifies p_s as p_i (the most relevant point in $\mathbb{S}_k$). NDC then constructs a Voronoi diagram based on those three points in $\mathbb{S}_{k+1} = \mathbb{S}_k$, partitioning the search space into three regions. According to Theorem 2, the target must be in VR_i. NDC thus ignoring the other two regions, performs a k-NN query anchored at p_s and retrieves $\mathbb{S}_k = \{p_3, p_4, p_5\}$, the three closest points inside VR_i. Again, the user correctly identifies p_5 as the most relevant point in $\mathbb{S}_{k+1} = \{p_s, p_3, p_4, p_5\}$. The system constructs a Voronoi diagram and searches only the Voronoi cell associated with p_5. The search continues and, finally, at the fourth iteration, the target point is reached as the result of a k-NN query of p_6, the most relevant point in $\{p_5, p_6, p_7, p_8\}$ retrieved in the third iteration. We now determine the worst-case complexity for NDC, assuming that $\mathbb{S}$ is uniformly distributed.

Theorem 3: Starting from any point in $\mathbb{S}$, NDC can reach any target point in $\mathcal{O}(\log_k |\mathbb{S}|)$ iterations.

Proof: At the first iteration, $\mathbb{S}$ is divided into k Voronoi cells. Since the points are uniformly distributed from which k points are randomly sampled, each VR is expected to contain $\left\lceil \frac{|\mathbb{S}|}{k} \right\rceil$ points. According to Theorem 2, we only need to search one VR, which contains about $\left\lceil \frac{|\mathbb{S}|}{k} \right\rceil$ points. In the second

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

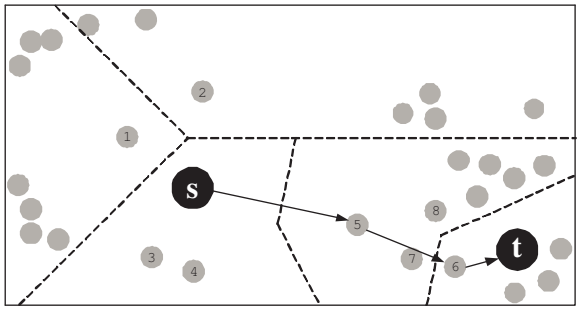

Fig. 8. Example of NDC

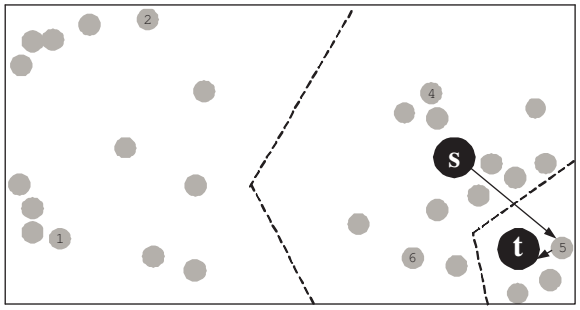

Fig. 9. Example of GDC

iteration, the searched VR contains $\left\lceil (\frac{|\mathbb{S}|}{k} - 1)/k \right\rceil \simeq \lceil |\mathbb{S}|/k^2 \rceil$ points. In the i^{th} iteration, each VR contains about $\left\lceil \frac{|\mathbb{S}|}{k^i} \right\rceil$ points. Since $\frac{|\mathbb{S}|}{k^i} \geq 1$, NDC will stop by $i \leq \log_k |\mathbb{S}|$. Hence, NDC reaches the target point in no more than $\mathcal{O}(\log_k |\mathbb{S}|)$ iterations. ■

When $\mathbb{S}$ is arbitrarily distributed, the worst case could take up to $\lceil \frac{\mathbb{S}}{k} \rceil$ iterations (e.g., all points are on a line), the same as that of NRS. In other words, NDC could still require $\mathcal{O}(|\mathbb{S}|)$ iterations to reach the target point in the worst case.

D. Global Divide-and-Conquer Method

To reduce the number of iterations in the worst case in NDC, we propose the GDC method. Instead of using a query point and its neighboring points to construct a Voronoi diagram, GDC uses the query point and k points randomly sampled from VR_i. Specifically, GDC replaces lines 15 and 16 in NDC with:

15 $Q_r \leftarrow \langle 0, P_Q, W_Q, D_Q, \mathbb{S}', k \rangle$
16 $\mathbb{S}_k \leftarrow$ EVALUATEQUERY(Q_r) /* randomly retrieve k points in $\mathbb{S}'$ */

Similar to NDC, when encountering a local maximum trap, GDC employs Voronoi diagrams to aggressively prune the search space and move towards the target image, thus significantly speeding up the convergence. Therefore, GDC can overcome local maximum traps and achieve fast convergence.

Figure 9 shows how the target could be located according to GDC. In the first iteration, $\mathbb{S}_k = \{p_1, p_2, p_s\}$ is the result of $k = 3$ randomly sampled points, of which p_s is picked as p_i. Next, GDC constructs a Voronoi diagram and searches the VR enclosing p_s. At the second iteration, $\mathbb{S}_{k+1} = \{p_s, p_4, p_5, p_6\}$ and p_5 is the most relevant point p_i. In the third and final iteration, the target point is located; GDC takes 3 iterations to reach the target point. We prove that the worst case for GDC is bounded by $\mathcal{O}(\log_k |\mathbb{S}|)$ regardless of data distribution.

Theorem 4: Starting from an initial point in $\mathbb{S}$, GDC can reach any target point in $\mathcal{O}(\log_k |\mathbb{S}|)$ iterations.

Proof: We will focus our attention on the size of VR at each iteration, keeping in mind that points are randomly sampled for Voronoi diagram construction. Thus, at the first iteration, the searched VR contains $\left\lceil \frac{|\mathbb{S}|}{k} \right\rceil$ points; at the second iteration, it contains $\left\lceil \frac{|\mathbb{S}|}{k \cdot (k+1)} \right\rceil$ points; and so on. At the i^{th} iteration, it contains $\left\lceil \frac{|\mathbb{S}|}{k \cdot (k+1)^{i-1}} \right\rceil$ points. Because $\frac{|\mathbb{S}|}{k \cdot (k+1)^{i-1}} > 1$, that is, it requires that $i < \log_k |\mathbb{S}|$. In other words, GDC can reach any target point in no more than $\mathcal{O}(\log_k |\mathbb{S}|)$ iterations. ■

Theorem 4 implies that for arbitrarily distributed datasets, GDC converges faster than NDC in general, although NDC might be as fast as GDC in certain queries, e.g., if the starting query point is close to the target point. In the previous example (Figure 8), NDC could also take three iterations, instead of four, to reach the target point if the initial k points were the same as in Figure 9, as opposed to Figure 8.

