Supporting Ranked Boolean Similarity Queries in MARS

Michael Ortega, *Student Member, IEEE*, Yong Rui,
Kaushik Chakrabarti, Kriengkrai Porkaew,
Sharad Mehrotra, *Member, IEEE*, and Thomas S. Huang, *Fellow, IEEE*

Abstract—To address the emerging needs of applications that require access to and retrieval of multimedia objects, we are developing the *Multimedia Analysis and Retrieval System* (MARS) [29]. In this paper, we concentrate on the retrieval subsystem of MARS and its support for content-based queries over image databases. Content-based retrieval techniques have been extensively studied for textual documents in the area of automatic information retrieval [40], [4]. This paper describes how these techniques can be adapted for ranked retrieval over image databases. Specifically, we discuss the ranking and retrieval algorithms developed in MARS based on the Boolean retrieval model and describe the results of our experiments that demonstrate the effectiveness of the developed model for image retrieval.

Index Terms—Database management, multimedia retrieval, Boolean queries, incremental query processing, ranked retrieval.

1 Introduction

While advances in technology allow us to generate, transmit, and store large amounts of digital images and video, research in content-based retrieval over multimedia databases is still at its infancy. Due to the difficulty in capturing the content of multimedia objects using textual annotations and the nonscalability of the approach to large data sets (due to a high degree of manual effort required in defining annotations), the approach based on supporting content-based retrieval over visual features has become a promising research direction. This is evidenced by several prototypes [42], [33], [28], [29] and commercial systems [16], [1], built recently. Such an approach can be summarized as follows:

1) Computer vision techniques are used to extract visual features from multimedia objects. Examples of visual features are: color, texture, and shape for images, and motion parameters for video.
2) For a given feature, a representation of the feature and a notion of similarity between instances of the feature are determined. For example, color histogram can be used to represent the color feature, and the intersection similarity (defined in Section 2) used to compute the similarity among color histograms. More than one representation is possible for a given feature.
3) Objects are represented as a collection of features and retrieval of objects is performed based on computing similarity in the feature space. The results are ranked based on the computed similarity values.

Since automatically extracted visual features (e.g., color, texture etc.) are too low level to be useful to the user's in specifying their information needs directly, content-based retrieval using visual features requires development of effective techniques to map higher-level user queries (e.g., retrieve images containing a field of yellow flowers) to visual features. Mapping a user's information need to a set of features extracted from textual documents has been extensively studied in the information retrieval literature [40]. This article describes how we have generalized these approaches for content-based retrieval over image features in the *Multimedia Analysis and Retrieval System* (MARS). An overview of the system architecture is shown in Fig. 1.

1.1 Information Retrieval Models

Before we describe the retrieval approach used in MARS, we briefly review the retrieval process in modern information retrieval (IR) systems [40]. In an IR system, a document is represented as a collection of features (also referred to as terms). Examples of features include words in a document, citations, bibliographic references, etc. A user specifies his information need to the system in the form of a query. Given a representation of the user's information need and a document collection, the IR system estimates the likelihood that a given document matches the user's information need. The representation of documents and queries, and the metrics used to compute the similarity among them constitute the *retrieval model* of the system. Existing retrieval models can be broadly classified into the following categories:

• *M. Ortega, Y. Rui, K. Chakrabarti, K. Porkaew, and T.S. Huang are with the Department of Computer Science and the Beckman Institute, University of Illinois at Urbana-Champaign, Urbana, IL 61801. E-mail: ortega-b@uiuc.edu.*
• *S. Mehrotra is with the Department of Information and Computer Science, University of California at Irvine, Irvine, CA 92697, and is also affiliated with the University of Illinois at Urbana-Champaign.*

Manuscript received 15 November 1997; revised 7 July 1998.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 107288.

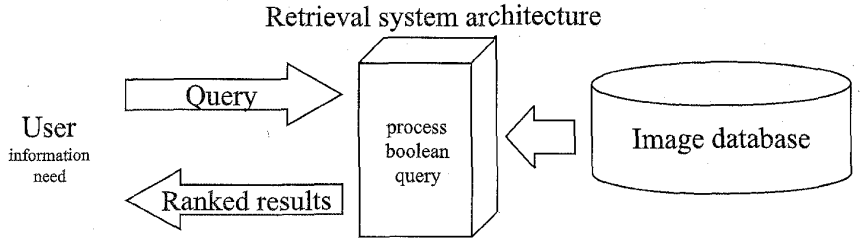


Fig. 1. Overall system architecture.

- **Boolean Models**: Let $\{r_1, r_2, \ldots, r_k\}$ be the set of terms in a collection. Each document is represented as a binary-valued vector of length k where the ith element of the vector is assigned *true* if r_i is assigned to the document. All elements corresponding to features / terms not assigned to a document are set to *false*. A query is a Boolean expression in which operands are terms. A document whose set of terms satisfies the Boolean expression is deemed to be relevant to the user and all other documents are considered not relevant.
- **Vector-Based Models**: Let $\{r_1, r_2, \ldots, r_k\}$ be the set of terms in a collection. Both documents and queries are represented as a vector of k dimensions where each element in the vector corresponds to a real-valued weight assigned to a term. Several techniques have been proposed to compute these weights, the most common being $tf \times idf$ weights [40], where tf refers to the term frequency in the document, and idf is a measure proportional to the inverse of its frequency in the collection. Also, many similarity measures between the document and the query have been proposed [40], the most common being the cosine of the angle between the document and the query vectors.
- **Probabilistic Retrieval Models**: In these models the system estimates the probability of relevance of a document to the user's information need specified as a query. Documents are ranked in decreasing order of relevance estimate. Given a document and a query, the system computes $P(R \mid d, q)$ which represents the probability that the document d will be deemed relevant to the user's information need expressed as the query q. These probabilities are computed and used to rank the documents using Bayes' theorem and a set of independence assumptions about the distribution of terms in the documents.

Traditionally, commercial IR systems have used the Boolean model. Systems based on Boolean retrieval partition the set of documents into either being relevant or not relevant and do not provide any estimate as to the relative importance of documents in a partition to the user's information need. To overcome this problem, many variations of the term-weighting and probabilistic retrieval models that provide ranked retrieval have been proposed. The Boolean model has also been extended to allow for ranked retrieval in the text domain (e.g., the p-norm model [39]). Vector-based models and probabilistic retrieval models are in a sense related and provide comparable performance.

1.2 Overview of the Retrieval Approach Used in MARS

With the large number of retrieval models proposed in the IR literature, MARS attempts to exploit this research for content-based retrieval over images. An image is represented as a collection of low-level image features (e.g., color, texture, shape, and layout) extracted automatically using computer vision methods, as well as a manual text description of the image. A user graphically constructs a query by selecting certain images from the collection. A user may choose specific features from the selected images. For example, using a point-and-click interface a user can specify a query to retrieve images similar to an image A in color and similar to an image B in texture. A user's query is then interpreted as a Boolean expression over image features. A Boolean retrieval model (adapted for retrieval over images) is used to interpret the query and retrieve a set of images ranked based on their similarity to the selected feature. Boolean queries provide a natural interface for the user to formulate and refine *conceptual queries* to the system using lower-level image features. For example, high level concepts like fields of yellow flowers or a sunset by a lake can be expressed as a Boolean combination of lower level features. Such a mapping of high to low level concepts can be provided explicitly by the user or alternatively learned via user interaction by a relevance feedback mechanism [35], [37], [36]. Being able to support such conceptual queries is critical for the versatility of large image databases.

To see how MARS adapts the Boolean model for image retrieval, consider first a query Q over a single feature F_i (say color represented as a color histogram). Let $H(I)$ be the color histogram of image I and $H(Q)$ be the color histogram specified in the query and $similarity(H(I), H(Q))$ be the similarity between the two histograms. Similarity values are in the range [0, 1] with 1 being the best and 0 the worst. The simplest way to adapt the Boolean model for image retrieval is to associate a *degree of tolerance* δ_i with each feature F_i such that:

$$I \text{ matches } Q = true, \text{ if } similarity(H(I), H(Q)) \geq \delta_i$$
$$= false, \text{ if } similarity(H(I), H(Q)) < \delta_i$$

Given the above interpretation of a match based on a single feature F_i, an image I matches a given query Q if it satisfies the Boolean expression associated with Q. For example, let $Q = v_1 \wedge v_2$, where v_1 is a color histogram, and v_2 is a texture representation. Image I matches Q if its color and texture representations are within the specified tolerances of v_1 and v_2.

Although the above straightforward adaptation of Boolean retrieval can be used for retrieval in MARS, it has several potential problems. First, it is not clear how the degree of tolerance δ_i, for a given feature F_i, should be determined. If an a priori value is set for δ_i, it may result in poor performance—two images I_1 and I_2 at similarity of $\delta_i + \epsilon$ and $\delta_i - \epsilon$ from a query Q, where $\epsilon \to 0$, are very similar as far as their relevance to Q is concerned but would be considered as very different by the system. While I_1 would be considered relevant to the query, I_2 would not be considered as relevant. This problem may be alleviated by dynamically computing δ_i for each query based on the image collection instead of using fixed a priori tolerance values for a given feature. However, the approach still suffers from the fundamental restriction of the basic Boolean retrieval in that it produces an unranked set of answers.

To overcome the above discussed problems, we have adopted the following two extensions to the basic Boolean model to produce a ranked list of answers:

- **Fuzzy Boolean Retrieval**: The similarity between the image and the query feature is interpreted as the degree of membership of the image to the fuzzy set of images that match the query feature. Fuzzy set theory is used to interpret the Boolean query and the images are ranked based on their degree of membership in the set.
- **Probabilistic Boolean Retrieval**: The similarity between the image and the query feature is considered to be the probability that the image matches the user's information need. Feature independence is exploited to compute the probability of an image satisfying the query which is used to rank the images.

Unlike the basic Boolean model, both the fuzzy and probabilistic Boolean models provide ranked retrieval over the image collection.

The rest of the paper is developed as follows. In Section 2, we describe the set of image features used in MARS and the techniques used to measure the similarity between images based on individual features. Section 3 discusses the techniques to normalize the low level features necessary to combine them with each other. Section 4 describes the Boolean retrieval models used and discusses issues related to their efficient implementation. Section 5 presents the experimental results demonstrating the retrieval effectiveness of the developed models. Section 6 describes the related work. Finally, Section 7 offers concluding remarks and future work.

2 Image Features Used in MARS

The retrieval performance of an image database is inherently limited by the nature and the quality of the features used to represent the image content. In this section, we briefly describe the image features used and the corresponding similarity functions for comparing images based on these features. The discussion is kept brief since the purpose of this section is only to provide a background for discussing issues related to normalization and ranked retrieval based on Boolean queries. Detailed discussion on the rationale and the quality of the chosen features can be found in [14], [48], [29], [31], [38]. The following features and their representation only describe features currently supported. The system allows for other features to also be incorporated:

- **Color Features**: The color feature is one of the most widely used visual features in image retrieval. Many approaches to color representation, such as color histogram [47], color moments [46], color sets [44], have been proposed in the past few years. In this paper we chose the color histogram approach in the Hue, Saturation, Value (HSV) color space. The Red, Green, Blue (RGB) color space used for image storage is transformed into the HSV space where Hue and Saturation are polar coordinates indicating the tint (Hue) of the color and the intensity of the pigmentation (Saturation). The Value coordinate refers to the brightness of the color. The precise method of transformation is described in [17]. We chose the HSV color space for our color feature since:
  - ✓ the color histogram is easy to extract and its similarity is fast to compute; and
  - ✓ the HSV color space has decorrelated and uniform coordinates, better matching the human perception of color.

  Furthermore, since the V coordinate in the HSV space is easily affected by the lighting condition, we only use the HS coordinates to form a two-dimensional histogram. The H and S dimensions are divided into N and M bins, respectively, for a total of $N \times M$ bins. We chose $N = 8$, $M = 4$ for our tests. Each bin contains the percentage of pixels in the image that have corresponding H and S colors for that bin; note that the sum of all bins equals one. More details can be found in [47].

  To measure the similarity between two color histograms, we use the *intersection similarity* which captures the amount of overlap between the two histograms:

$$similarity_{color} = \sum_{i=1}^{i=N} \sum_{j=1}^{j=M} min\big(H_1(i,j), H_2(i,j)\big) \quad (1)$$

  where H_1 and H_2 are the two histograms; and N and M are the number of bins along the H and S coordinates. Since the histograms are normalized to add to one, this measure ranges from zero (not similar) to one (identical). The above intersection-based measure of similarity provides an accurate and efficient measure of similarity between two images based on their color [42].

