Supporting Ranked Boolean Similarity Queries in MARS [*]

Michael Ortega, Yong Rui, Kaushik Chakrabarti, Alex Warshavsky,

Sharad Mehrotra, and Thomas S. Huang

Department of Computer Science and Beckman Institute

Image Formation and Processing Laboratory

University of Illinois at Urbana-Champaign

Urbana, IL 61801, USA

E-mail:{ortega-b,kaushikc,warshavs,sharad}@cs.uiuc.edu, {yrui, huang}@ifp.uiuc.edu

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) in our group at the University of Illinois [26]. 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 [36, 3]. 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.

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 non-scalability of the approach to large data sets (due to a high degree of manual effort required in defining annotations), the

[*]This work was supported in part by the Army Research Laboratory under Cooperative Agreement No. DAAL01-96-2-0003; in part by NSF/DARPA/NASA Digital Library Initiative Program under Cooperative Agreement No. 94-11318; in part by NSF CISE Research Infrastructure Grant CDA-9624396. Yong Rui is supported in part by CSE, College of Eng. UIUC. Michael Ortega is supported in part by CONACYT grant 89061. The example images used in this article are used with permission from the Fowler Museum of Cultural History at the University of California–Los Angeles. These images were part of an image database delivery project called the Museum Educational Site Licensing Project (MESL), sponsored by the Getty Information Institute. This goal of the two-year MESL project was to test the licensing and delivery of digital images and meta data from seven U. S. museums to seven U. S. universities. The University of Illinois was selected as a participant in the MESL project.

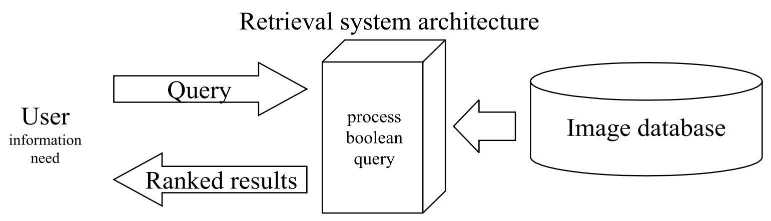


Figure 1: Overall system architecture

approach based on supporting content-based retrieval over visual features has become a promising research direction. This is evidenced by several prototypes [38, 29, 25, 26] and commercial systems [14, 1] that have been built recently. Such an approach can be summarized as follows:

1. Computer vision techniques are used to extract visual features from multimedia objects. For example, color, texture, shape features 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 is used to represent color feature, and the intersection distance is to compute the similarity between 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 users 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 [36]. This article describes how we have generalized these approaches for content-based retrieval over image features in the *Multimedia Analysis and Retrieval System* (MARS) under development in our group at the University of Illinois. An overview of the system architecture is shown in figure 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 [36]. 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 users 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:

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 i^{th} 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 [36], 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 [36], 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 $P(R|d,q)$ which represents the probability that the document d will be deemed relevant to the users 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 also has been extended to allow for ranked retrieval in the text domain (e.g. the p-norm model [35]). Vector-based models and probabilistic retrieval models are in a sense related and provide comparable performance. The primary difference is that while the vector models are ad hoc and based on intuitive reasoning, probability based models have a more rigorous theoretical base.

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. In MARS, an image is represented as a collection of low-level image features (e.g., color, texture, shape and layout features) 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 interpreted as a Boolean expression over image features and a Boolean retrieval model (adapted for retrieval over images) is used to retrieve a set of images ranked based on their similarity of match. 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 be alternatively learned via user interaction by a relevance feedback mechanism. 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:

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

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* values for tolerance for a given feature, δ_i was computed dynamically for each query based on the image collection. 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, in MARS 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 the 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 the 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 in MARS 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 the 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 in MARS and the corresponding distance functions used for comparing similarity of images based on the 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 references [12, 44, 26, 28, 34].