IV. Handle Inaccurate Relevance Feedback

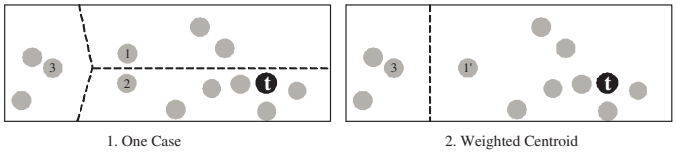

Fig. 10. One case and the weighted centroid

Users' inaccurate relevance feedback is a major issue for CBIR systems with RF. We need to make our system less sensitive to users' uncertainty. For simplicity, we have assumed that users accurately picked the most relevant image out of the returned images for each iteration in the above discussion. In practice, however, users could make a wrong choice, or they might pick several seemingly good choices instead of settling on one in a target search query. Hence, we should not assume the system is always presented with correct queries.

To deal with this situation, we construct, in each iteration, a single query point that is a weighted centroid of all the picked images, as in MARS and MindReader. For example, the visual difference between images 1 and 2 (illustrated in Figure 10.1) could be so small that there is a high probability that users select the wrong image (i.e., image 1) for the next iteration. If this happens, the target image may never be found unless backtracking is allowed in NDC and GDC (NRS and LNM still work). When a single good choice is uncertain such as in this case, users are allowed to mark those images as relevant, and our system will choose their weighted centroid as the refined query point, shown in Figure 10.2.

Detecting inaccurate relevance feedback is also desirable. The following theorem and lemma can facilitate the detection.

Theorem 5: If $cos(\alpha) < 0$ where α is the angle between one vector from the previous query point p_o to the new query

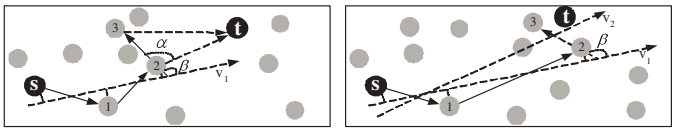

Fig. 11. Inaccurate relevance feedback and linear regression

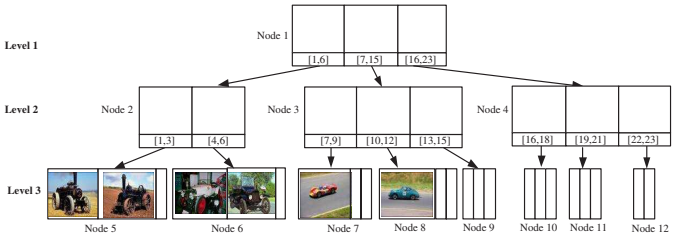

Fig. 12. Our new index structure

point p_n and the other from p_o to the target p_t, the user must be giving inaccurate relevance feedback.

Proof: When the user is giving inaccurate relevance feedback, the new query point p_n is farther from the target p_t compared with the previous query point p_o. Because $-180° \leq \alpha \leq 180°$ and $cos(\alpha) < 0$, then $90° < |\alpha| \leq 180°$. When $|\alpha| = 180$, p_n heads in the direction away from the target, and therefore the user must have given inaccurate relevance feedback. When $90° < |\alpha| < 180°$, $|\alpha|$ should be the largest angle in the triangle, made by p_o (e.g., point 2 in Figure 11.1), p_n (e.g., point 3 in Figure 11.1) and p_t. This is because it is impossible to have two obtuse angles in this triangle. Then the edge connecting p_n and p_t is the longest edge in the triangle based on triangle's properties. This means p_n is farther from p_t compared with p_o, indicating the user must have given inaccurate relevance feedback. ■

Lemma 1: The relevance feedback must be inaccurate if

$$c_1\overrightarrow{p_op_n} \cdot c_2\overrightarrow{p_op_t} = c_1c_2|\overrightarrow{p_op_n}||\overrightarrow{p_op_t}|cos(\alpha) < 0$$

where c_1 and c_2 are two positive constants.

Lemma 1 simplifies the detection of inaccurate relevance feedback since only the vectors' directions, and not their magnitudes matter. Even though the exact location of the target might be unknown, but its position relative to other results can be inferred from its visual features the user already knows. Thus, for a given target image, the user knows how to move towards the target in the search space. Based on the user's feedback, our target search technique is able to zoom in a narrower space the target must be in. In other words, we now know the approximate whereabouts of the target, though not its exact location (within that space). Lemma 1 relaxes the requirement of the exact location (i.e., only the direction of $\overrightarrow{p_op_t}$ is needed to approximate). Assume that most of user's behavior is consistent; i.e., the probability of accurate relevance feedback is larger than 0.5. This approximation problem can be treated as a probability problem. Basically, the more query points, the better the approximation. One way to estimate the direction is to use linear regression. For example, suppose the user has made four feedback iterations (see Figure 11.1), moving the query point from p_s, p_1, p_2 to p_3. Then the direction of $\overrightarrow{p_op_t}$ (i.e., $\overrightarrow{p_2p_t}$ in this case) can be approximated by vector $\overrightarrow{v_1}$, which moves towards the search space containing p_t and is a linear regression of points p_s, p_1 and p_2. In other words, α is approximated by β, where β is the angle between $\overrightarrow{p_2p_3}$ and $\overrightarrow{v_1}$. If the dot product of $\overrightarrow{p_2p_3}$ and $\overrightarrow{v_1}$ is less than 0, it suggests that the user is likely to have given inaccurate relevance feedback, and a warning should be issued. However, if the relevance feedback is in fact accurate and the dot product is less than 0, this indicates that the direction approximation of $\overrightarrow{p_op_t}$ is not accurate and we need to adjust it. For example, the dot product of $\overrightarrow{p_2p_3}$ and $\overrightarrow{v_1}$ is less than 0 in Figure 11.2 while the user's relevance feedback is accurate. We can replace p_2 with p_3 for linear regression (i.e., only use points p_s, p_1 and p_3 for linear regression, omitting p_2), producing $\overrightarrow{v_2}$ for the approximation of $\overrightarrow{p_op_t}$ in the next iteration.

Detecting users' inaccurate relevance feedback is a difficult and open problem. We rely on short-term memory—the last few relevance feedbacks—to predict the general direction towards the destination, and focus on warning users if their feedbacks seem to be contradictory (our technique is only able to give a summary warning to the user, who may not be able to tell which one among the previous steps is inaccurate). Such a warning points out to users that their consecutive feedbacks appear contradiction, and is helpful to users in providing a better relevance feedback for the subsequent rounds. Actually, we have taken the following steps to ensure our system is less sensitive to users' inaccurate relevance feedback, in design and in implementation. First, we still keep LNM besides GDC and NDC in our prototype. Although converging slowly, LNM is robust against inaccurate relevance feedback because it basically enumerates the candidate images. Second, we use the above proposed method to automatically monitor users' feedbacks, and issue warnings if inconsistent behaviors are detected. These warnings prompt the users to re-evaluate their feedbacks. Finally, our prototype allows users to backtrack their selections if missteps have been made.

V. Index Structure and Query Processing Technique for Target Search

In this section, we will discuss how to construct the index structure and to efficiently evaluate the three typical types of queries (i.e., sampling queries, constrained sampling queries and constrained k-NN queries) involved in the proposed target search methods.