- **Texture Features**: Texture refers to the visual patterns that have properties of homogeneity that do not result from the presence of only a single color or intensity [45]. It is an innate property of virtually all surfaces, including clouds, trees, bricks, hair, fabric, etc. Texture contains important information about the structural arrangement of surfaces and their relationship to the surrounding environment [23]. Because of its importance

and usefulness in pattern recognition and computer vision, extensive research has been conducted on texture representation in the past three decades, including the co-occurrence matrix-based representation [23], Tamura et al. texture representation [49], and wavelet-based representation [43], [6], [26], [20], [25], [50]. Many research results have shown that the wavelet-based texture representation achieves good performance in texture classification [43]. Therefore, we chose the wavelet approach for texture representation. In this approach, an input image is fed into a wavelet filter bank and is decomposed into decorrelated subbands. Due to the orthogonality of the wavelet decomposition, each subband captures the property of some scale and orientation of the original image. A wavelet decomposition of an image results in four quadrants. This decomposition is recursively applied to the quadrant closest to the origin. Specifically, we decompose an image into three wavelet levels; thus having 10 subbands; three each for the first two levels and four for the third level [6]. For each subband, we extract the standard deviation of the wavelet coefficients giving a 10-dimensional vector that represents the texture.

The similarity between two texture feature vectors is defined as the Euclidean distance in this 10-dimensional feature space. To convert this distance in a 10-dimensional space to a similarity value, refer to Section 3.

- **Shape Features**: Shape of an object in an image is represented by its boundary. A technique for storing the boundary of an object using a modified Fourier descriptor (MFD) is described in [38]. The Euclidean distance can be used to measure similarity between two shapes. Rui et al. [38] proposes a similarity measure based on standard deviation that performs significantly better compared to the simple Euclidean distance. The proposed representation and similarity measure provide invariance to translation, rotation, and scaling of shapes, as well as the starting point used in defining the boundary sequence.
- **Color Layout Features**: Although the global color feature is simple to calculate and can provide reasonable discriminating power in retrieval, it tends to give too many false alarms when the image collection is large. Many research results suggested that using color layout (both color feature and spatial relations) is a better solution. To extract the color layout, the whole image is first split into $k \times k$ subimages. Then the 2D color histograms are extracted from each subimage, similar to the procedure described earlier.

  The similarity between two images in terms of color layout feature is then defined as the average of the similarities of each subimage.
- **Textual Annotation Features**: In addition to its visual content, each image may contain a textual description. This may come in the form of an image caption, a museum description or closed caption decoding in video frames and can be manually added to the image. In our model, we use a vector space representation with a cosine similarity measure to support this feature.

In our model, incorporating a new feature is simple. As will become clear, as long as all feature evaluation modules conform to a consistent interface, the addition of a module is almost instantaneous. Other image features are available, however we restrict ourselves to queries involving only the above features in this paper.

3 Feature Sequence Normalization

Depending on the extracted feature, some normalization may be needed. The normalization process serves two purposes:

1) It puts an equal emphasis on each feature element within a feature vector. To see the importance of this, notice that in the texture representation, the feature elements may be different physical quantities. Their magnitudes can vary drastically, thereby biasing the Euclidean distance measure. This is overcome by the process of *intra-feature* normalization.
2) It maps the distance values of the query from each atomic feature into the range [0, 1] so that they can be interpreted as the degree of membership in the fuzzy model or relevance probability in the probability model. While some similarity functions naturally return a value in the range of [0, 1], e.g., the color histogram intersection; others do not, e.g., the Euclidean distance used in texture. In the latter case the distances need to be converted to the range of [0, 1] before they can be used. This is referred to as *inter-feature normalization.*

3.1 Intra-Feature Normalization

This normalization process is only needed for features using a *vector-based* representation, as in the case of the wavelet texture feature representation.

For the vector-based feature representation, let $F = [f_1, f_2, \ldots, f_j, \ldots, f_N]$ be the feature vector, where N is the number of feature elements in the feature vector and $I_1, I_2, \ldots, I_M$ be the images. For image I_i, we refer to the corresponding feature F as $F_i = [f_{i,1}, f_{i,2}, \ldots, f_{i,j}, \ldots, f_{i,N}]$. Since there are M images in the database, we can form a $M \times N$ feature matrix $F = f_{i,j}$, where $f_{i,j}$ is the jth feature element in feature vector F_i. Each column of F is a length-M sequence of the jth feature element, represented as F_j. The goal is to normalize the entries in each column to the same range so as to ensure that each individual feature element receives equal weight in determining the Euclidean distance between the two vectors.

Assuming the feature sequence F_j to be a Gaussian sequence, we compute the mean m_j and standard deviation σ_j of the sequence. We then normalize the original sequence to a N(0, 1) sequence as follows:

$$f'_{i,j} = \frac{f_{i,j} - m_j}{\sigma_j} \tag{2}$$

Note that after the Gaussian normalization, the probability of a feature element value being in the range of [–1, 1] is 68 percent. If we use $3\sigma_j$ in the denominator, the probability of a feature element value being in the range of [–1, 1] is approximately 99 percent. In practice, we can consider all of the feature element values within the range of [–1, 1] by mapping the out-of-range values to either –1 or 1.

3.2 Inter-Feature Normalization

Intra-feature normalization ensures that equal emphasis is put on each feature element within a feature vector. On the other hand, inter-feature normalization ensures equal emphasis of each feature within a composite query. The aim is to convert similarity values (or distance in some cases like wavelet) into the range [0, 1].

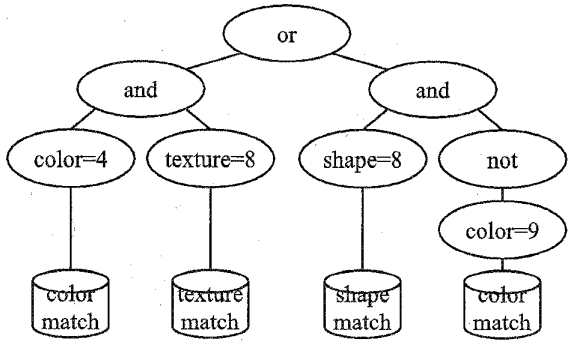
The feature representations used in MARS are of various forms, such as vector-based (wavelet texture representation), histogram-based (histogram color representation), irregular (MFD shape representation), etc. The distance computations of some of these features (e.g., color histogram) naturally yield a similarity value between 0 and 1 and hence do not need additional normalization. Distance calculations in other features are normalized to produce values in the range [0, 1] with the process described below.

1) For any pair of images I_i and I_j, compute the distance $D_{(i,j)}$ between them:
$$D_{(i,j)} = dist(F_{I_i}, F_{I_j}) \quad i, j = 1, ..., M, \; i \neq j \tag{3}$$
where F_{I_i} and F_{I_j} are the feature representations of images I_i and I_j.
2) For the
$$C_2^M = \frac{M \times (M-1)}{2}$$
possible distance values between any pair of images, treat them as a value sequence and find the mean m and standard deviation σ of the sequence. Store m and σ in the database to be used in later normalization.
3) After a query Q is presented, compute the raw (unnormalized) distance value between Q and the images in the database. Let $s_1, ..., s_M$ denote the raw distance values.
4) Normalize the raw distance values as follows:
$$s_i' = \frac{s_i - m}{3\sigma} \tag{4}$$
As explained in the intra-feature normalization section, this Gaussian normalization will ensure 99 percent of s_i' to be within the range of [–1, 1]. An additional shift will guarantee that 99 percent of distance values are within [0, 1]:
$$s_i'' = \frac{s_i' + 1}{2} \tag{5}$$
After this shift, in practice, we can consider all the values within the range of [0, 1], since an image whose distance from the query is greater than 1 is very dissimilar and can be considered to be at a distance of 1 without affecting retrieval.
5) Convert from distance values into similarity values. This can be accomplished by the following operation:
$$similarity_i = 1 - s_i'' \tag{6}$$
At the end of this normalization, all similarity values for all features have been normalized to the same range [0, 1] with the following interpretation: 1 means full similarity (exact match) and 0 denotes maximum dissimilarity.

4 Retrieval Models Used In MARS

This section discusses how we support Boolean queries based on the simple feature similarity values. We support two mechanisms for generating the ranking of Boolean queries—the first is based on a fuzzy interpretation of similarity and the second is based on a probabilistic interpretation. In the discussion below, we will use the following notation. Images in the collection are denoted by $I_1, I_2, ..., I_m$. Features over the images are denoted by $F_1, F_2, ..., F_r$, where F_i denotes both the name of the feature as well as the domain of values that the feature can take. The jth instance of feature F_i corresponds to image I_j and is denoted by f_{ij}. For example, say F_1 is the color feature which is represented in the database using an HS histogram. In that case, F_1 is also used to denote the set of all the color histograms, and $f_{1,5}$ is the color histogram for image 5. Query variables are denoted by $v_1, v_2, ..., v_n \mid v_k \in F_i$ so each v_k refers to an instance of a feature F_i (an f_{ij}). Note that $F_i(I_j) = f_{ij}$. During query evaluation, each v_k is used to rank images in the collection based on the feature domain of $f_i(F_i)$, that is v_k's domain. Thus, v_k can be thought of being a list of images from the collection ranked based on the similarity of v_k to all instances of F_i. For example, say F_2 is the set of all wavelet texture vectors in the collection, if $v_k = f_{2,5}$, then v_k can be interpreted as being both, the wavelet texture vector corresponding to image 5 and the ranked list of all
$$\left\langle I, S_{F_2}\left(F_2(I), f_{2,5}\right)\right\rangle$$
with S_{F_2} being the similarity function that applies to two texture values. A query $Q(v_1, v_2, ..., v_n)$ is viewed as a query tree whose leaves correspond to single feature variable queries. Internal nodes of the tree correspond to the Boolean operators. Specifically, nonleaf nodes are of one of three forms: $\wedge(v_1, v_2, ..., v_n)$, a conjunction of positive literals; $\wedge(v_1, v_2, ..., v_p, \neg v_{p+1} ..., \neg v_n)$, a conjunction consisting of both positive and negative literals; and $\vee(v_1, v_2, ..., v_n)$, which is a disjunction of positive literals. Notice that we do not consider an unguarded negation or a negation in the disjunction (that is, $p \geq 1$), since it does not make much sense. Typically, a very large number of entries will satisfy a negation query virtually producing the universe of the collection [3]. We therefore allow negation only when

it appears within a conjunctive query to rank an entry on the positive feature discriminated by the negated feature. The following is an example of a Boolean query: $Q(v_1, v_2) = (v_1 = f_{1,5}) \wedge (v_2 = f_{2,6})$ is a query where v_1 has a value equal to the color histogram associated with image I_5 and v_2 has a value of the texture feature associated with I_6. Thus, the query Q represents the desire to retrieve images whose color matches that of image I_5 *and* whose texture matches that of image I_6. Fig. 2 shows an example query $Q(v_1, v_2, v_3, v_4) = ((v_1 = f_{1,4}) \wedge (v_2 = f_{2,8})) \vee ((v_3 = f_{3,8}) \wedge \neg(v_4 = f_{1,9}))$ in its tree representation.



Operators: And, Or, Not
Basic features and representations:
Color histogram, color moment, wavelet texture, ...

Fig. 2. Sample query tree.

4.1 Finding the Best N Matches

While the Boolean retrieval model provides a mechanism for computing a similarity of match for all images given a query, for the approach to be useful, techniques must be developed to retrieve the best N matches efficiently without having to rank each image. Such a technique consists of two steps:

- Retrieve images in rank order based on each feature variable v_i in the query.
- Combine the results of the single feature variable queries to generate a ranked retrieval for the entire query.

The first step is discussed in Section 4.3. The second step is elaborated upon in Section 4.4 for the background, and in Sections 4.5 and 4.6 for the fuzzy model and Sections 4.7 and 4.8 for the probabilistic model.

Once efficient ranked retrieval based on a single feature has been achieved, the ranked lists are *normalized* and then the normalized ranked lists are merged into a ranked set of images corresponding to a query. The normalization process used was described in Section 3. To merge the normalized ranked lists, a query $Q(v_1, v_2, \ldots, v_n)$ is viewed as a query tree whose leaves correspond to single feature variable queries and the internal nodes correspond to Boolean operators. The query tree is evaluated as a *pipeline* from the leaves to the root. Each node in the tree exposes to its parent a ranked list of images, where the ranking corresponds to the degree of membership (in the fuzzy model), or the measure of probability (in the probabilistic model). We say it exposes, because the key point is that each node produces the next best result strictly *on demand*. For example, in the fuzzy model, a node N in a tree exposes to its parent a ranked list of

$$\left\langle I, similarity_{Q_N}(I) \right\rangle$$

where Q_N corresponds to the query associated with the subtree rooted at node N. The algorithms used to create the nodes ranked list of images from its children depend upon the retrieval model used.