The following features and their representation only describe features currently supported in MARS. 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 [43], color moments [42], color sets [40], have been proposed in the past few years. In this paper we choose the color histogram approach in the HSV color space as our color feature as

- the color histogram is easy to extract and its similarity is fast to compute; and
- the HSV color space has de-correlated and uniform coordinates, which better matches the human perception of color

Furthermore, since the V coordinate in HSV space is easily affected by the lighting condition, we use only HS coordinates to form an 8×4 two-dimensional histogram.

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

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

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. The above intersection based measure of similarity provides an accurate and efficient measure of similarity between two images based on their color [38].

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 [41]. 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 [20]. 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 [20], Tamura texture representation [45], and wavelet based representation [39, 5, 23, 17, 22, 46]. Many research results have shown that the wavelet based texture representation captures the texture property and achieves good performance in texture classification [39]. Therefore, we choose the wavelet approach for texture representation in this paper. In this approach, an input image is fed into a wavelet filter bank and is decomposed into de-correlated sub-bands. Due to the orthogonality of wavelet decomposition, each sub-band captures the property of some scale and orientation of the original image. Specifically, we decompose an image into three wavelet levels; thus having 10 sub-bands. For each sub-band, we extract the standard deviation of the wavelet coefficients. The 10 standard deviations are used as the texture representation for the image.

The similarity between two texture feature vectors is defined as the Euclidean distance in the 10D feature space. To convert this distance in a 10D 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 modified Fourier descriptor (MFD) is described in [34]. The Euclidean distance can be used to measure similarity between two shapes. [34] 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 Image 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 $k \times k$ sub-images. Then the 2D color histograms are extracted from each sub-image, 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 sub-images.

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 totally different physical quantities. For example, one feature can be a mean while the other can be a standard deviation. 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 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 *vector based* feature representation, as in the case of the wavelet texture feature representation. In other cases, such as color histogram intersection, where all the feature elements are defined over the same physical domain, no intra-feature normalization is needed.

For the vector based feature representation, let $F = [f_1, f_2, ..., f_j, ..., f_N]$ 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 the corresponding feature F as $F_i = [f_{i,1}, f_{i,2}, ..., f_{i,j}\ , ..., f_{i,N}]$. 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. One way of normalizing the sequence F_j is to find the maximum and minimum values of F_j and normalize the sequence to [0, 1] as follows:

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

where min_j and max_j refer to the smallest and the biggest value of $f_{i,j}$, $i = 1, 2, \ldots, M$. Although simple, this strategy may not produce the desired normalization. Considering the sequence {1.0, 1.1, 1.2, 1.3, 100}, if we use (2) to normalize the sequence, most of the [0, 1] range will be taken away by a single element 100, and most of the useful information in {1.0, 1.1, 1.2, 1.3} will be warped into a very narrow range.

A better approach is to use Gaussian normalization. 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:

$$f'_{i,j} = \frac{f_{i,j} - m_j}{\sigma_j} \tag{3}$$

Note that after the Gaussian normalization, the probability of a feature element value being in the range of [-1, 1] is 68%. 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%. In practice, we can consider all of the feature element values are within the range of [-1,1] by mapping the out-of-range values to either -1 or 1. The advantage of this normalization process over (2) is that the presence of a few abnormally large or small values does not bias the importance of the feature element in computing the distance between feature vectors.

3.2 Inter-feature Normalization

Intra-feature normalization ensures 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 similarity distance $D_{(i,j)}$ between them:

$$D_{(i,j)} = dist(F_{I_i}, F_{I_j}) \tag{4}$$

$$\begin{aligned} i,j &= 1,...,M, \\ i &\neq j \end{aligned}$$

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 (un-normalized) similarity value between Q and the images in the database. Let $s_1, ..., s_M$ denote the raw similarity values.