A. Index Structure for Target Search

Our index structure (see Figure 12) is constructed in two stages as follows:

Hierarchical Clustering: A hierarchical clustering technique, similar to the R*-tree [3], is used to organize the entire image database into a hierarchical tree structure. With each node in this hierarchy representing a cluster, we extend the original node structure of the R*-tree to include also information to identify the images in their children nodes. We selected the R*-tree for our study because it is well known and

has been widely used in practice, although other hierarchical clustering techniques [4] can be used as well. The hierarchical clustering is constructed as follows. When a new element (i.e., an image represented as a high-dimensional point) is inserted into the tree, this element is placed in a leaf node that requires the least enlargement in its bounding box, and a leaf node's MBB is based on all dimensions of its contained image points. If a leaf node overflows, this node is split (i.e., a portion of its entries are removed from the node and reinserted into the tree), and such splits propagate up the tree [3].

Information Augmenting: We traverse the tree in a postorder fashion. In the original R*-tree, an *internal node* contains an array of *node entries*. Each node entry is a pair (*mbb*, *node-id*), where *mbb* is the minimum bounding box (MBB) that spatially contains the MBBs in the child node, with *node-id* as the child node address. In our index structure, each node entry is extended to be a tuple (*mbb*, *node-id*, *imageID-range*), where *imageID-range* refers to the range of image identifications contained in the pointed child node and *imageID-range* $\subseteq \big[1, |\mathbb{S}|\big]$. Let us describe how to augment the structure illustrated in Figure 12. We start from the root node (i.e., Node 1) which has three node entries. We first visit the first node entry which points to Node 2. Node 2 has two node entries, pointing to leaf nodes 5 and 6 in order. Our depth-first traversal leads us to Node 5, which contains 3 image points. Then we set the *imageID-range* of the first entry in Node 2 to $\big[1, 3\big]$, and each image contained in this node can randomly pick an exclusive ID within this range. That is, the three images in Node 5 can be assigned IDs 1, 2, and 3, respectively. We of course need to build up a one-one mapping between image IDs and exact image names, such as building a B^+-tree index on the image ID field, or simply changing the image names to their corresponding image IDs as in our current implementation. Similarly, we set the *imageID-range* of the second entry in Node 2 to $\big[4, 6\big]$. As Node 2 doesn't have any more node entries, we track back to Node 1 and set *imageID-range* to encompass the ranges of all its children, which is $\big[1, 3\big] \cup \big[4, 6\big] = \big[1, 6\big]$ in this example. The above procedure is repeated for the second entry of Node 1. The *imageID-range* values in different internal node entries are shown in Figure 12.

When a new image is inserted, the structure has to be rebuilt. Because image databases are fairly static [13], the reconstruction is still acceptable considering the performance gains (shown in Section VI) we obtain. Our index structure has two properties as stated in the following theorems. Let M (i.e., node capacity or fanout) denote the maximum number of entries that can fit in a node, m the minimum number of entries in a node (we set $m = M/2$ assuming M is an even number), L the total number of leaf nodes in our index structure, and L_Q the total number of leaf nodes related to a user query Q (assuming all related leaf nodes are contained in $\mathbb{S}'$ of Q).

Theorem 6: For sampling queries, if $k < L/2$, no retrieved points will be sampled from the same leaf node.

Proof: Recall that R*-tree is a height-balanced tree and the number of image points in each leaf node is between m and M. For sampling queries, $\mathbb{S}$ is the search space, therefore the corresponding *imageID-range* is $\big[1, |\mathbb{S}|\big]$. One possible solution is the following sampled image ID set $\big\{1, M+1, 2M+1, 3M+1, \ldots, (k-1)*M+1\big\}$. We first prove that $(k-1)M+1 \leq |\mathbb{S}|$, which means the highest ID in the above set is a valid ID. Obviously, $|\mathbb{S}| \geq L*m = L*M/2$. On the other hand, $(k-1)M+1 < (L/2-1)*M+1 \leq L*M/2$. It follows that $|\mathbb{S}| > (k-1)M+1$. Next we prove that no two images in this set are from the same leaf node. Clearly, if two images are in the same node, the difference between the values of the two corresponding IDs should be less than M (see Figure 12 for an example, $M = 3$). The difference between any two IDs in the above set is cM where c is a natural number, therefore no two images in the above set are from the same leaf node. ■

Theorem 7: For constraint sampling queries, if $k > 2L_Q$, each leaf node related to Q will be sampled.

Proof: Each relevant leaf node corresponds to an *imageID-range*. We union those *imageID-range*s, and transform the union into a consecutive range $\big[1, |\mathbb{S}'|\big]$ for analysis simplicity. One possible solution is: we sample the consecutive range at fixed interval $M/2$, then we obtain the following sampled image ID set

$$\big\{M/2, 2M/2, 3M/2, \ldots, s*M/2\big\},$$

where $s = \left\lfloor \frac{|\mathbb{S}'|}{M/2} \right\rfloor$. Obviously, $L_Q * M/2 \leq |\mathbb{S}'| \leq L_Q * M$, from which we have $L_Q \leq s \leq 2L_Q$. Since $k > 2L_Q$, then $L_Q \leq s < k$, which implies the number of sampled points in the above set is even fewer than k. Each relevant leaf node contains a *imageID-range* $\subseteq \big[1, |\mathbb{S}'|\big]$ with length at least $M/2$, therefore its *imageID-range* will contain at least one sampled point in the above set. Hence, each leaf node related to Q will be sampled. ■

The above desirable properties show that our index structure can facilitate sampling as many relevant leaf nodes as possible, and the sampled points can better capture the data distribution, thus are more representative. The results of our empirical study in Section VI confirm that our index structure, in fact, can help improve both retrieval effectiveness and retrieval efficiency.

B. Efficient Query Processing for Target Search

We discuss our query processing technique EVALUATEQUERY(Q) on top of the above index structure for the three types of queries (i.e., sampling queries, constrained sampling queries and constrained k-NN queries) based on our four target search methods. k-NN queries are omitted because our target search methods do not use them. The query cost is the sum of disk seek (including cylinder seek and rotation), data transfer and CPU time, in which seek time dominates the total query cost. Figure 13 lays out our query processing technique, designed to minimize the disk I/O cost.