4.2 Weighting in the Query Tree

In a query, one feature can receive more importance than another according to the user's perception. The user can assign the desired importance to any feature by a process known as *feature weighting*. Traditionally, retrieval systems [16], [1] use a linear scaling factor as feature weights. Under our Boolean model, this is not desirable. It has been noted [13] that such linear weights do not scale to arbitrary functions used to compute the combined similarity of an image. The reason is that the similarity computation for a node in a query tree may be based on operators other than a weighted summation of the similarity of the children. For example if the fuzzy model is used, and the node is $\wedge$, the similarity computation is done as

$$similarity_{\wedge} = \min(\alpha \times S_{F_i}, \beta \times S_{F_j})$$

If F_i carries a weight α, F_j a weight β and the above method is used, then

$$similarity_{\wedge} = \min(\alpha \times S_{F_i}, \beta \times S_{F_j})$$

will be in the range $[0, \min(\alpha, \beta)]$ which is distinct from $[0, 1]$ in general. Fagin and Wimmers [13] present a way to extend linear weighting to the different components for arbitrary scoring functions as long as they satisfy certain properties. We are unable to use their approach since their mapping does not preserve orthogonality properties on which our later algorithms rely. Instead, we use a mapping function from $[0, 1] \rightarrow [0, 1]$ of the form

$$similarity' = similarity^{\frac{1}{weight}}, \quad 0 < w < \infty \qquad (7)$$

which preserves the range boundaries [0, 1] and boosts or degrades the similarity in a smooth way. Sample mappings are shown in Fig. 3. This method preserves most of the properties explained in [13], except it is undefined for a weight of 0. In [13], a weight of 0 means the node can be dismissed. Here, $\lim_{weight \rightarrow 0} similarity' = 0$ for $similarity \in [0, 1)$. A perfect similarity of 1 will remain at 1. This mapping is performed at each link connecting a child to a parent in the query tree.

4.3 Leaf Node Evaluation

Each leaf node in the query tree corresponds to a selection operation on a single feature. For example, in Fig. 2, the leaf nodes correspond to selection operations based on color, texture and shape features (the selection predicates being

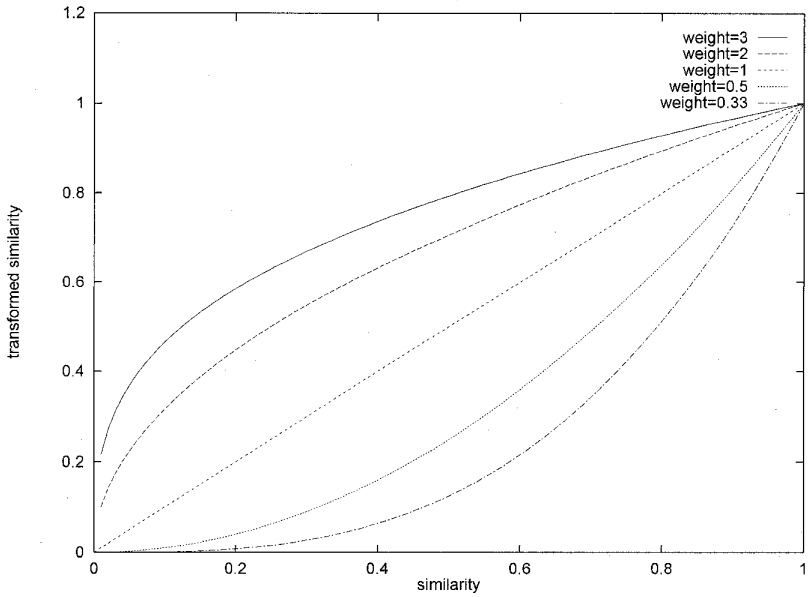


Fig. 3. Various samples for similarity mappings.

color = 4, *texture* = 8, *shape* = 8, etc.). This selection corresponds to ranking the collection of vectors based on their similarity to the query vector. A selection operation has a query feature vector F_Q and a similarity (or distance) function S as arguments and *iteratively* returns the image whose corresponding feature vector next best matches the given query vector F_Q. A simple way to implement the selection operation is a sequential file scan over the collection of feature vectors. However, the I/O cost of the sequential scan operation increases linearly with the size of the feature database and hence may be expensive for large databases. The efficiency of the leaf node evaluation can be improved by using appropriate indexing mechanisms that support nearest neighbor search over multidimensional feature vectors. Several indexing mechanism suited for multimedia features (referred to as the *F-index* or the feature index [15]) have been proposed recently (e.g., R-trees [22], R+-trees [41], R*-trees [2], k-d-B-trees [34], hB-trees [11], TV-trees [27], SS-trees [52], vp-trees [8], M-trees [9]). Any such indexing mechanism can be used for indexing the feature vectors. We have developed a hybrid multidimenional index mechanism that combines the advantages of the two broad classes of multidimensional index structures, namely the space partitioning (kd-tree-based) index structures and the object clustering (bounding region-based) ones. The hybrid data structure scales very well to high dimensionality, large database sizes and arbitrary distance measures. The hybrid tree provides efficient support for range queries as well as nearest neighbor queries on the indexed feature based on any distance function specified by the user [5]. For example, MARS currently uses the intersection distance for color histograms and the Euclidean distance for the texture feature but more complex distance measures can be supported. In this paper, we concentrate on developing techniques for evaluating Boolean query nodes and hence do not elaborate on indexing techniques to improve leaf node evaluation any further. In the rest of the paper, we assume the presence of appropriate indexing mechanisms which provide efficient support for nearest neighbor search over multidimensional data and hence ranked retrieval at the leaf nodes.

4.4 Background on Evaluation Algorithms

This section defines some background concepts to be used in the following sections. As described above, a Boolean query produces a ranked list of $\langle I, similarity_Q(I)\rangle$ based on the similarity of each image I to the query Q. Our evaluation model for the rest of the paper is as follows:

- Each node N defines a subquery rooted at node N denoted by Q_N.
- Each node N returns a list of $\rho_i = \left\langle I_j, similarity^i_{Q_N}(I_j)\right\rangle$ to its parent where:
  - ✓ $i = 1, 2, \ldots, n$ is the sequence number in which the ρs are returned and n is the number of images in the collection.
  - ✓ j is an image number (id) and is unrelated to i.
  - ✓ Q_N is the query subtree rooted at node N.
  - ✓ $similarity^i_{Q_N}(I_j)$ is the similarity value of image j to the subquery rooted at Q_N.
  - ✓ for any two

$$\rho_i = \left\langle I_j, similarity^i_{Q_N}(I_j)\right\rangle$$

and

$$\rho_k = \left\langle I_{j'}, similarity^k_{Q_N}(I_{j'})\right\rangle$$

if $i < k$ then

$$similarity^i_{Q_N}(I_j) \geq similarity^k_{Q_N}(I_{j'})$$

holds. That is, any ρ returned as an answer for the subquery Q_N will have higher similarity than any pair returned later for the same subquery. In other words, ρs are returned in sorted order by similarity.

- Evaluation of a subquery rooted at Q_N produces a sequence of ρ's. A cursor is maintained in this sequence to support the concept of current element; this sequence with cursor is called a *stream*.
- The notion of *best element* of a stream at any point is defined as the next

$$\rho_i = \left\langle I_j, similarity_{Q_N}(I_j) \right\rangle$$

that would be obtained from a stream satisfying the above criteria.

- A *stream* of ρs will support the operations
  - ✓ *PeekNext* (or just *Peek*) that returns the *best element* of the stream without removing it from the stream.
  - ✓ *GetNext* that returns the *best element* of the stream and removes it from the stream.
  - ✓ *Probe*(I_j, Q_N) that performs random access to image j and returns a

$$\rho_i = \left\langle I_j, similarity_{Q_N}(I_j) \right\rangle$$

that is, the image id and similarity pair corresponding to image j based on the subquery Q_N. Not all operators will require this support, we however define it for all as a convenience.

Our Boolean model defines operators that work on such streams. The algorithms defined in the following sections assume binary operators. n-ary operators can be implemented by either nesting binary operators (using the associativity property) or extending the algorithms to cope with n input streams. Extension of binary to n-ary operators is straightforward in all cases.

Given that the operators discussed are binary, and the inputs are streams as defined above, we can create a two-dimensional representation where each axis corresponds to similarity values from one stream. Fig. 4 depicts such a scenario. In this figure, the horizontal axis corresponds to stream A and the vertical axis to stream B. Points on this graph correspond to images whose similarity in stream A defines its A-axis coordinate and the similarity in B defines its B-axis coordinate. For instance, the point shown corresponds to image I with similarity values a' and b' in the respective streams.

Since streams are traversed in rank order of similarity, we obtain coordinates in sorted order from each stream. In the figure, a and b show the current similarity values of the best element currently in the streams (the cursor contents). Since all images from stream A with similarity values in the range $[a, 1]$ and all from stream B with similarity values in the range $[b, 1]$ have been read already, we can construct a rectangle bounded by the points (a, b) and (1, 1) such that for all images in the rectangle, the similarity values

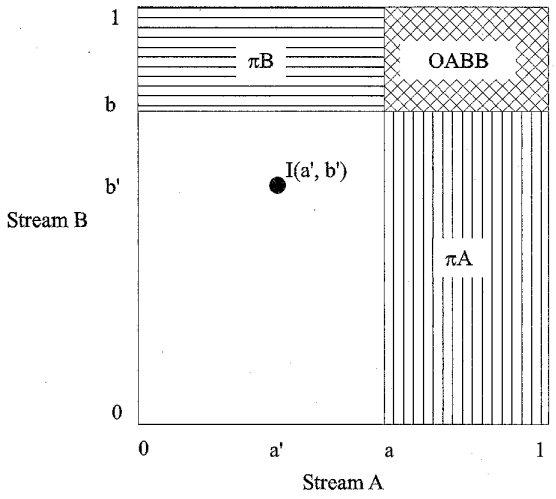


Fig. 4. A sample OABB rectangle.

corresponding to both streams have been observed. We refer to this rectangle as the *Observed Area Bounded Box* (OABB). Another interpretation of OABB is that it is the current intersection of the images observed so far in both streams. Projecting OABB onto the A axis yields another rectangle (called πA) that contains only images whose A coordinate is known, but its b coordinate is unknown; OABB and πA do not overlap. The same is true for the projection of OABB onto the B axis (the rectangle is called πB). $\pi A \cup \pi B$ denotes the images of which we have partial knowledge of their location in this 2D space (i.e., exactly one coordinate known). Thus any image of which we have complete knowledge (both similarity values seen) must lie in OABB.

The following sections make use of these definitions to explain the functioning of the algorithms.

4.5 Fuzzy Boolean Model

Let $Q(v_1, v_2, \ldots, v_n)$ be a query and I be an image. In the fuzzy retrieval model, a query variable v_i is considered to be a fuzzy set of images and the relevance of any image I to Q with respect to v_i is interpreted as the degree of membership of I in that fuzzy set.

With the above interpretation of the similarity measure between the image feature and the feature specified in the query, a Boolean query Q is interpreted as an expression in fuzzy logic and fuzzy set theory is used to compute the degree of membership of an image to the fuzzy set represented by the query Q. Specifically, the degree of membership for a query Q is computed as follows:

$$\begin{aligned} \textbf{And} \quad & S_{Q=Q_1 \wedge Q_2}(I) = \min\left(S_{Q_1}(I), S_{Q_2}(I)\right) \\ \textbf{Or} \quad & S_{Q=Q_1 \vee Q_2}(I) = \max\left(S_{Q_1}(I), S_{Q_2}(I)\right) \\ \textbf{Not} \quad & S_{Q=\neg Q_1}(I) = 1 - S_{Q_1}(I) \end{aligned}$$

Consider for example a query Q:

$$Q = (v_1 \vee v_2 \vee v_3) \wedge (v_4 \vee (v_5 \wedge v_1)) \tag{8}$$

The degree of membership of an image I in the fuzzy set corresponding to Q can be determined as follows:

$$S_Q(I) = min\begin{pmatrix} max\left(S_{v_1}(I), S_{v_2}(I), S_{v_3}(I)\right), \\ max\left(S_{v_4}(I), min(S_{v_5}(I), S_{v_1}(I)\right) \end{pmatrix} \quad (9)$$

The value $S_{v_i}(I)$ in (9) is determined using the appropriate similarity or distance measure for the feature v and appropriately normalized. Once the membership value of the image in the fuzzy set associated with the query is determined, these values are used to rank the images, where a higher value of $S_Q(I)$ represents a better match of the image I to the query Q.

4.6 Fuzzy Model Evaluation Algorithms

In this section, we present the algorithms used to compute the nodes in the query tree for the fuzzy Boolean retrieval model. For simplicity we restrict ourselves to compute only binary nodes. That is, we assume that the query node Q has exactly two children, A, and B. Algorithms are presented for the following three cases: $Q = A \wedge B$, $Q = A \wedge \neg B$, and $Q = A \vee B$. As described in Section 4, we only develop algorithms for positive conjunctive, negated conjunctive queries with a positive term and disjunctive queries.

In describing the algorithms the following notation is used.