4. Normalize the raw similarity values as follows:

$$s'_i = \frac{s_i - m}{3\sigma} \tag{5}$$

As explained in the intra-feature normalization section, this Gaussian normalization will ensure 99% of s'_i to be within the range of [-1,1]. An additional shift will guarantee that 99% of similarity values are within [0,1]:

$$s''_i = \frac{s'_i + 1}{2} \tag{6}$$

After this shift, in practice, we can consider all the values are 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:

$$similarity_i = 1 - s''_i \tag{7}$$

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 the least similarity.

3.3 Weights for feature vectors and feature elements

On completion of the intra- and inter-feature normalization processes discussed above, the feature elements within a feature as well as the features within a composite query are of equal weights. This *objective* equality allows us to further associate *subjective* unequal intra- and inter-feature weights for a particular object and a particular query.

Intra-feature weights associated with a feature vector reflect the individual contributions of the feature elements to the feature vector. For example, in the wavelet texture representation, we know that the mean of a sub-band may be corrupted by the lighting condition, while the standard deviation of a sub-band is independent of the lighting condition. Therefore the user may want to put more weight on the standard deviation feature element, and less weight on the mean feature element. The support of the different intra-feature weights enables the system to have more reliable feature representation and thus better retrieval performance.

Inter-feature weights associated with a composite query reflect the user's emphasis on each atomic feature in the composite query. For example, for a composite query based on color and texture, a user may assign color a weight of 90% and 10% for texture. The support of different inter-feature weights enables the user to specify his information need more precisely. This method is further discussed in section 4.2 where result propagation is discussed.

In MARS, we have explored techniques to automatically associate subjective weights with feature elements and feature vectors of both the object and the query. Associating subjective weights improves the retrieval performance considerably [31, 33, 32]. Our experiments demonstrate the effect of weighting on retrieval performance (refer to Section 5.3).

4 Retrieval Models Used In MARS

This section discusses how MARS supports Boolean queries based on the simple feature similarity values. MARS supports two mechanisms for generating the ranking of Boolean queries – the first is based on the fuzzy interpretation of the distance 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 $I_1, I_2, \ldots, I_m$. Features over the images are denoted by $F_1, F_2, \ldots, F_r$, where F_i denotes both the name of the feature as well as the domain of values that the feature can take. Instances of feature F_i are denoted by f_i. 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. Query variables are denoted by $v_1, v_2, \ldots, v_n \mid v_j \in F_i$ so each v_j refers to an instance of a feature F_i (an f_i). Each v_j is used to rank images in the collection based on the feature domain of f_i (F_i), that is v_j's domain. In certain contexts, the query variable v_j can also denote the set of all images of F_i ranked based on the f_i value assigned to v_j. For example, say that F_i is the set of all wavelet texture vectors in the collection, if v_j is f_5, then v_j can be interpreted as being both, the wavelet texture vector corresponding to image 5 and the ranked list of all $\langle I, D_{F_i}(I)\rangle$ with D being the similarity function that applies to texture. A query $Q(v_1, v_2, \ldots, v_n)$ 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, non–leaf nodes are of either of three forms: $\wedge(v_1, v_2, \ldots, v_n)$ which is a conjunction of positive literals; $\wedge(v_1, v_2, \ldots, v_m, \neg v_{m+1} \ldots, \neg v_n)$, which is a conjunction consisting of both positive and negative literals; and $\vee(v_1, v_2, \ldots, v_n)$ which is a disjunction of positive literals. Notice that we do not consider an unguarded negation or a negation in the disjunction. Presence of an unguarded negation or negation in a disjunction does not make intuitive sense. Typically, a very large number of entries will satisfy a negation