For sampling queries, we just need to retrieve the root node (in line 3), which is stored in memory to reduce the disk I/O cost for the subsequent sampling queries. The root node contains all possible image IDs. If k is relatively small (i.e., $k < L/2$), we will choose the technique based on Theorem 6 to guarantee that no images will be sampled from the same leaf node in order to make the sampled images as representative

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

EVALUATEQUERY(Q)

Input:
given query Q
Output:
query result $\mathbb{S}_k$

```
01  S_k ← ∅
02  if Q is a sampling query then /* based on n_Q and S' */
03      read the root node of our index structure
04      if k < L/2 then
05          choose a random non-negative integer n such that
            imageIDset = {1+n, M+1+n, 2M+1+n,
            3M+1+n, ..., (k-1)*M+1+n}
            ⊆ {1, ..., |S|}
06      else
07          imageIDset ← randomly sample k IDs from
            {1, ..., |S|} stored in root
08      endif
09      S_k ← retrieve images whose ID ∈ imageIDset
10  elseif Q is a constrained sampling query then
11      IDset ← All image IDs whose corresponding points
        ∈ S' by performing a modified range query
12      if k > 2L_Q then
13          transform IDs ∈ IDset into a consecutive range
            [1, |S'|]
14          choose a random non-negative integer n such that
            imageIDset = {⌊M/2⌋ + n, ⌊2M/2⌋ + n,
            ⌊3M/2⌋+n, ..., ⌊s*M/2⌋+n} ⊆ {1, ..., |S'|}
15          transform all IDs ∈ imageIDset back to the original
            ones
16          add some different IDs ∈ IDset − imageIDset into
            imageIDset such that |imageIDset|=k
17      else
18          imageIDset ← randomly sample k IDs from IDset
19      endif
20      S_k ← retrieve images whose ID ∈ imageIDset
21  else /* Q is a constrained k-NN query */
22      Queue ← NEWPRIORITYQUEUE()
23      Insert nodeSet into Queue, where nodeSet is pruned
        by mbb(S')
24      while not ISEMPTY(Queue) and |S_k| < k do
25          Element ← DEQUEUE(Queue)
26          if Element is an image object and Element is
            inside mbb(S') then
27              S_k ← S_k ⋃ {the image corresponding to
                Element}
28          elseif Element is a leaf node then
29              for each Object in leaf node Element do
30                  ENQUEUE(Queue, Object,
                    OBJDIST(P_Q, Object))
31              enddo
32          else    /* Element is a non-leaf node */
33              for each Child node of node Element do
34                  ENQUEUE(Queue, Child,
                    DIST(P_Q, Child))
35              enddo
36          endif
37      enddo
38  endif
39  return S_k
```

Fig. 13. Query Processing Technique

as possible (in line 5). Otherwise, random sampling can be performed to retrieve the query result (in line 7).

For constrained sampling queries, we will use the minimum boundary box of $\mathbb{S}'$, denoted by $mbb(\mathbb{S}')$, as the range, then perform a modified range query on our index structure to collect sampling image IDs (in line 11). Since in our index structure, each internal node entry is a tuple (*mbb*, *node-id*, *imageID-range*), thus if $mbb(\mathbb{S}')$ contains a node's mbb, we can put the node's *imageID-range* into $IDset$ without visiting its children. If $mbb(\mathbb{S}')$ overlaps with or is contained in a node's mbb, we will visit its children recursively. Consequently, we can prune a lot of nodes to answer a constrained sampling query, therefore significantly reducing the disk I/O accesses. If k is relative large (i.e., $k > 2L_Q$), the technique based on Theorem 7 will guarantee that each leaf node related to Q will be sampled in order to make the sampled images more representative (from line 13 to 16). Otherwise, random sampling over $IDset$ is performed to retrieve the query result (in line 18). To take advantage of the shrinking of the search space $\mathbb{S}'$ after each iteration (e.g., in GDC), we can recycle the nodes visited in the previous iterations to avoid re-reading those nodes from the disk. That is, we can put those visited nodes into $nodeSet$ residing in memory, and perform the range query over $nodeSet$.

For constrained k-NN queries, we extend the well-known k-NN algorithm proposed in [16]. Considering the search space $\mathbb{S}'$ is shrunk after each iteration and the results of two consecutive constrained k-NN queries may overlap, we recycle the visited nodes in the previous iterations as does for constrained sampling queries. We first prune $nodeSet$ by $mbb(\mathbb{S}')$ (in line 23); that is, only nodes that either overlap with or are contained in $mbb(\mathbb{S}')$ are kept. To answer a given constrained k-NN query, all nodes in $nodeSet$ are inserted into a new priority queue $Queue$ (in line 23) while $nodeSet$ only contains the root node in the first iteration. In the **while** loop (from line 24 to 37), an element is dequeued from $Queue$ (in line 22). If the element is an image object and is contained in $mbb(\mathbb{S}')$, this image object is put into the query result S_k and the next element is dequeued from $Queue$ until all k nearest neighbors are found or $Queue$ is empty. If the element is a leaf node, all image objects in it are inserted into $Queue$ (in line 30). If the element is an internal node, its child nodes are inserted into $Queue$ (in line 34). In sum, we reuse the visited nodes in the previous iterations and prune $nodeSet$ by $mbb(\mathbb{S}')$ to reduce the disk I/O cost for answering constrained k-NN queries.

VI. EXPERIMENTS

In this section, we present experimental results for target search in both simulated and realistic environments, and evaluate the effectiveness of the query processing technique described in Section V.B. Our dataset consists of more than 68,040 images from the COREL library. 37 visual image features divided into three main groups were used: colors (9 features) [35], texture (10 features) [33], and edge structure (18 features) [44]. The combination of those features captures essential image characteristics and facilitates effective similarity