- An image I is represented by a pair of components $\langle I, similarity_Q(I)\rangle$, denoted by the key (*I.image*) and the degree of membership (*I.degree*). The key identifies the image id and the degree of membership describes the similarity of match between the query feature and the database entries.
- A and B are assumed to be streams as defined in Section 4.4.
- Associated with each query node Q are three sets S_a, S_b, and S_{res}. Initially each of these sets are empty. The query node Q extracts images from the child streams (that is, A and B) and may buffer them into S_a and S_b (these represent the πA and πB rectangles from Fig. 4, respectively). The set S_{res} acts as a buffer of the images for the query node Q. Once a query node Q is able to establish the degree of membership of image I for Q (that is, $degree_Q(I)$), it places I in S_{res}(the result set). Thus, *I.degree* refers to the degree of membership of I according to Q, where $I \in S_{res}$.

The following three subsections describe the algorithms. For clarity purposes, when describing the algorithms we omit some critical error and boundary checking which needs to be considered in any implementation.

4.6.1 Conjunctive Query with Positive Subqueries

The algorithm shown in Fig. 6 computes the list of images ranked on their degree of membership to the query $Q = A \wedge B$, given input streams A and B which are ranked based on the degree of membership of images in A and B.

The operation performed in a binary operator node can be viewed as a function $S(x \in [0, 1], y \in [0, 1]) \rightarrow [0, 1]$. As an aid to explain the algorithm, we use contour plots that show the value of $S(x, y)$. These plots depict lines along which the value of S is the same over different parameters, so called iso similarity curves. In reality there are infinitely many such curves; the figures only show a few. The highest values of S (degree of membership) are in the white areas, the darker the region, the lower the value. Fig. 5a) shows the plot that corresponds to the fuzzy *and* operator.

Imagine an overlay of Fig. 4 on top of Fig. 5a. As OABB grows, whole iso similarity curves are completely contained in OABB. Given the geometry of the curves, we notice that for any OABB defined as the rectangle bounded by (a, b)-(1, 1), there is a curve of minimum similarity along the square (c, c)-(1, 1) where $c = \max(a, b)$. Images contained in this square are completely determined and are safe to be returned as answers. As an example, I_1 is contained in the first such square to appear. This is indeed the best image. Discriminating between I_2 and I_3 is more difficult. They both yield similar degrees of membership. Once the OABB has grown to contain both images, a decision as to the ranking is done. I_4 does not participate in this process since I_2 and I_3 are definitely better than I_4. The algorithm relies on this fact, but grows the OABB by exactly one image at a time, thus the next lower iso similarity curve

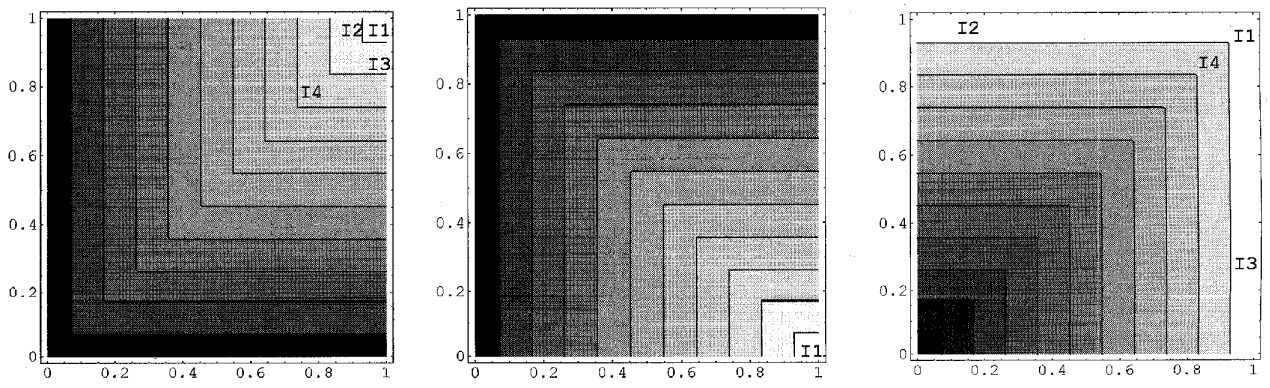
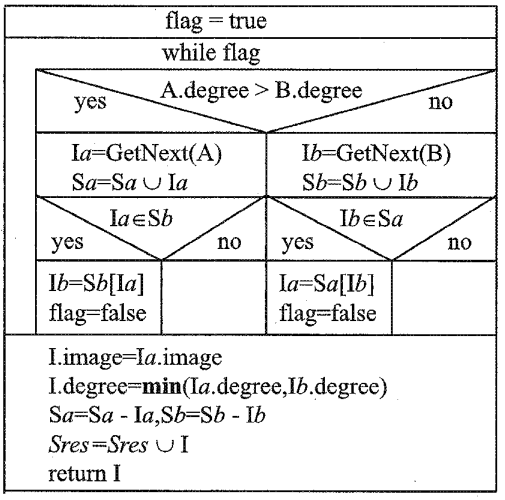


Fig. 5. Contour graphs for fuzzy operators. Whiter is higher, darker is lower similarity value: (a) fuzzy *and* operator, (b) fuzzy *and not* operator, (c) fuzzy *or* operator.

```
Algorithm GetNextAnd_Fuzzy(A, B)
;returns: next best image in A and B
  while (TRUE)
    I_a = Peek (A), I_b = Peek (B)
    if I_a.degree > I_b.degree then
      I_a = GetNext(A)
      S_a = S_a ∪ I_a
      if I_a.image ∈ S_b then ;image already seen in B
        I_b = image S_b[I_a.image]
        exit loop
      exit if
    else
    if I_b.degree > I_a.degree then
      I_b = GetNext(B)
      S_b = S_b ∪ I_b
      if I_b.image ∈ S_a then ;image already seen in A
        I_a = image S_a[I_b.image]
        exit loop
      end if
    end if
end while
; reached upon finding a common image in S_a and S_b
I.image = I_a.image
I.degree = min (I_a.degree, I_b.degree)
S_a = S_a − I_a, S_b = S_b − I_b, S_res = S_res ∪ I
return I
```

(a)



(b)

Fig. 6. Algorithm returning the next best for the fuzzy *and* case: (a) pseudocode, and (b) flow graph.

is exposed and the latest image to join OABB is the next answer. At each stage, the best image out of the sources A and B is chosen and added to sets $S_a(\pi A)$ and $S_b(\pi B)$ which function as buffers of images already observed from the corresponding stream. When an image is found that was already observed in the other stream, the loop is terminated and this is the next best image according to the query node Q (it just joined the rectangle OABB, thus encompassing the next iso similarity curve that has an image). Notice that $|S_a \cup S_b|$ will never exceed the size of the feature collection. The resulting image is returned with the degree equal to the minimum degree of the image in both streams and lastly recorded in the result set.

4.6.2 Conjunctive Query with Negative Subquery

We next present the algorithm for computing the query $Q = A \wedge \neg B$; it is presented in Fig. 7. Fig. 5b shows the contour plot that corresponds to this query. A strategy similar to the previous subsection could be used if traversing the stream B in reverse order was possible. This implies a furthest neighbor query that is not supported. The positive term is used to guide the search and the negative subquery used to determine the final degree of membership. The OABB thus only considers entries from stream A and never grows in the B stream (which is never constructed). *Probe* is then used to complete the degree of membership of an image. As an example, image I_1 is best if it is located early in stream A and its similarity to the query feature that corresponds to B is very low.

This algorithm contains an auxiliary set S_{aux} to hold images retrieved from stream A and whose final degree of membership is established, but resulted lower than the membership degree in A. These images need to be delayed until such time that it is safe to return them. For each iteration of the loop, there are three possibilities:

- $S_{aux} \neq \emptyset \wedge$ Peek(A).*degree* $\leq$ MaximumDegree(S_{aux}) the best image in the auxiliary set has higher membership degree than than the top image from A. In this case, the result is clear (return top image form S_{aux}), since *min* is used, no better image will come from A.
- ($S_{aux} = \emptyset \vee$ Peek(A).*degree* > MaximumDegree(S_{aux})) $\wedge$ Peek(A).*degree* $\leq$ Probe(Peek(A).*image*, $\neg B$).*degree* there is no better candidate on hold and the degree of the best image from A is lower (and thus determines the answer) than the probe on the negative subquery. The answer is the best image from A.
- ($S_{aux} = \emptyset \vee$ Peek(A).*degree* > MaximumDegree(S_{aux})) $\wedge$ Peek(A).*degree* > Probe(Peek(A).*image*, $\neg B$).*degree* there is no better candidate on hold and the degree of the best image from A is higher than the probe on the negative subquery. The final membership degree is determined by the probe and the image is sent to the auxiliary set to wait until it is safe to return it.

The loop iterates until a result is found.

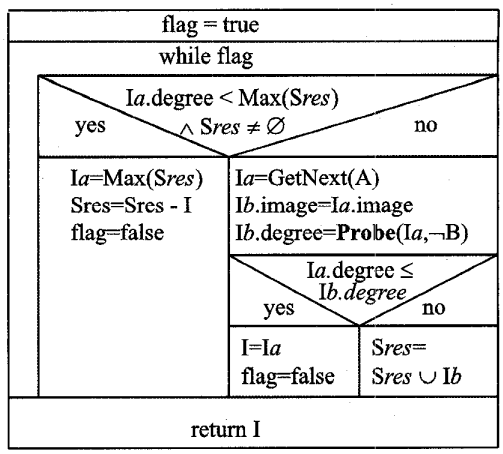
4.6.3 Disjunctive Query

The algorithm shown in Fig. 8 computes the set of images ranked on their degree of membership to the query $Q = A \vee B$, given input streams A and B which are ranked based on the degree of membership of images in A and B.

Fig. 5c shows the contour plot for the disjunctive fuzzy operator. By overlaying Fig. 4 on Fig. 5c it can be seen that any OABB intersects iso similarity curves (unless it is the whole space). This means no curve will be contained in any

```
Algorithm GetNextAnd_Not_Fuzzy(A, B)
;returns: next best image in A and not B
  while (TRUE)
    I_a = Peek (A)
    if S_aux ≠ Ø ∧ I_a.degree < MaximumDegree(S_aux) then
      I = image from S_aux with maximum degree
      S_aux = S_aux − I
      exit loop
    else
      I_a = GetNext(A) ; consume from A
      I_b.image = I_a.image
      I_b.degree = Probe(I_a, ¬B)
      if I_a.degree ≤ I_b.degree then
        I = I_a
        exit loop
      else
        S_aux = S_aux ∪ I_b
      end if
    end if
  end while
  S_res = S_res ∪ I
  return I
```

(a)

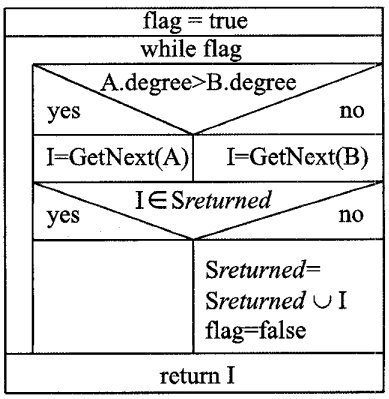


(b)

Fig. 7. Algorithm returning the next best for the fuzzy *and not* case: (a) pseudocode, (b) flow graph.

```
Algorithm GetNextOr_Fuzzy(A, B)
;returns: next best image in A or B
  flag = TRUE
  while (flag)
    I_a = Peek (A), I_b = Peek (B)
    if I_a.degree > I_b.degree then
      I = GetNext(A)
    else
      I = GetNext(B)
    end if
    flag = FALSE
    if I.image ∈ S_res then
      flag = TRUE
    end if
  end while
  S_res = S_res ∪ I
  return I
```

(a)



(b)

Fig. 8. Algorithm returning the next best for the fuzzy *or* case: (a) pseudocode, (b) flow graph.

OABB, so unless the whole collection is retrieved, no definite ranking exists. This results in two options,

1) return only those images in the OABB, and
2) follow a different strategy.

In the first case, to return I_2, the OABB would cover most of the collection, including I_4, but I_4 which is in OABB much earlier than any of I_2 or I_3 is worse than I_1, I_2, and I_3.

Fortunately, we can follow a different strategy instead. By exploiting the properties of the *max* operator, I_1, I_2, and I_3 have the same membership degree, they only rely on one (the maximum) of their membership degrees in subqueries and thus can safely ignore the other. Since images with better membership degrees are examined first, this is sufficient to determine the final membership degree.

The algorithm essentially consists of a merge based on the degree of membership value but makes sure that an image that was already returned is ignored as a result (duplicate removal). This accomplishes the desired *max* behavior of the degree function associated with the disjunction in the fuzzy model.