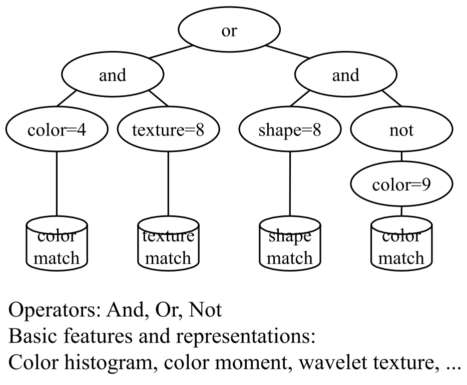


Figure 2: Sample query tree for the query

query virtually producing the universe of the collection. We therefore consider negation only when it appears within a conjunctive query to rank an entry on the positive feature discriminated by the negated feature. a boolean query supported in MARS: $Q(v_1, v_2) = v_1 \wedge v_2$ is query where v_1 has a value equal to the color histogram associated with image I_2 and v_2 has a value of the texture feature associated with I_5. Thus, the query Q represents the desire to retrieve images whose color matches that of image I_2 *and* whose texture matches that of image I_5. Figure 2 shows an example query $Q(v_1, v_2, v_3, v_4) = (v_1 \wedge v_2) \vee (v_3 \wedge \neg v_4)$ in its tree representation.

4.1 Finding the Best N Matches

While the Boolean retrieval model provide 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 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 in MARS 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 provides 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

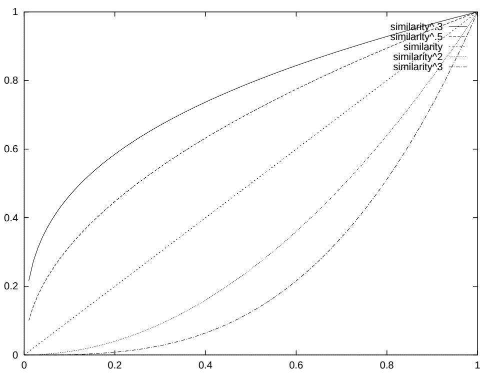


Figure 3: Various samples for similarity mappings

probabilistic model). For example, in the fuzzy model, a node n in a tree provides to its parent a ranked list of $\langle I, similarity_{Q_N}(I)\rangle$, 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

As suggested in section 3.3, one feature can receive more importance than the other according to the user's perception. The user can assign desired importance to any feature by a process known as *feature weighting*. Traditionally, retrieval systems use feature weights as a linear scaling factor. Each feature produces a similarity value in the range [0,1] and percentages are assigned to the importance of each feature (adjusted for a total of 100%). The final similarity score for an image is a weighed sum of these features according to:

$$\begin{aligned} similarity(I) &= \sum_{i=1}^{i=N} w_i \times D_{F_i}(I) \qquad (8) \\ where \sum_{i=1}^{i=N} w_i &= 1 \end{aligned}$$

While the above approach of associating feature weights in similarity computations is suited for a vector model, in the case of a boolean query tree, where weights are associated with nodes of a tree, the approach may have undesirable consequences. 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 based on the child nodes. For example if the fuzzy model is used, and the node is $\wedge$, the similarity computation is done as $similarity_\wedge = \min(D_{F_i}, D_{F_j})$. If F_i is carries a weight α, F_j a weight β and the above method is used, then $similarity_\wedge = \min(\alpha \times D_{F_i}, \beta \times D_{F_j})$ will be in the range $[0, \min(\alpha, \beta)]$ which is distinct from $[0, 1]$ in general. One approach is to scale this range back into $[0, 1]$, but this may defeat the purpose of linear weights. Instead, we use a mapping function from $[0, 1] \rightarrow [0, 1]$ of the form

$$similarity' = similarity^{weight}, 0 < weight < \infty \qquad (9)$$

which preserves the range boundaries [0,1] and yet boosts or degrades the similarity in a smooth way. Sample mappings are shown in figure 3. Notice in this case the meaning of weight is reversed. A higher weight will reduce rather than improve similarity.