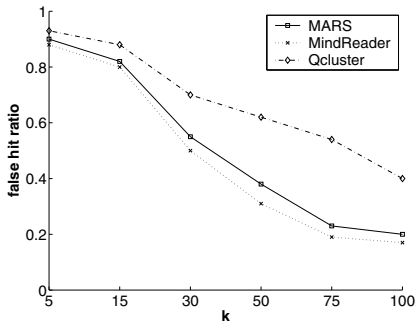

Fig. 14. False Hit Ratio

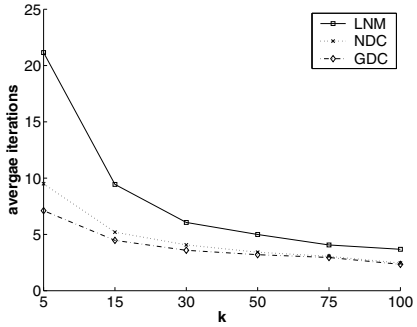

Fig. 15. Average Iterations

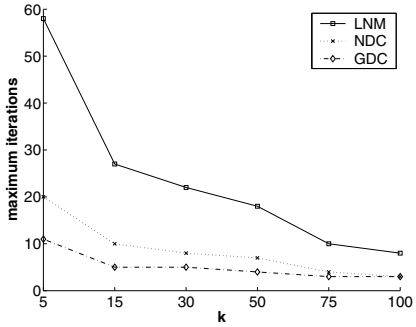

Fig. 16. Maximum Iterations

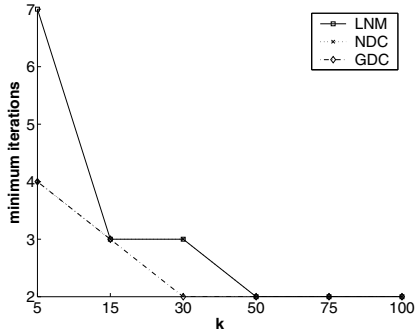

Fig. 17. Minimum Iterations

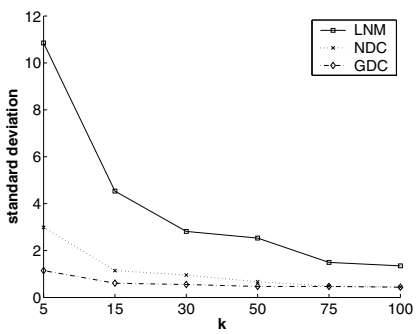

Fig. 18. Standard Deviation of Iterations

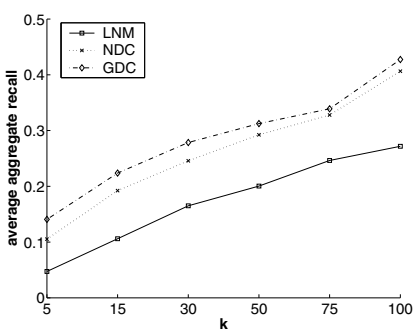

Fig. 19. Average Aggregate Recall

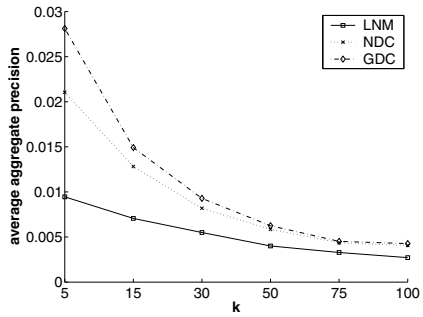

Fig. 20. Average Aggregate Precision

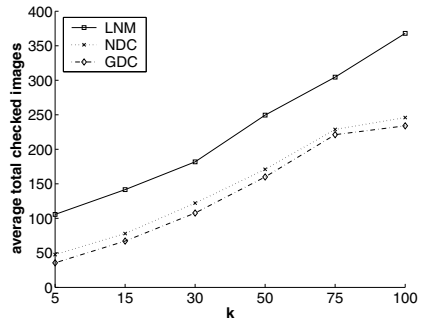

Fig. 21. Average Total Checked Images

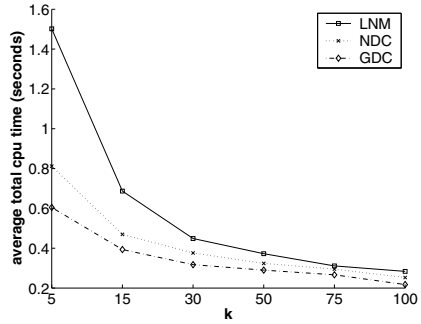

Fig. 22. CPU Time

comparison. Our experiments were run on Sun UltraSPARC with 2GB memory.

A. Simulated Experiments

In these experiments, we evaluated the performances of MARS [26], [30], MindReader [18], and Qcluster [20] against our techniques (NRS's results are omitted since its performance can be statistically predicted). All the data resided in memory. The performance metrics of interest are the average total visited images, precision, recall, computation time and the number of iterations (average, maximum, minimum, and their variance) needed for each method to retrieve an intended target. These were measured as k takes different values in $\{5, 15, 30, 50, 75, 100\}$. There were 100 pairs of starting points-target points selected randomly for the experiments.

In order to accurately evaluate the prime metrics, relevance feedback in these experiments was simulated: the point in the retrieval set closest to the target point was automatically selected as the most relevant point. To save computation overhead for NDC and GDC, we constructed the Voronoi region VR_i containing the most relevant point instead of the whole Voronoi diagram, and approximated VR_i by its minimum boundary box if VR_i contains too many surfaces.

To illustrate the common problems of slow convergence and local maximum traps with the existing approaches, we demonstrate that MARS, MindReader and Qcluster have poor false hit ratios for small k. Figure 14 shows that when k is small, their performance is affected by local maximum traps, i.e., their false hit ratios are very high even for a fairly large k. For example, when $k = 100$, MARS's false hit ratio is about

20% and Qcluster's exceeds 40%, while the best performer MindReader is just below the 20% mark. As a result, users of these techniques have to examine a large number of returned images, but might not find their intended targets.

In the experiments that produced the number of iterations, we had to make sure that the compared techniques could successfully reach the intended targets. We thus used LNM in place of MindReader (LNM is an improved version of MindReader, see Section III). The experimental results for LNM, NDC and GDC are shown in Figures 15 to 22. They show that NDC and GDC perform more efficiently when k is small, with GDC being slightly better than NDC. Specifically, when $k = 5$, the average numbers of iterations for LNM, NDC and GDC (see Figure 15) are roughly 21, 10 and 7, respectively (compared to $\frac{68040}{5} = 13608$ iterations in NRS); the maximum numbers are 58, 20 and 11, respectively (see Figure 16); and the minimum numbers are 7, 4 and 4, respectively (see Figure 17). The results also confirm our analysis of GDC complexity (see Figure 15): GDC can reach the target point in $\mathcal{O}(\log_k |\mathbb{S}|) = (\log_5 68040) = 6.9141 \simeq 7$ iterations.

The standard deviations of the iterations are shown in Figure 18. GDC and NDC are much more stable than LNM, with GDC's slightly more uniform than NDC's. This indicates that GDC and NDC can achieve fast convergence even with a poor selection of initial query points.

The average 'aggregate' recalls and precisions, defined in Section III, are shown in Figures 19 and 20 respectively. Again, experimental results show that NDC and GDC achieve better retrieval effectiveness (precision and recall) when k is small compared to LNM, with GDC being slightly better than NDC.

The average total checked images for LNM, NDC, and GDC in the experiments are plotted in Figure 21. The figure shows that GDC and NDC examined fewer than half of the total checked images of LNM (compared to $\frac{68040}{2} = 34020$ images need to be checked in NRS). In terms of CPU time, GDC is the most efficient, although the difference is smaller as k increases (see Figure 22). This is because NDC and GDC take some computation overhead to construct VR_i, while LNM requires more iterations and associated computation time for adjusting the generalized distance function. Overall, GDC and NDC significantly outperform LNM, with GDC slightly outdoing NDC. Figure 23 shows that using our index structure, the average iterations of GDC over 100 simulated target searches can be reduced by 1 when $k = 5$ and 15, although the difference is smaller as k increases. The reason is that our index structure is designed to sample as many relevant leaf nodes as possible and to better capture the data distribution, thus facilitating GDC to prune the search space.