4.7 Probabilistic Boolean Model

Let $Q(v_1, v_2, \ldots, v_n)$ be a query and I an image. In the probabilistic Boolean model, the similarity

$$S_{F_i}(F_i(I), v_j)$$

between the query variable v_j and the corresponding feature in the image is taken to be the probability of the image I matching the query variable v_j, denoted by $P(v_j \mid I)$. These

probability measures are then used to compute the probability that I satisfies the query $Q(v_1, v_2, \ldots, v_n)$ (denoted by $P(Q(v_1, v_2, \ldots, v_n) \mid I)$) which is in turn used to rank the images. To enable computation of $P(Q(v_1, v_2, \ldots, v_n) \mid I)$, an assumption of *independence* is made. That is, we assume that for all variables v_i, v_j following holds:

$$P(v_i \wedge v_j \mid I) = P(v_i \mid I) \times P(v_j \mid I) \tag{10}$$

Developing a term and feature dependence model and incorporating it may improve retrieval performance further and is an important extension to our current work.

Once the probability of match is known for a basic feature, we next need to estimate the probability that the image satisfies the Boolean query $Q(v_1, v_2, \ldots, v_n)$, denoted by $P(Q \mid I)$. If Q is a disjunction ($Q = Q_1 \vee Q_2$), following the laws of probability, $P(Q_1 \vee Q_2 \mid I)$ can be estimated as follows:

$$P(Q_1 \vee Q_2 \mid I) = P(Q_1 \mid I) + P(Q_2 \mid I) - P(Q_1 \wedge Q_2 \mid I) \tag{11}$$

Since all probabilities are conditioned on the image I, we will omit this for brevity from now on. Similarly, $P(\neg Q)$ can be computed as follows:

$$P(\neg Q_1) = 1 - P(Q_1) \tag{12}$$

To compute conjunction queries, i.e., $Q = Q_1 \wedge Q_2$ we use

$$P(Q_1 \wedge Q_2) = P(Q_1) \times P(Q_2) \tag{13}$$

Our retrieval results (see Section 5) show that even if query terms are considered as independent, the resulting retrieval performance is quite good. It should be noted that although

$$S_{F_i}(F_i(I_j),\ v_k)$$

has the same value for the fuzzy and probabilistic models, their interpretation is different and yields different results (see Section 5).

4.8 Probabilistic Model Evaluation Algorithms

In this section, we present the algorithms used to compute the nodes in the query tree in the case of the probabilistic Boolean retrieval model. For simplicity we restrict ourselves to compute only binary nodes. That is, we assume that the query node Q has exactly two children, A and B. As for the fuzzy model, algorithms are only developed for the following three cases: $Q = A \wedge B$, $Q = A \wedge \neg B$, and $Q = A \vee B$.

Based on Section 4.7, the probability is computed at the internal nodes according to the equations below and is restricted to lie in [0, 1].

$$\begin{aligned}
\textbf{And} \quad & S_{Q=Q_1 \wedge Q_2}(I) = S_{Q_1}(I) \times S_{Q_2}(I) \\
\textbf{Or} \quad & S_{Q=Q_1 \vee Q_2}(I) = S_{Q_1}(I) + S_{Q_2}(I) - S_{Q_1}(I) \times S_{Q_2}(I) \\
\textbf{Not} \quad & S_{Q=\neg Q_1}(I) = 1 - S_{Q_1}(I)
\end{aligned}$$

In describing the algorithms the following notation is used:

- An image I is represented by a pair of components $\langle I, similarity_Q(I)\rangle$, composed by the key (*I.image*) which identifies the image id, and the similarity which identifies the probability that the image satisfies the query (*I.prob*).
- A and B are assumed to be streams as defined in Section 4.4.
- Associated with each query node Q are three sets S_a, S_b, and S_{res}. Initially each of these sets are empty. The query node Q extracts images from the child streams (that is, A and B) and may buffer them into S_a and S_b (these represent the πA and πB rectangles from Fig. 4, respectively). The set S_{res} acts as a buffer of the images for the query node Q. Once a query node Q is able to establish the probability of match of image I for Q (that is, $probability_Q(I)$), it places I in S_{res} (the result set). Thus, *I.prob* refers to the probability that image I matches the query Q.

The following three subsections describe the algorithms used to implement the above shown operations in an efficient manner. For clarity purposes, when describing the algorithms below we omit some critical error and boundary checking which needs to be considered in any implementation.

4.8.1 Conjunctive Query with Positive Subqueries

The algorithm in Fig. 10 computes the set of images ranked on their probability of match to the query $Q = A \wedge B$, given input streams A and B which are ranked based on their matching probability of images in A and B.

It is interesting to note that an algorithm similar to the one proposed in Section 4.6.1 will not work properly. To understand this, observe Fig. 9a and recall the OABB suggested in Section 4.4. The rectangle will contain a region with images that have been observed in both streams, yet the distribution of probability is complex within this rectangle. This requires a modified algorithm that returns images only when it is safe to do so. Similarly to the fuzzy case, there is a minimum value iso similarity curve completely covered by any OABB. The probability value for this curve is defined by its intersection with the axes. So, for an OABB bounded by (a, b)-(1, 1), all images with known probability of more than the maximum of a and b are safe to be returned. Note however that the OABB will also contain images with known final probability less than this amount, these are retained in an auxiliary set. Images in this auxiliary set become safe to return when the OABB covers a sufficiently low iso probability curve such that its probability is lower or equal to that of the now safe image. As an example, consider Fig. 9a. There are four images in the whole collection. I_2 is the first to be included in an OABB. When this happens, I_1 is partially known in πA. Even though OABB contains only one image with known final probability, it cannot yet be returned since it does not lie on an iso probability curve completely covered by OABB. Then I_1 will be included in OABB, but it also cannot yet be returned. The curve just below I_1 intersects with a vertical line drawn from I_3. Until this is cleared, I_1 and thus I_2 cannot be returned. When I_4 is added, the highest iso probability curve that is lower than I_1, I_2, and I_3 is clear of the projection of I_4 onto the axes, thus it is safe to return all of I_1, I_2, and I_3 at this stage.

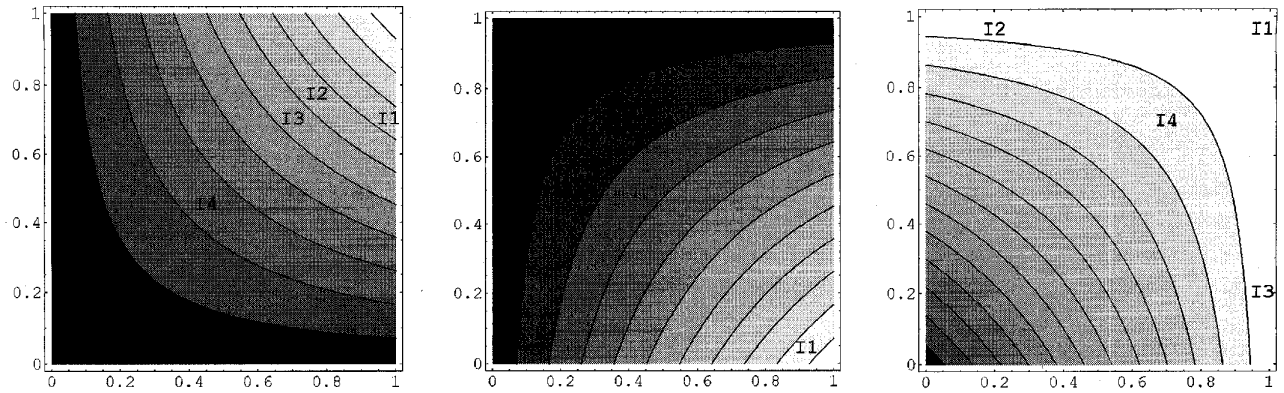
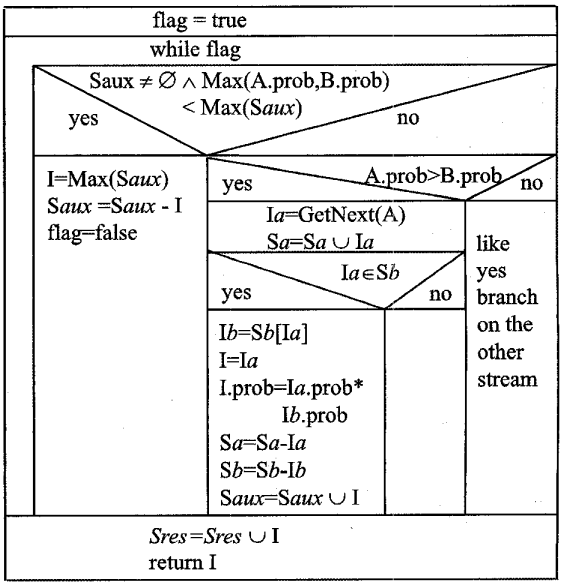


Fig. 9. Contour graphs for probabilistic operators: (a) probabilistic *and* operator, (b) probabilistic *and not* operator, and (c) probabilistic *or* operator. Whiter is higher, darker is lower similarity value.

```
Algorithm GetNextAnd_Probability(A, B)
;returns: next best image in A and B
  flag = TRUE
  while (flag)
    I_a = Peek (A), I_b = Peek (B)
    if S_aux ≠ ∅ ∧ Max(I_a.prob, I_b.prob) <
        MaximumProbability(S_aux) then
      I = image from S_aux with maximum probability
      S_aux = S_aux − I
      flag = FALSE
    else
      if I_a.prob > I_b.prob then
        I_a = GetNext(A)
        S_a = S_a ∪ I_a
        if I_a ∈ S_b then
          I_b = image from S_b equivalent to I_a
          I = I_a
          I.prob = I_a.prob × I_b.prob
          S_a = S_a-I_a, S_b = S_b-I_b, S_aux = S_aux ∪ I
        end if
      else
        ; symmetric code to then branch
      end if
    end if
  end while
  S_res = S_res ∪ I
return I
```

(a)



(b)

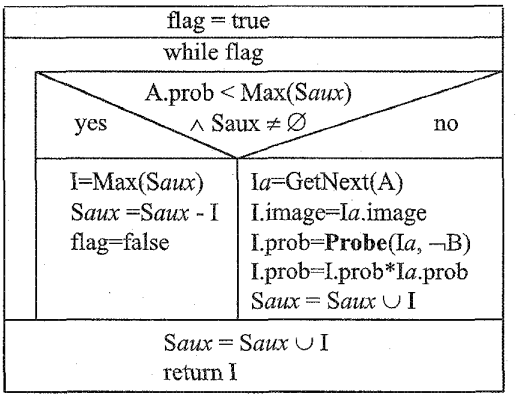
Fig. 10. Algorithm that implements the *and* operator for the probabilistic case: (a) pseudocode, (b) flow graph.

The algorithm first tests if there is a safe image in the auxiliary set to return and does so if there is one. Otherwise, it extracts the next best image from the better of A or B and tries to include it in OABB by finding it to be in the intersection. If unsuccessful, it is stored in one of the sets corresponding to πA or πB. The loop iteratively checks for safety and fetches images until a safe image can be returned. Note that unlike in the fuzzy case, the only way to exit the loop is by an image being safe as defined above. Of course in the fuzzy algorithms, returned images were also safe, but the safety criteria is so simple, that multiple loop exists exist.

An optimization on this algorithm is to slightly modify the safety criteria. The criteria described above is simple to understand: an image is not safe until all the region of higher probability has been seen. The danger of not following this strategy is that for some images, only one probability has been retrieved, and the other is unknown. The above safety criteria is pessimistic in that it assumes that the other probability could be any value, while it is in fact bounded by the top probability in the stream where the image has not yet been retrieved. If I_k.prob requires $I_k.\text{prob}_A$ and $I_k.\text{prob}_B$ to compute $I_k.\text{prob} = I_k.\text{prob}_A \times I_k.\text{prob}_B$, then an upper bound on the probability of image I_k is:

```
Algorithm GetNextAnd_Not_Probability(A, B)
;returns: next best image in A and not B
  flag = TRUE
  while (flag)
    I_a = Peek(A); best from A
    if S_aux ≠ ∅ ∧ I_a.prob < MaximumProbability(S_aux) then
      I = image from S_aux with maximum probability
      S_aux = S_aux − I
      flag = FALSE
    else
      I = GetNext(A)
      I_b.prob = Probe(I, ¬B)
      I_a.prob = I_a.prob × I_b.prob
      S_aux = S_aux ∪ I
    end if
  end while
  S_res = S_res ∪ I
  return I
```

(a)



(b)

Fig. 11. Algorithm that implements the *and not* operator for the probabilistic case: (a) pseudocode, (b) flow graph.

$$\text{Peek}(A).\text{prob} \times I_k.\text{prob}_B \quad \text{if } I_k.\ \text{prob}_B \text{ is known, or} \qquad (14)$$
$$\text{Peek}(B).\text{prob} \times I_k.\ \text{prob}_A \quad \text{if } I_k.\ \text{prob}_A \text{ is known}$$

This more sophisticated criteria is not incorporated in Fig. 10, instead the simpler criteria described above is included.

4.8.2 Conjunctive Query with Negative Subquery

We next develop the algorithm for computing the query $Q = A \wedge \neg B$; it is shown in Fig. 11. The algorithm is different compared to the one developed for the conjunctive query with no negative subquery. As described for the fuzzy model, a similar method to the conjunctive query with only positive subqueries could be used if traversing the B stream in inverse was feasible. This is however not the case. This algorithm follows the safety criteria specified in the previous subsection, however only the stream for A is used in computing the probability of images according to $A \wedge \neg B$. Images are retrieved from the input stream A in rank order. For a given image I its probability with respect to the subquery $\neg B$ is evaluated by performing a probe on image I and evaluating its probability of match. Once the probability of match of an image I according to $\neg B$ has been established, we can determine its final probability according to the query Q, and the image is inserted into an auxiliary set that is used to verify the safety criteria. An image is only returned if it successfully passes the safety test, thus every returned image was in the auxiliary set. Effectively, every image retrieved from A results in a probe to B.

4.8.3 Disjunctive Query

Finally, to compute a disjunctive query node, we need the algorithm shown in Fig. 12. Disjunctive queries are hard to compute in this case. Consider Fig. 9c, images I_1, I_2, and I_3 have very similar probabilities. In the fuzzy case, the iso similarity curves were parallel to the axes and we could exploit the max behavior. This is not possible here. In addition, notice that no iso probability curve will be contained in any OABB (unless everything is read in). Two distinctions exist with the fuzzy version:

1) The final probability does depend on all the query terms, while in the fuzzy model, only the best one is relevant.
2) Iso probability curves are not even piecewise parallel to the axes.

Since image I may have a higher probability in one stream than another, we would need to store it until a possibly much worse (and, much later) match occurs from the other stream. Indeed, to return I_1, both I_2 and I_3 need to be included in the OABB. Potentially, this results in a very large initial overhead (latency) to find the first few results. To overcome this limitation, once an image is seen for the first time, its full probability is established with appropriate probes.

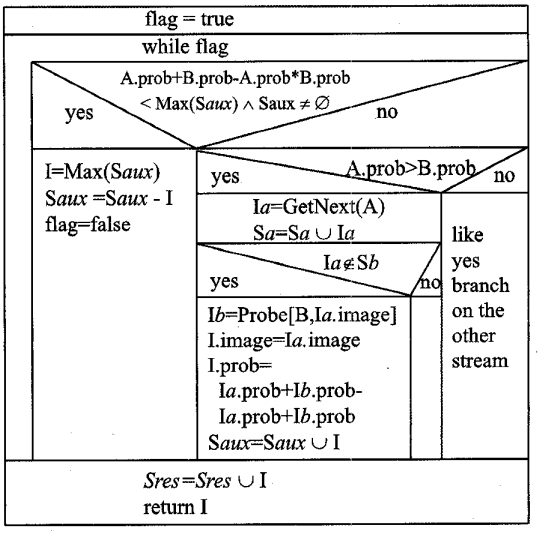
To follow the algorithm, the notion of safety is used again. When is it safe to return I_1 given that we only have partial knowledge for I_2 and I_3? Probes are used to establish missing probabilities and a final probability score is computed. Images are then stored into an auxiliary set until they can safely be returned.

Images can safely be returned when their known probability is larger than the best to come. All images in S_{aux} can be partitioned into those with probability above (*safe set*) and below (*unsafe set*) the value Peek(A).prob + Peek(B).prob − Peek(A).prob × Peek(B).prob. Those in the *safe* partition necessarily have higher probability than those in the *unsafe* partition, but also any combination of images that remain to be considered in streams A and B would fall into the current *unsafe* partition. Images from the safe set can now be returned in rank order. The algorithm grows S_{aux} one by one and at each stage verifies for safety. The safe set may contain at most one element, if present it is returned as an answer and removed from the safe set.

The algorithm assumes that probing is possible on subqueries. So far, only algorithms based on negation have required this and then only for the negation operator. If probing on subqueries is expensive, an alternate algorithm (not shown here) can be constructed as in the conjunctive query case. When one component probability of an image I_k

```
Algorithm GetNextOr_Probability (A, B)
;returns:  next best image in A or B
  flag = TRUE
  while (flag)
    I_a = Peek (A), I_b = Peek (B)
    if S_aux ≠ ∅ ∧ I_a.prob + I_b.prob - I_a.prob × I_b.prob ≤
        MaximumProbability(S_aux)then
      I = image from S_aux with maximum probability
      S_aux = S_aux - I
      flag = FALSE
    else
      if I_a.prob > I_b.prob then
        I_a = GetNext(A)
        S_a = S_a ∪ I_a
        if I_a ≠ ∅ S_b then ; do a probe
          I_b = probe(B, I_a.id)
          I.image - I_a.image
          I.prob - I_a.prob + I_b.prob - I_a.prob × I_b.prob
          S_aux = S_aux ∪ I
        end if
      else
      ; symmetric code to then branch
      end if
    end if
  end while
  S_res = S_res ∪ I
  return I
```

(a)



(b)

Fig. 12. Algorithm that implements the *or* operator for the probabilistic case: (a) pseudocode, (b) flow graph.

is known, an upper bound on the final probability can be established by:

$$\begin{aligned} upper(I_k) = \;& \text{Peek}(A).\text{prob} + I_k.\text{prob}_B \\ & - \text{Peek}(A).\text{prob} \times I_k.\ \text{prob}_B \\ & \qquad \text{if } I_k.\text{prob}_B \text{ is known, or} \\ upper(I_k) = \;& \text{Peek}(B).\text{prob} + I_k.\text{prob}_A \\ & - \text{Peek}(B).\text{prob} \times I_k.\text{prob}_A \\ & \qquad \text{if } I_k.\text{prob}_A \text{ is known} \end{aligned} \tag{15}$$

And the known probability component is a lower bound ($lower(I_k)$). Based on the known bounds for I_k, instead of waiting to complete its final probability, it is estimated as its lower bound ($lower(I_k)$). Once no upper bound ($upper(I_j)$) of any unsolved image can exceed $lower(I_k)$, and no combination of any images left in A and B can exceed $lower(I_k)$, then I_k is safe to return and is returned with probability $lower(I_k)$.

4.9 Comparison of Algorithms to Other Work

Recently, Fagin [12] proposed an algorithm to return the top k answers for queries with monotonic scoring functions that has been adopted by the Garlic multimedia information system under development at the IBM Almaden Research Center [10]. A function F is monotonic if $F(x_1, \ldots, x_m) \leq F(x'_1, \ldots, x'_m)$ for $x_i \leq x'_i$ for every i. Note that the scoring functions for both conjunctive and disjunctive queries for both the fuzzy and probabilistic Boolean models satisfy the monotonicity property. In [12], each stream i is accessed in sorted order based on the degree of membership to form a ranked set X_i, and a set $L = \cap_i X_i$ that contains the intersection of the objects retrieved from all streams. Once L contains k objects (to answer a top k query), all objects in $\cup_i X_i$ are used to perform probes on whichever streams they were not read from. This essentially completes all the information for the objects in the union and enables a final definite scoring and ranking of all objects in $\cup_i X_i$, then the top k are the final answer. This algorithm works in the general case, and is tailored in [12] to some specific scoring functions. This algorithm relies on reading a number of objects from each stream until it has k in the intersection. Then it falls back on probing to enable a definite decision. In contrast, our algorithms are tailored to specific functions that combine object scoring (here called fuzzy and probabilistic models). Our algorithms follow a *demand driven data flow* approach [19]. Instead of asking for the top k objects, only the next best element is requested and returned. This follows a fine grained pipelined approach. According to the cost model proposed in [12], the total database access cost due to probing can be much higher compared to the total cost due to sorted access. Only our algorithms involving negation require probing. We used probing in Section 4.8.3 for convenience, but sketched an alternate algorithm that does not require probing. Our demand driven approach reduces the wait time of intermediate answers in

a temporary file or buffer between operators in the query tree. This model is efficient in its time-space product memory costs [19]. On the other hand, in Garlic, the data items returned by each stream must wait in a temporary file until the completion of the probing and sorting process. Also, in the query processing model followed in MARS, the operators are implemented as iterators which can be efficiently combined with parallel query processing [18].

Another approach to optimizing query processing over multimedia repositories has been proposed in [7]. It presents a strategy to optimize queries when user's specify thresholds on the grade of match of acceptable objects as filter conditions. It uses the results in [12] to convert top-k queries to threshold queries and then process them as filter conditions. It shows that under certain conditions (uniquely graded repository), this approach is expected to access no more objects than the strategy in [12]. Like the former approach, this approach also requires temporary storage of intermediate answers and sorting before returning the answers to the user. Furthermore, while the above approaches have mainly concentrated on the fuzzy Boolean model, we consider both the fuzzy and probabilistic model in MARS. This is significant since the experimental results illustrate that the probabilistic model consistently outperforms the fuzzy model in terms of retrieval performance (discussed in Section 5).

5 Experimental Results

We have conducted extensive experiments of varied data sets to measure the performance of the retrieval models and query processing algorithms developed. This section presents the results of our experiments. First we briefly describe the parameters used to measure retrieval performance followed by a description of the data sets. Finally, we present the results along with our observations.

5.1 Evaluation Technique

Text retrieval systems typically use the following two metrics to measure the retrieval performance: *precision* and *recall* [40], [4]. Note that these metrics measure the retrieval performance as opposed to execution performance (retrieval speed).

Precision and recall are based on the notion that for each query, the collection can be partitioned into two subsets of documents. One subset is the set of relevant documents and is based on the user's criteria for relevance to the query. The second is the set of documents actually returned by the system as the result of the query. Now *precision* and *recall* can be defined as follows:

- **Precision** is the ratio of the number of relevant images retrieved to the total number of images retrieved. Perfect precision (100 percent) means that all retrieved images are relevant.

$$precision = \frac{|relevant \cap retrieved|}{|retrieved|} \tag{16}$$

- **Recall** is the ratio of the number of relevant images retrieved to the total number of relevant images. Perfect recall (100 percent) can be obtained by retrieving the entire collection, but the precision will be poor.

$$recall = \frac{|relevant \cap retrieved|}{|relevant|} \tag{17}$$

An IR system can be characterized in terms of performance by constructing a precision–recall graph for each query by incrementally increasing the size of the retrieved set i.e., by measuring the precision at different recall points. Usually, the larger the retrieved set, the higher the recall and the lower the precision. This is easily done in MARS since the query processing algorithms are implemented as a pipeline.

5.2 Data Sets Used

We have conducted experiments on two data sets. The first data set is a collection of images of ancient artifacts from the Fowler Museum of Cultural History. We used a total of 286 images of such artifacts. The relevance judgments for this collection were obtained from a class project in Library and Information Science department at the University of Illinois. Experts in librarianship consulted with the curator of the collection to determine appropriate queries and their answers. Queries posed to this collection range from simple single feature queries to complicated queries involving all the operators described above and both retrieval models, namely fuzzy and probabilistic. In all, five groups of related images were chosen. For each group several queries involving single features and arbitrary operations between them as well as different weightings were constructed. These relevant query groups ranged in their cardinality from 9 to 33 images.

The second data set is the Corel collection of images available online at http://corel.digitalriver.com. This collection contains around 70,000 images mostly of natural scenes. All images are catalogued into broad categories and each image carries an associated description. In this case, manually separating the collection into relevant and nonrelevant sets was infeasible due to the size. Instead we made use of our results in [32] to automatically determine the appropriate result set of each query.

5.3 Results

5.3.1 Fowler Collection

In this section, we describe the results of some experiments performed on the image collection from the Fowler Museum. Since the complete set of experiments are too large to include, we present only the results of certain representative experiments.

We conducted experiments to verify the role of feature weighting in retrieval. Fig. 13a shows results of a *shape or color* query i.e., to retrieve all images having either the same shape or the same color as the query image. We obtained four different precision recall curves by varying the feature weights. The retrieval performance improves when the shape feature receives more emphasis.