B. Realistic Experiments

In simulated experiments, the most relevant points were assumed to be accurately selected among the returned points. In practice, however, this cannot be easily achieved by human evaluators, unless the most relevant images are distinctly stood out. To evaluate our methods' performance in realistic environments, we have extended the previous prototype [21] (based

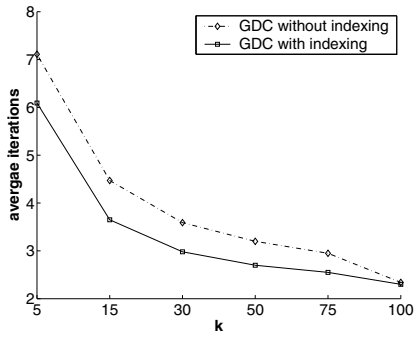

Fig. 23. GDC Average Iterations Comparison

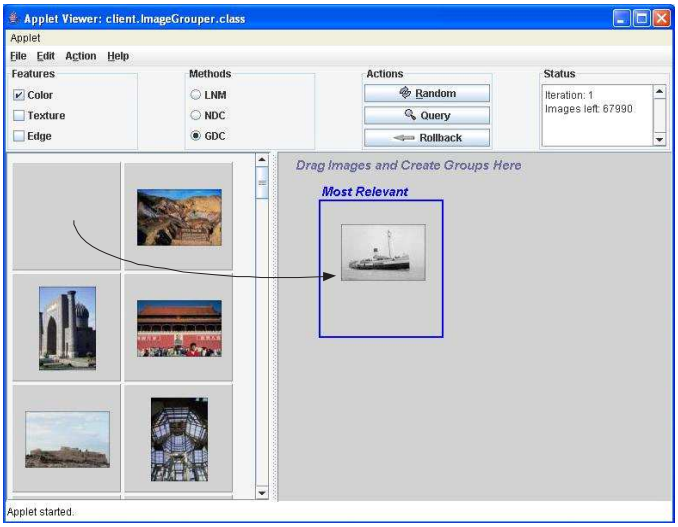

Fig. 24. Target Search GUI Interface

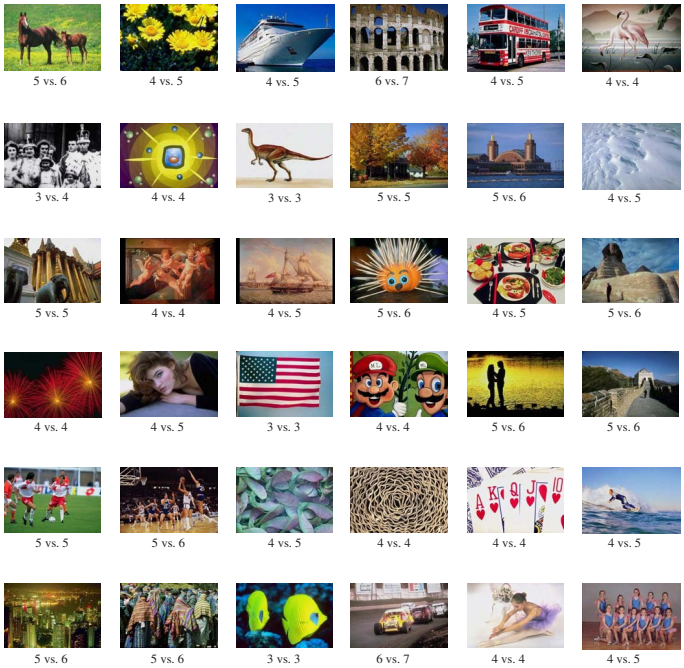

Fig. 25. Number of Iterations Comparison (with our index structure vs. without our index structure) with k=50

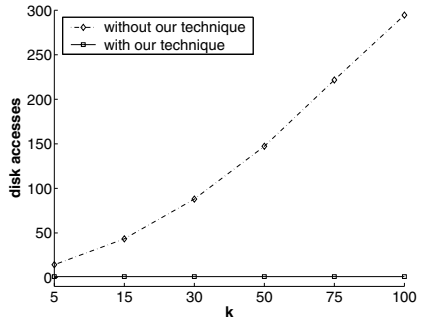

Fig. 26. Sampling Queries

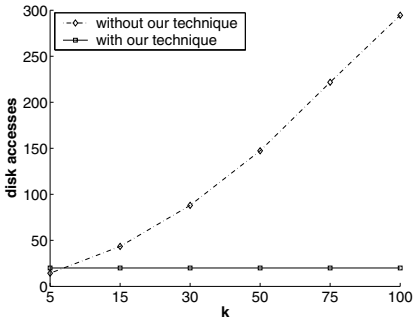

Fig. 27. Constrained Sampling Queries

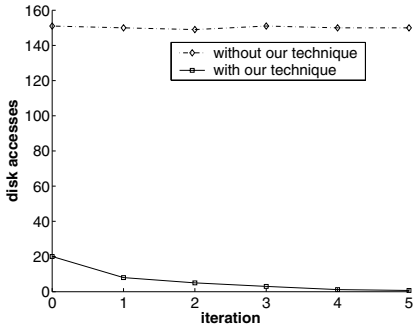

Fig. 28. Constrained Sampling Queries in Feedback Iterations ($k = 50$)

on ImageGrouper [23]) to couple with our index structure. Our prototype, shown in Figure 24, allows users to pose queries by dragging and grouping multiple relevant images on the work space (i.e., the right pane), choose discriminative visual features, and select one of the three retrieval methods (LNM, NDC and GDC). It monitors users' feedback and prompts users to reexamine their relevance feedback if certain conditions are true as discussed in Section IV. It also allows users to rollback their feedback in the previous iteration if they wish. Thus, for instance, if there are several relevant images, the user can group them together to form a query, and if he reaches a dead-end without finding the target image, he can rollback.

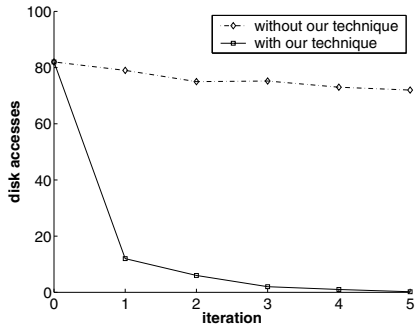

Fig. 29. Constrained k-NN Queries in Feedback Iterations ($k = 50$)

We trained 20 graduate students (i.e., 15 engineering students and 5 art students) to use the target search system and asked them to find 36 given target images from different semantic categories in both situations (with or without our index structure). In Figure 25, we show the results for finding the given 36 target images with $k = 50$ (i.e., 50 images were retrieved at each feedback iteration). The two images (i.e., an ancient building and race cars) took, on average, more iterations than the others to retrieve, mainly because many similar images exist in the collection. Even so, only 6 iterations on average were needed to locate them, while 7 iterations were needed without our index structure [21]. The results illustrate that our index structure can help reduce the number of iterations. The reason is that our index structure is designed to sample as many relevant leaf nodes as possible, and the sampled images can be very representative, which facilitates target search. To evaluate our target search technique in more practical real world scenarios, we conducted experiments on the collection through Google Image Search (randomly chose Corel category descriptions as queries). This collection contains the same number (i.e., 68040) of images as the COREL dataset, and we also randomly picked the same number (i.e., 36) of target images. The results show that 8 iterations on average were needed to locate them. The log information indicates that LNM has been used by most users probably because some of the Google images have low image quality, and are hard to give the accurate relevance feedback. After analyzing the experimental results, we also found out that art students on average took fewer iterations than the engineering ones in both experimental settings (using Corel images, and using Google images), probably because the former are better at recognizing the visual features, and then gave more accurate relevance feedback.