We also conducted experiments to observe the impact of the retrieval model used to evaluate the queries. We observed

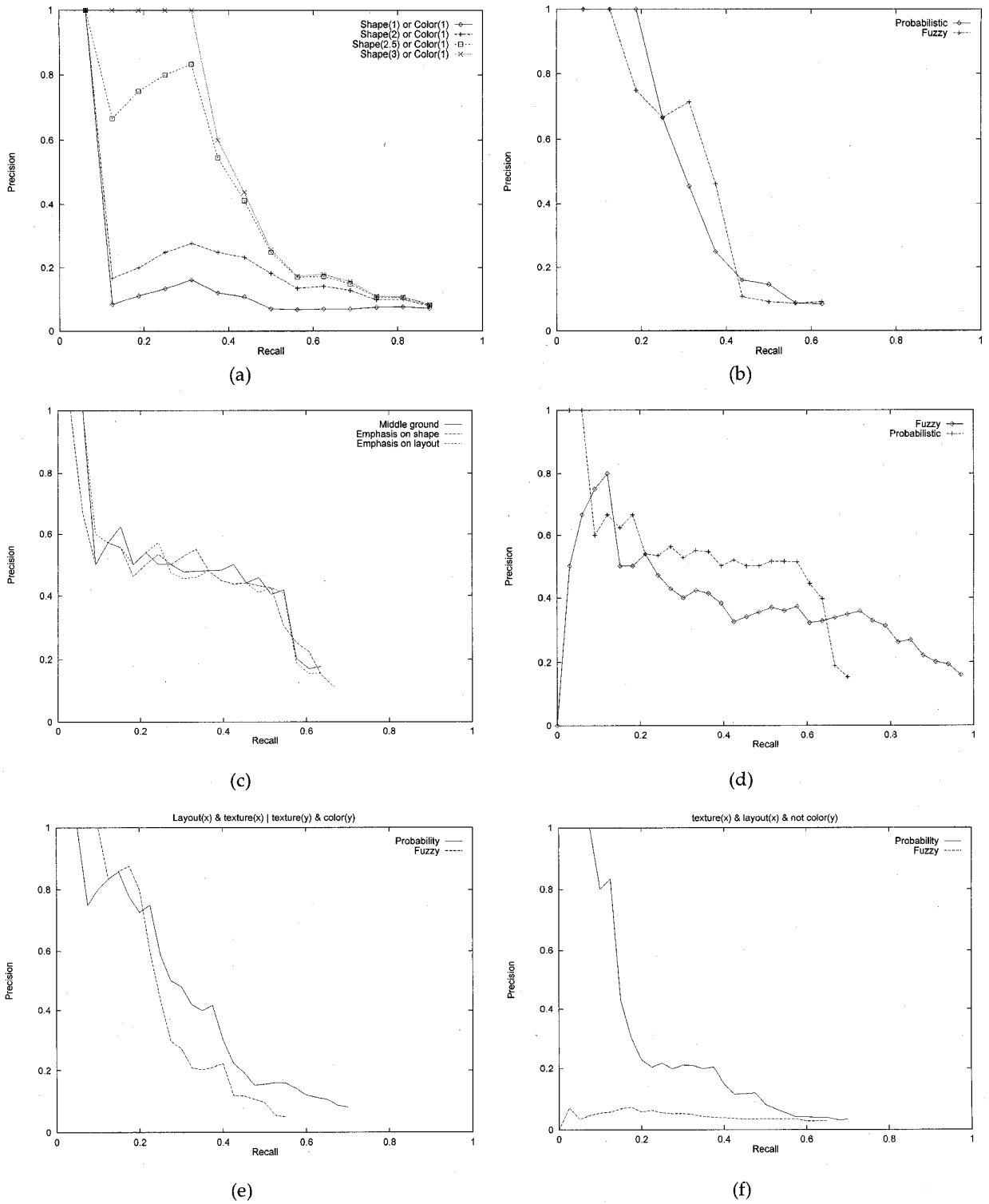


Fig. 13. Experimental result graphs: (a) effects of varying the weighting on a query, (b) fuzzy vs. probabilistic performance for query, (c) complex query with different weights, (d) fuzzy vs. probabilistic for same complex query, (e) corel query (comparing probability and fuzzy models), (f) corel query (comparing probability and fuzzy models).

that the fuzzy and probabilistic interpretation of the same query yields different results. Fig. 13b shows the performance of the same query (a *texture or color* query) in the two models. The result shows that neither model is consistently better that the other in terms of retrieval.

Fig. 13c shows a complex query (shape(I_i) *and* color(I_i) *or* shape(I_j) *and* layout(I_j) query) with different weightings. The three weightings fared quite similar, which suggests that complex weighings may not have a significant effect on retrieval performance. We used the same complex query to compare the performance of the retrieval models. The result is shown in Fig. 13d. In general, the probabilistic model outperforms the fuzzy model.

5.3.2 Corel Collection

For the Corel data set, Fig. 13e shows a query involving two database objects ranked under both fuzzy and probabilistic models. In this query, all terms are positive literals. Although at first the fuzzy model has good performance, the probabilistic model soon improves and stays better. Fig. 13f shows a query involving a negation. In this case the fuzzy model performed well below the probabilistic model. The next section discusses possible explanations.

5.4 Analysis of Data

Note the graphs shown are not always monotonic. The precision is expected to monotonically decrease as more and more images are retrieved. The small peaks in the graphs imply that a sequence of relevant images was quickly retrieved following a possibly long sequence of nonrelevant images. Taking averages over several queries would help in smoothing out these peaks. However, we do not take averages to depict the peculiar effects of individual queries. The way to read these graphs is that a higher curve is better than a lower one. This would mean that at all recall points, the precision was better.

We observe from Fig. 13a that the weighting of features can improve performance dramatically. The weights for the queries were determined subjectively and several combinations were tried. Automatic learning of these weights in MARS is an interesting extension of this work explored in [35], [37], [36]. We also observed (from Fig. 13c) that complex weighting strategies may not always improve performance significantly.

We observed that the probabilistic model is superior to the fuzzy model for Corel queries. A possible explanation for this (specially for Fig. 13f) is that the *min* and *max* operations used in the fuzzy model are too restrictive. They take into account only one of all their parameters while the probabilistic operators take into account all the parameters. These results scaled from 286 to 70,000 images. This gives us confidence in the robustness of our approach.

6 Related Work

Content-based retrieval of images is an active area of research being pursued independently by many research teams. Similar to MARS, most existing content-based image retrieval systems also extract low-level image features like color, texture, shape, and structure [14], [48], [29], [16], [31], [28], [33], [42]. However, compared to MARS the retrieval techniques supported in some of these systems are quite primitive. Many of these systems support queries only on single features separately. Certain other systems allow queries over multiple feature sets by associating a degree of tolerance with each feature. An image is deemed similar to the query if it is within the specified tolerance on all the query features. As discussed in Section 1.2, this approach has many drawbacks.

Some commercial systems have been developed. QBIC [16], standing for Query By Image Content, is the first commercial content-based Image Retrieval system. Its system framework and techniques had profound effects on later Image Retrieval systems. QBIC supports queries based on example images, user-constructed sketches and drawings and selected color and texture patterns, etc. The color features used in QBIC are the average (R, G, B), (Y, i, q), (L, a, b), and MTM (Mathematical Transform to Munsell) coordinates, and a k element Color Histogram. Its texture feature is an improved version of the Tamura texture representation [48], i.e., combinations of coarseness, contrast, and directionality. Its shape feature consists of shape area, circularity, eccentricity, major axis orientation, and a set of algebraic moments invariants. QBIC is one of the few systems which take into account high dimensional feature indexing. In its indexing subsystem, the KL transform is first used to perform dimension reduction and then R^*-tree is used as the multidimensional indexing structure.

Virage is a content-based image search engine developed at Virage Inc. Similar to QBIC, Virage [1] supports visual queries based on color, composition (color layout), texture, and structure (object boundary information). But Virage goes one step further than QBIC. It also supports arbitrary combinations of the above four atomic queries. Users can adjust the weights associated with the atomic features according to their own emphasis. In [1], Jeffrey et al. further proposed an open framework for image management. They classified the visual features ("primitive") as general (such as color, shape, or texture) and domain specific (face recognition, cancer cell detection, etc.). Various useful "primitives" can be added to the open structure depending on the domain requirements. To go beyond the query-by-example mode, Gupta and Jain proposed a nine-component *query language* framework in [21].

Photobook [33] is a set of interactive tools for browsing and searching images developed at the MIT Media Lab. Photobook consists of three subbooks, from which shape, texture, and face features are extracted respectively. Users can then query based on corresponding features in each of the three subbooks. In its more recent version of Photobook, FourEyes, Picard et al. proposed to include human in the image annotation and retrieval loop [30]. The motivation of this was based on the observation that there was no single feature which can best model images from each and every domain. Furthermore, human perception is subjective. They proposed a "society of models" approach to incorporate the human factor. Experimental results show that this approach is very effective in interactive image annotation.

In [24], Jain and Vailaya propose an image retrieval system based on color and shape. Their color measure is based on the RGB color space and Euclidean and histogram

intersection measures are used. For shape, they use a polygonal description that is resilient to scaling, translation and rotation. The proposed integration uses a weighted sum of shape and color to arrive at the final result. They address high dimensional feature indexing with a clustering approach, where clusters are build upon database creation time.

To date, no systematic approach to answering content-based queries based on image features has emerged. To address this challenge, similar to the approaches taken in information retrieval system, the approach we have taken in developing MARS is to support an "intelligent retrieval" model using which a user can specify their information need to the image database and the database provides a ranked retrieval of images to user's request. The retrieval model supported is a variation of the Boolean model based on probabilistic and fuzzy interpretations of distances between the image and the query.

7 Conclusions

To address the emerging needs of applications that require access to and retrieval of multimedia objects, we are developing the *Multimedia Analysis and Retrieval System* (MARS). In this paper, we described the retrieval subsystem of MARS and its support for content-based queries over image databases. To support content-based retrieval, in MARS many visual features are extracted from images—color, texture, shape, color, and texture layout. Information retrieval (IR) techniques, modified to work over visual features, are then used to map user's queries to a collection of relevant images. Specifically, extended Boolean models based on a probabilistic and fuzzy interpretation of Boolean operators are used to support ranked retrieval. Our results show that using IR techniques for content-based retrieval in image databases is a promising approach.

The work reported in this paper is being extended in many important directions. In our current system, we have concentrated on adapting the Boolean retrieval model for content-based retrieval of images. Many other retrieval models that have a better retrieval performance compared to the Boolean approach have been developed in the IR literature for textual databases [40], [4], [51]. We are currently exploring how these models can be adapted for content-based image retrieval. Furthermore, our current work has concentrated on image databases. We are also generalizing our approach to content-based retrieval in multimedia databases. Weighting is an important tool for the user to communicate to the system the relative importance of query components. We plan to explore the approach described in [13] and compare the impact of the weighting strategies on the quality of the retrieval. Finally, we are also exploring the use of relevance feedback techniques in our extended Boolean model.

Acknowledgments

The authors thank the anonymous reviewers for their suggestions for improvements. We also thank Alex Warshavsky for his help in implementing diverse similarity functions. Some example images used in this article are used with permission from the Fowler Museum of Cultural History at the University of California at Los Angeles. These images were part of an image database delivery project called the Museum Educational Site Licensing (MESL) Project, sponsored by the Getty Information Institute. A complementary set of images was obtained from the Corel collection available at http://corel.digitalriver.com.

This work was supported by the National Science Foundation CAREER Award No. IIS-9734300; in part by NSF CISE Research Infrastructure Grant No. CDA-9624396; and in part by the Army Research Laboratory under Cooperative Agreement No. DAAL01-96-2-0003. Michael Ortega is also supported, in part, by CONACYT Grant No. 89061. Yong Rui is also supported, in part, by the Department of Computer Science and Engineering in the College of Engineering at the University of Illinois at Urbana-Champaign.

References

[1] J.R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and C.-f. Shu, "The Virage Image Search Engine: An Open Framework for Image Management," *Proc. SPIE Storage and Retrieval for Still Image and Video Databases IV*, 1996.

[2] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles," *Proc. ACM SIGMOD*, May 1990.

[3] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When Is 'Nearest Neighbor' Meaningful," submitted for publication, 1998.

[4] J.P. Callan, W.B. Croft, and S.M. Harding, "The INQUERY Retrieval System," *Proc. Third Int'l Conf. Database and Expert Systems Applications*, Valencia, Spain, 1992.

[5] K. Chakrabarti and S. Mehrotra, "High Dimensional Feature Indexing Using Hybrid Trees," *Proc. ICDE 15, Int'l Conf. Data Eng.*, Mar. 1999, to appear.

[6] T. Chang and C.-C.J. Kuo, "Texture Analysis and Classification with Tree-Structured Wavelet Transform," vol. 2, no. 4, pp. 429-441, Oct. 1993.

[7] S. Chaudhari and L. Gravano, "Optimizing Queries over Multimedia Repositories," *Proc. SIGMOD*, 1996.

[8] T. Chiueh, "Content-Based Image Indexing," *Proc. VLDB*, 1994.

[9] P. Ciaccia, M. Patella, and P. Zezula, "M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces," *Proc. VLDB*, 1997.

[10] W.F. Cody et al., "Querying Multimedia Data from Multimedia Repositories by Content: The Garlic Project," *Proc. VDB-3, Visual Database Systems*, 1995.

[11] G. Evangelidis, D. Lomet, and B. Salzberg, "The hb^{π}-Tree: A Modified hB-Tree Supporting Concurrency, Recovery and Node Consolidation," *Proc. VLDB*, 1995.

[12] R. Fagin, "Combining Fuzzy Information from Multiple Systems," *Proc. 15th ACM Symp. PODS*, 1996.