Observe that users' inaccurate response may compromise the benefits of any CBIR systems with relevance feedback. To minimize its effects, we make our system, in design and in implementation, less sensitive to users' inaccurate relevance feedback. First, our prototype (see Figure 24) still keeps LNM as a useful option. This is because LNM is robust against inaccurate relevance feedback as mentioned before, although converging slowly. Based on our observations, users in practice can use GDC or NDC to prune a lot of non-target images at the first few iterations, and use LNM to finally locate the target. Second, in the experimental study, our system monitored users' feedback, and issued warnings in the Status window if inconsistent behaviors were being detected (discussed in Section IV). These warnings prompt the users to re-evaluate their feedback. Finally, our prototype allows users to backtrack their selections if missteps have been made. The results were satisfactory overall, indicated by the successful finding of the intended targets. Of course users' inaccurate relevance feedback is a difficult and open problem but our results are encouraging.

C. Query Processing Technique

In this section, we evaluate the effectiveness of the proposed query processing technique described in Section V.B. The node size of the original R*-tree and our index structure were both set to 4KB, and both had three levels in our experimental

settings. Following the suggestion of the R*-tree [3], the minimum utilization parameter of each node was set to 40% and the reinsert fraction parameter was set to 30% for all index structures. We compare the performance of the proposed query processing technique on top of our index structure (denoted as *QNEW*) against the existing technique with R*-tree (denoted as *QOLD*). Specifically, our methods to evaluate sampling queries, constrained sampling queries, constrained k-NN queries are from line 2 to 9, 10 to 20, and 21 to 38 in Figure 13, respectively. The existing counterparts are proposed in [25], [25] with a straightforward extension by integrating the constraint, and [11], respectively. As mentioned in Section V.B, the new index structure is introduced specifically for our query processing technique. To be fair, we also utilize an index structure (i.e., R*-tree) for the existing counterparts. It is not our claim that the proposed index structure is better than R*-tree. Actually, we claim that our query processing technique coupled with the new index structure outperforms existing ones with R*-tree. We use the number of disk accesses as the main measure of performance to compare *QNEW* and *QOLD*. Sampling queries, constrained sampling queries, and constrained k-NN queries were executed in these experiments; they were randomly generated, and relevance feedback was simulated as in Section VI.A. For constrained sampling queries, $mbb(\mathbb{S}')$ was randomly chosen up to 75% of $mbb(\mathbb{S})$. The dataset and image features were those used in Section VI.A. The results are averaged over 100 runs.

Figure 26 depicts that *QNEW* significantly outperforms *QOLD* for answering sampling queries in terms of disk accesses. For example, *QOLD* performs about 5 times more disk accesses than *QNEW* when $k = 5$, 150 times when $k = 50$, and 300 times when $k = 100$. This figure shows that *QNEW* is independent of the number of sample points (i.e., k) because *QNEW* just needs to access the root node of our index structure, resulting in only one disk access for answering a sampling query. On the other hand, *QOLD* is proportional to k, which is because *QOLD* has to traverse the R*-tree to obtain sample points one by one, incurring almost 3 disk accesses per sample point. Figure 27 compares the performance of both approaches for answering constrained sampling queries. *QNEW* is again independent of k because *QNEW* just needs to perform a modified range query instead of sampling one by one as done by *QOLD*. Although *QOLD* slightly outdoes *QNEW* when k is very small, *QNEW* is superior when $k > 8$ and the performance gap widens as k increases. Specifically, *QOLD* requires about 7 times more disk accesses than *QNEW* when $k = 50$, and almost 15 times when $k = 100$. In feedback iterations, *QNEW* can reduce the cost further as shown in Figure 28 while *QOLD* cannot. The reason is that *QNEW* reuses the visited nodes in the previous iterations, instead of reloading them from the disk. Similarly, *QNEW* significantly outperforms *QOLD* by almost two orders of magnitude for answering constrained k-NN queries in terms of the overall I/O cost, as illustrated in Figure 29.

The performance difference between *QNEW* and *QOLD* confirms that the proposed query processing technique reduce the disk I/O cost significantly by taking advantage of our index structure, reusing the visited nodes in the previous iterations, and pruning non-relevant nodes as early as possible.

VII. Conclusions

In this paper, we proposed four target search methods using relevance feedback for content-based image retrieval systems. Our research was motivated by the observation that revisiting of checked images can cause many drawbacks including local maximum traps and slow convergence. Our methods outperform existing techniques including MARS (employing feature weighting), MindReader (employing complex feature weighting), and Qcluster (employing probabilistic models). All our methods are capable of guaranteeing finding intended target images, with NDC and GDC converging faster than NRS and LNM (which represents an improved version of MindReader). Simulated experiments have shown that NDC and GDC work more efficiently and effectively when k (i.e., the number of allowed returned images) is smaller, and GDC achieving $\mathcal{O}(\log_k |\mathbb{S}|)$ iterations is slightly better than NDC. We also proposed an index structure and efficient query processing technique. Experiments with our prototype show that our approach can achieve fast convergence (i.e. $\mathcal{O}(\log_k |\mathbb{S}|)$ iterations) even in the realistic environments, and is very promising for large CBIR systems.

Acknowledgments

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions on a previous draft.

References

[1] C. B. Barber, D. P. Dobkin, and H. T. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[2] I. Bartolini, P. Ciacci, and F. Waas. Feedbackbypass: A new approach to interactive similarity query processing. In *Proceedings of the 27th VLDB Conference*, pages 201–210, 2001.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the ACM SIGMOD Conference*, pages 322–331, 1990.

[4] C. Böhm, S. Berchtold, and D. A. Keim. Searching in High-dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

[5] P. Browne and A. F. Smeaton. Video Information Retrieval Using Objects and Ostensive Relevance Feedback. In *Proceedings of the ACM Symposium on Applied Computing (SAC)*, pages 1084–1090, 2004.

[6] R. Brunelli and O. Mich. Image retrieval by examples. *IEEE Transactions on Multimedia*, 2(3):164–171, 2000.

[7] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1026–1038, 2002.

[8] K. Chakrabarti, M. Ortega-Binderberger, S. Mehrotra, and K. Porkaew. Evaluating refined queries in top-k retrieval systems. *IEEE Transactions on Knowledge and Data Engineering*, 16(2):256–270, 2004.

[9] L. Chen, M. T. Özsu, and V. Oria. MINDEX: An efficient index structure for salient-object-based queries in video databases. *Multimedia Systems*, 10(1):56–71, 2004.

[10] I. J. Cox, M. L. Miller, T. P. Minka, T. V. Papathomas, and P. N. Yianilos. The Bayesian image retrieval system, PicHunter: theory, implementation, and psychophysical experiments. *IEEE Transactions on Image Processing*, 9(1):20–37, 2000.

[11] H. Ferhatosmanoglu, I. Stanoi, D. Agrawal, and A. E. Abbadi. Constrained nearest neighbor queries. In *Proceedings of the 7th International Symposium on Spatial and Temporal Databases (SSTD)*, pages 257–278, 2001.

[12] M. Flickner, H. S. Sawhney, 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. *IEEE Computer*, 28(9):23–32, 1995.

[13] T. Gevers and A. Smeulders. Content-based image retrieval: An overview. In G. Medioni and S. B. Kang, editors, *Emerging Topics in Computer Vision*. Prentice Hall, 2004.

[14] A. Gupta and R. Jain. Visual information retrieval. *Communications of the ACM*, 40(5):70–79, 1997.