[13] R. Fagin and E.L. Wimmers, "Incorporating User Preferences in Multimedia Queries," *Int'l Conf. Database Theory*, 1997.

[14] C. Faloutsos, M. Flocker, W. Niblack, D. Petkovic, W. Equitz, and R. Barber, "Efficient and Effective Querying By Image Content," IBM Research Report No. RJ 9453 (83074), Aug. 1993.

[15] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Databases," *Proc. SIGMOD*, 1994.

[16] M. Flickner et al., "Query by Image and Video Content: The QBIC System," *Computer*, Sept. 1995.

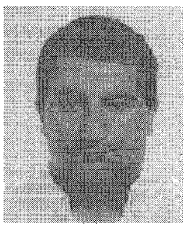
[17] J. Foley, A. van Dam, S. Feiner, and J. Hughes, *Computer Graphics*, Addison-Wesley, second ed., 1990.

[18] G. Graefe, "Encapsulation of Parallelism in the Volcano Query Processing System," *Proc. SIGMOD*, ACM, 1990.

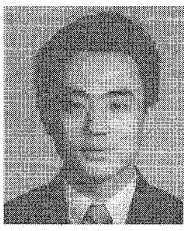
[19] G. Graefe, "Query Evaluation Techniques for Large Databases," *ACM Computing Surveys*, vol. 25, no. 2, 1993.

[20] M.H. Gross, R. Koch, L. Lippert, and A. Dreger, "Multiscale Image Texture Analysis in Wavelet Spaces," *Proc. IEEE Int'l Conf. Image Processing*, 1994.

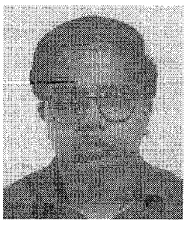
[21] A. Gupta and R. Jain, "Visual Information Retrieval," *Comm. ACM*, vol. 40, no. 5, May, 1997.
[22] A. Guttman, "R-Tree: A Dynamic Index Structure for Spatial Searching," *Proc. SIGMOD*, ACM, pp. 47-57, 1984.
[23] R.M. Haralick, K. Shanmugam, and I. Dinstein, "Texture Features for Image Classification," *IEEE Trans. Systems, Man, and Cybernetics*, , vol. 3, no. 6, 1973.
[24] A.K. Jain and A. Vailaya, "Image Retrieval Using Color and Shape," *Pattern Recognition*, vol. 29, no. 8, 1996.
[25] A. Kundu and J.-L. Chen, "Texture Classification Using QMF Bank-Based Subband Decomposition, *CVGIP: Graphical Models and Image Processing*, vol. 54, no. 5, pp. 369-384, Sept. 1992.
[26] A. Laine and J. Fan, "Texture Classification by Wavelet Packet Signatures," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1,186-1,191, 1993.
[27] K. Lin, H.V. Jagadish, and C. Faloutsos, "The TV-Tree—An Index Stucture for High Dimensional Data," *VLDB J.*, 1994.
[28] B.S. Manjunath and W.Y. Ma, "Texture Features for Browsing and Retrieval of Image Data," Technical Report No. TR-95-06, CIPR, July 1995.
[29] S. Mehrotra, K. Chakrabarti, M. Ortega, Y. Rui, and T.S. Huang, "Towards Extending Information Retrieval Techniques for Multimedia Retrieval," *Proc. Third Int'l Workshop Multimedia Information Systems*, Como, Italy, 1997.
[30] T.P. Minka and R.W. Picard, "Interactive Learning Using a 'Society of Models'," Technical Report No. 349, Media Lab, MIT, 1996.
[31] M. Miyahara et al., "Mathematical Transform of (R, G, B) Color Data to Munsell (H, V, C) Color Data," *Proc. Visual Comm. and Image Processing*, vol. 1,001, SPIE, 1988.
[32] M. Ortega, K. Chakrabarti, K. Porkaew, and S. Mehrotra, "Cross Media Validation in a Multimedia Retrieval System," *Proc. Digital Libraries Workshop on Metrics in Digital Libraries*, ACM, 1998.
[33] A. Pentland, R.W. Picard, and S. Sclaroff, "Photobook: Tools for Content-Based Manipulation of Image Databases," *Proc. Storage and Retrieval for Image and Video Databases*," vol. 2, pp. 34-47, SPIE, Bellingham, Wash., 1994.
[34] J.T. Robinson, "The K-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes," *Proc. SIGMOD* ACM, 1981.
[35] Y. Rui, T.S. Huang, and S. Mehrotra, "Content-Based Image Retrieval with Relevance Feedback in MARS," *Proc. IEEE ICIP Int'l Conf. Image Processing*, 1997.
[36] Y. Rui, T.S. Huang, S. Mehrotra, and M. Ortega, "Automatic Matching Tool Selection via Relevance Feedback in MARS," *Proc. Second Int'l Conf. Visual Information Systems*, 1997.
[37] Y. Rui, T.S. Huang, S. Mehrotra, and M. Ortega, "A Relevance Feedback Architecture in Content-Based Multimedia Information Retrieval Systems," *Proc. IEEE Workshop Content-Based Access of Image and Video Libraries*, IEEE, 1997.
[38] Y. Rui, A.C. She, and T.S. Huang, "Modified Fourier Descriptors for Shape Representation—A Practical Approach," *Proc. First Int'l Workshop Image Databases and Multimedia Search*, Amsterdam, 1996.
[39] G. Salton, E. Fox, and E. Voorhees, "Extended Boolean Information Retrieval," *Comm. ACM*, vol. 26, no. 11, pp. 1, 022-1, 036, Nov. 1983.
[40] G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill Computer Science Series, 1983.
[41] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+-Tree: A Dynamic Index for Multi-Dimensional Objects," *Proc. 12th VLDB*, 1987.
[42] J.R. Smith and S.-F. Chang, "Tools and Techniques for Color Image Retrieval," *Proc. IS&T, Storage and Retrieval for Image and Video Databases*, vol. 2,670, SPIE, 1994.
[43] J.R. Smith and S.-F. Chang, "Transform Features for Texture Classification and Discrimination in Large Image Databases," *Proc. IEEE ICIP, Int'l Conf. Image Processing*, 1994.
[44] J.R. Smith and S.-F. Chang, "Single Color Extraction and Image Query," *Proc. IEEE ICIP, Int'l Conf. Image Processing*, 1995.
[45] J.R. Smith and S.-F. Chang, "Automated Binary Texture Feature Sets for Images Retrieval," *Proc. ICASSP*, Atlanta, 1996.
[46] M. Stricker and M. Orengo, "Similarity of Color Images," *Proc. Conf Visual Comm. and Image Processing*, pp. 381-392, SPIE, 1995.
[47] M. Swain and D. Ballard, "Color Indexing," *Int'l J. Computer Vision*, vol. 7, no. 1, 1991.
[48] H. Tamura et al., "Texture Features Corresponding to Visual Perception," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 8, no. 6, June 1978.
[49] H. Tamura, S. Mori, and T. Yamawaki, "Texture Features Corresponding to Visual Perception," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 8, no. 6, SMC, 1978.
[50] K.S. Thyagarajan, T. Nguyen, and C. Persons, "A Maximum Likelihood Approach to Texture Classification Using Wavelet Transform," *Proc. IEEE ICIP, Int'l Conf. Image Processing*, 1994.
[51] C.J. Van Rijsbergen, *Information Retrieval*. London: Butterworths, 1979.
[52] D. White and R. Jain, "Similarity Indexing with the SS-Tree," *Proc. ICDE*, 1995.



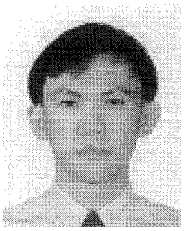
Michael Ortega received his BE degree with honors from the Mexican Autonomous Institute of Technology in August 1994, with a SEP fellowship for the duration of the studies. Currently he is pursuing his graduate studies at the University of Illinois at Urbana-Champaign. He received a Fulbright/CONACYT/García Robles scholarship to pursue graduate studies, as well as the Mavis Award at the University of Illinois. His research interests include multimedia databases, database optimization for uncertainty support, and content-based multimedia information retrieval. He is a member of the Phi Kappa Phi honor society, the IEEE Computer Society, and the ACM, and a student member of the IEEE.



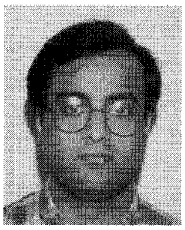
Yong Rui received the BS degree from Southeast University, Peoples Republic of China, in 1991 and the MS degree from Tsinghua University, Peoples Republic of China, in 1994, both in electrical engineering. He is currently finishing the PhD degree in electrical and computer engineering at the University of Illinois at Urbana-Champaign. He was a graduate research assistant in the Department of Automation at Tsinghua University from 1991 to 1994. He joined KangTuo Microcomputer Corporation, Beijing, in 1994 as a research engineer. Since 1995, he has been a graduate research assistant in the Image Formation and Processing Group of the Beckman Institute for Advance Science and Technology at UIUC. During the summer of 1998, he was a research intern at the Vision Technology Group of Microsoft Research, in Redmond, Washington. His research interests include multimedia information retrieval, multimedia signal processing, computer vision, and artificial intelligence. He has published more than 20 technical papers in the above areas. He was a Huitong University Fellowship recipient in 1989 and 1990, a Guanghua University Fellowship recipient in 1992 and 1993, and a CSE Engineering College Fellowship recipient in 1996 to 1998.



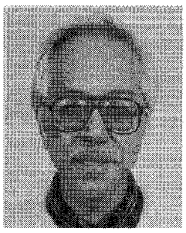
Kaushik Chakrabarti received the BTech degree in computer science and engineering from the Indian Institute of Technology, Kharagpur, in 1996. Currently, he is working towards his PhD degree in computer science at the University of Illinois at Urbana-Champaign. His research interests include multimedia databases, geographical information systems, and spatial and temporal databases.



Kriengkrai Porkaew received his BS degree in computer science from Ramkhamhaeng University, Bangkok, Thailand; his MS degree in computer science from the University of Illinois at Urbana-Champaign, in 1996; and he is currently pursuing his PhD degree at the University of Illinois at Urbana-Champaign. His research interests include multimedia databases and information retrieval.



Sharad Mehrotra received his MS and PhD degrees in computer science at the University of Texas at Austin in 1990 and 1993, respectively. Subsequently he worked at MITL, Princeton as a scientist from 1993-1994. He has been an assistant professor in the Computer Science Department at the University of Illinois at Urbana-Champaign since 1994. He specializes in the areas of database management, distributed systems, and information retrieval. His current research projects are on multimedia analysis, content-based retrieval of multimedia objects, multidimensional indexing, uncertainty management in databases, and concurrency and transaction management. He has authored in excess of 50 research publications in these areas. He is the recipient of the National Science Foundation Career Award and the Bill Gear Outstanding Junior Faculty Award in 1997. He is a member of the IEEE and the IEEE Computer Society.



Thomas S. Huang received his BS degree from National Taiwan University, Taipei, Taiwai, Republic of China; and his MS and ScD degrees from the Massachusetts Institute of Technology, all in electrical engineering. He was on the faculty of the Department of Electrical Engineering at MIT from 1963 to 1973; and the School of Electrical Engineering at Purdue University from 1973 to 1980. While at Purdue, he was also director of its Laboratory for Information and Signal Processing. In 1980, he joined the University of Illinois at Urbana-Champaign, where he is now the William L. Everitt Distinguished Professor of Electrical and Computer Engineering and a research professor in the Coordinated Science Laboratory. He is also head of the Image Formation and Processing Group at the Beckman Institute for Advanced Science and Technology. During his sabbatical leaves, Dr. Huang has worked at the MIT Lincoln Laboratory, the IBM Thomas J. Watson Research Center, and the Rheinishes Landes Museum in Bonn, West Germany, and has held visiting professorship positions at the Swiss Institutes of Technology in Zurich and Lausanne; the University of Hannover in West Germany; INRS-Telecommunications; University of Quebec, Montreal, Canada; and the University of Tokyo, Japan. He has served as a consultant to numerous industrial firms and government agencies, both in the United States and abroad. His professional interests lie in the broad area of information technology, especially the transmission and processing of multidimensional signals. He has published 12 books, and more than 300 papers in network theory, digital filtering, image processing, and computer vision. He is a fellow of the International Association of Pattern Recognition, the IEEE, and the Optical Society of America; and he has received a Guggenheim Fellowship, an A.V. Humboldt Foundation Senior United States Scientist Award, and a fellowship from the Japan Association for the Promotion of Science. He received the IEEE Acoustics, Speech, and Signal Processing Society's Technical Achievement Award in 1987, and the Society Award in 1991. He is a founding editor of the *International Journal Computer Vision, Graphics, and Image Processing*; and an editor of the Springer Series in information sciences, published by Springer-Verlag.