[15] A. Guttman. The R-tree: a Dynamic Index Structure for Spatial Searching. In *Proceedings of the ACM SIGMOD Conference*, pages 47–57, June 1984.

[16] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM Transaction on Database Systems*, 24(2):265–318, 1999.

[17] K. A. Hua, N. Yu, and D. Liu. Query Decomposition: A Multiple Neighborhood Approach to Relevance Feedback Processing in Content-based Image Retrieval. In *Proceedings of the IEEE ICDE Conference*, 2006.

[18] Y. Ishikawa, R. Subramanya, and C. Faloutsos. MindReader: Querying databases through multiple examples. In *Proceedings of the 24th VLDB Conference*, pages 218–227, 1998.

[19] H. V. Jagadish, B. C. Ooi, K.-L. Tan, C. Yu, and R. Zhang. iDistance: An adaptive B+-tree based indexing method for nearest neighbor search. *ACM Transactions on Database Systems*, 30(2):364–397, 2005.

[20] D.-H. Kim and C.-W. Chung. Qcluster: relevance feedback using adaptive clustering for content-based image retrieval. In *Proceedings of the ACM SIGMOD Conference*, pages 599–610, 2003.

[21] D. Liu, K. A. Hua, K. Vu, and N. Yu. Fast Query Point Movement Techniques with Relevance Feedback for Content-based Image Retrieval. In *Proceedings of the 10th EDBT Conference*, pages 700–717, 2006.

[22] W. Y. Ma and B. Manjunath. Netra: a toolbox for navigating large image databases. In *Proceedings of the IEEE International conference on Image Processing*, pages 568–571, 1997.

[23] M. Nakazato, L. Manola, and T. S. Huang. ImageGrouper: a group-oriented user interface for content-based image retrieval and digital image arrangement. *Journal of Visual Languages and Computing*, 14(4):363–386, 2003.

[24] V. Ogle and M. Stonebraker. Chabot: retrieval from a relational database of images. *IEEE Computer*, 28(9):40–48, 1995.

[25] F. Olken and D. Rotem. Sampling from Spatial Databases. In *Proceedings of the IEEE ICDE Conference*, pages 199–208, 1993.

[26] M. Ortega-Binderberger and S. Mehrotra. Relevance feedback techniques in the MARS image retrieval systems. *Multimedia Systems*, 9(6):535–547, 2004.

[27] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: content-based manipulation for image databases. *International Journal of Computer Vision*, 18(3):233–254, 1996.

[28] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York Inc., 1985.

[29] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of the ACM SIGMOD Conference*, pages 71–79, 1995.

[30] Y. Rui, T. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: A power tool for interactive content-based image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644–655, 1998.

[31] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima. The A-tree: An Index Structure for High-Dimensional Spaces Using Relative Approximation. In *Proceedings of the 26th VLDB Conference*, pages 516–526, 2000.

[32] H. T. Shen, B. C. Ooi, and X. Zhou. Towards effective indexing for very large video sequence database. In *Proceedings of the ACM SIGMOD Conference*, pages 730–741, 2005.

[33] J. R. Smith and S.-F. Chang. Transform features for texture classification and discrimination in large image databases. In *Proceedings of the International Conference on Image Processing*, pages 407–411, 1994.

[34] J. R. Smith and S.-F. Chang. VisualSEEk: A fully automated content-based image query system. In *Proceedings of the 4th ACM Multimedia Conference*, pages 87–98, 1996.

[35] M. A. Stricker and M. Orengo. Similarity of color images. In *Proceedings of Storage and Retrieval for Image and Video Databases (SPIE)*, pages 381–392, 1995.

[36] S. Tong and E. Y. Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ACM Multimedia Conference*, pages 107–118, 2001.

[37] K. Vu, K. A. Hua, H. Cheng, and S.-D. Lang. A non-linear dimensionality-reduction technique for fast similarity search in large databases. In *Proceedings of the ACM SIGMOD Conference*, pages 527–538, 2006.

[38] K. Vu, K. A. Hua, and W. Tavanapong. Image retrieval based on regions of interest. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):1045–1049, 2003.

[39] J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):947–963, 2001.

[40] L. Wu, C. Faloutsos, K. Sycara, and T. R. Payne. FALCON: feedback adaptive loop for content-based retrieval. In *Proceedings of the 26th VLDB Conference*, pages 297–306, 2000.

[41] Y. L. Wu, D. Agrawal, and A. E. Abbadi. Applying the golden rule of sampling for query estimation. In *Proceedings of the ACM SIGMOD Conference*, pages 449–460, 2001.

[42] Y. L. Wu, D. Agrawal, and A. E. Abbadi. Query estimation by adaptive sampling. In *Proceedings of the IEEE ICDE Conference*, pages 639–648, 2002.

[43] W. Yuan and K. Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. In *Proceedings of the ACM Symposium on Operating Systems Principles*, pages 149–163, 2003.

[44] X. S. Zhou and T. S. Huang. Edge-based structural features for content-based image retrieval. *Pattern Recognition Letters*, 22(5):457–468, 2001.

Danzhou Liu received the BE degree in information engineering from the Beijing University of Aeronautics and Astronautics in 1994, the ME degree in computer engineering from the Nanyang Technological University in 2002, and the MS degree in computer science from the University of Central Florida in 2005. He is currently a PhD candidate in the Data System Group at the University of Central Florida. His research interests include multimedia retrieval, multimedia communications, data mining, and machine learning. He received the 2007 IEEE Orlando Section Outstanding Graduate Student Award. He is a member of the IEEE.

Kien A. Hua received the BS degree in computer science and the MS and PhD degrees in electrical engineering, all from the University of Illinois at Urbana-Champaign, in 1982, 1984, and 1987, respectively. From 1987 to 1990, he was with the IBM Corporation. He joined the University of Central Florida in 1990 and is currently a professor in the School of Electrical Engineering and Computer Science. He served as the Interim Associate Dean for Research of the College of Engineering and Computer Science at the University of Central Florida from 2004 to 2006. Dr. Hua has published widely, including several papers recognized as best/top papers at various international conferences. He has served as a general chair, vice-chair, associate chair, demo chair, and program committee member for numerous conferences. He was an associate editor of the *IEEE Transactions on Knowledge and Data Engineering* from 2001 to 2005. He is currently serving on the editorial board of the *Journal of Multimedia Tools and Applications* and the *International Journal of Advanced Information Technology*. He is a senior member of the IEEE.

Khanh Vu received his BS and PhD degrees in Computer Science from the University of Central Florida in 1995 and 2002, respectably. After a brief time with Oklahoma State University, he has been working at University of Central Florida as a Research Associate. His research interest includes Information Retrieval, Databases, Image/Video Processing, and Networking.

Ning Yu received the BS degree in computer science from the Tsinghua University in 2000, and the MS degree in computer science from the University of Central Florida in 2005. She is currently a PhD candidate in the Data System Group at the University of Central Florida. Her research interests include indexing and relevance feedback retrieval in multimedia database, affective computing and human-computer interface, and medical image processing and applications. She is a member of the IEEE.