4.3 Leaf node evaluation

Each leaf node in the query tree corresponds to a selection operation on a single feature. For example, in Figure 2, the leaf nodes correspond to selection operations based on color, texture and shape features (the selection predicates being $\mathit{color} = 4$, $\mathit{texture} = 8$, $\mathit{shape} = 8$ 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 D 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 [13]) have been proposed recently (e.g., R-trees [19], R+-trees [37], R*-trees [2], k-d-B-trees [30], hB-trees [10], TV-trees [24], SS-trees [48], vp-trees [7], M-trees [8].) Any such indexing mechanism can be used for indexing the feature vectors in MARS. In MARS, we have developed an indexing mechanism based on dynamic incremental clustering which scales to the high dimensional multimedia feature spaces, supports arbitrary distance (or similarity) measures among feature vectors and supports nearest neighbor search [4]. In this paper, we concentrate on developing techniques of evaluation of query nodes in MARS 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 provides 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, any boolean query in MARS produces a ranked list of $\langle I, \mathit{similarity}_Q(I)\rangle$ based on the similarity of each image to the query Q. Our evaluation model for the rest of the paper is as follows:

- Each node N defines a sub query rooted at node N denoted be Q_N.
- Each node N returns a list of $\rho_i = \langle I_j, \mathit{similarity}^i_{Q_N}(I_j)\rangle$ to its parent where:
  - $i = 1, 2, \ldots, n$ is the sequence number in which the ρ's 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 sub query rooted at Q_N.
  - for any two $\rho_i = \langle I_j, similarity^i_{Q_N}(I_j)\rangle$, and $\rho_k = \langle I_{j'}, similarity^k_{Q_N}(I_{j'})\rangle$ 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 sub query Q_N will have higher similarity than any pair returned later for the same sub query. In other words, ρ's are returned in sorted order by similarity.

- Evaluation of a sub query 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 $\rho = \langle I_j, similarity_{Q_N}(I_j)\rangle$ that would be obtained from a stream satisfying the above criteria.

- A *stream* of ρ's will support the operations
  - *PeekNext* 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)* that performs random access to image j and returns a $\rho = \langle I_j, similarity_{Q_N}(I_j)\rangle$, that is, the image id and similarity pair corresponding to image j based on the sub query 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. Figure 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 corresponding to both streams have been observed. We refer to this rectangle as the *Observed Area*

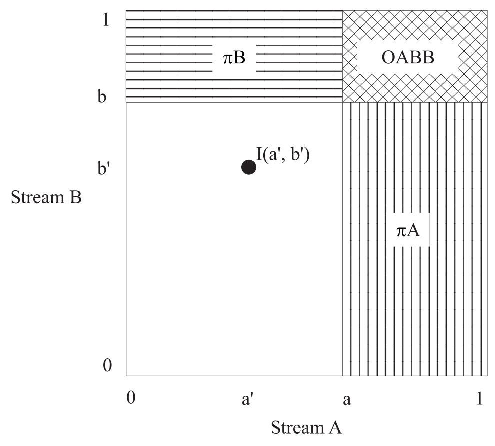


Figure 4: A sample OABB Rectangle

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 no dot overlap. The same is true for the projection of OABB onto the B axis (the rectangle is called πB). The union of these rectangles denotes the images of which we have partial knowledge of their location in this 2-d space (i.e. at least one co-ordinate 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:

$$\begin{array}{lll} \textbf{And} & f_{Q=Q_1 \wedge Q_2}(I) & = \min(f_{Q_1}(I), f_{Q_2}(I)) \\ \textbf{Or} & f_{Q=Q_1 \vee Q_2}(I) & = \max(f_{Q_1}(I), f_{Q_2}(I)) \\ \textbf{Not} & f_{Q=\neg Q_1}(I) & = 1 - f_{Q_1}(I) \end{array}$$

Consider for example a query Q:

$$Q = (v_1 \vee v_2 \vee v_3) \wedge (v_4 \vee (v_5 \wedge v_1)) \tag{10}$$

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

$$\begin{aligned} f_Q(I) &= min(max(f_{v_1}(I), f_{v_2}(I), f_{v_3}(I)), \\ &\quad max(f_{v_4}(I), min(f_{v_5}(I), f_{v_1}(I)))) \end{aligned} \tag{11}$$

The value $f_{v_i}(I)$ in (11) 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 $f_Q(I)$ 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 at 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.

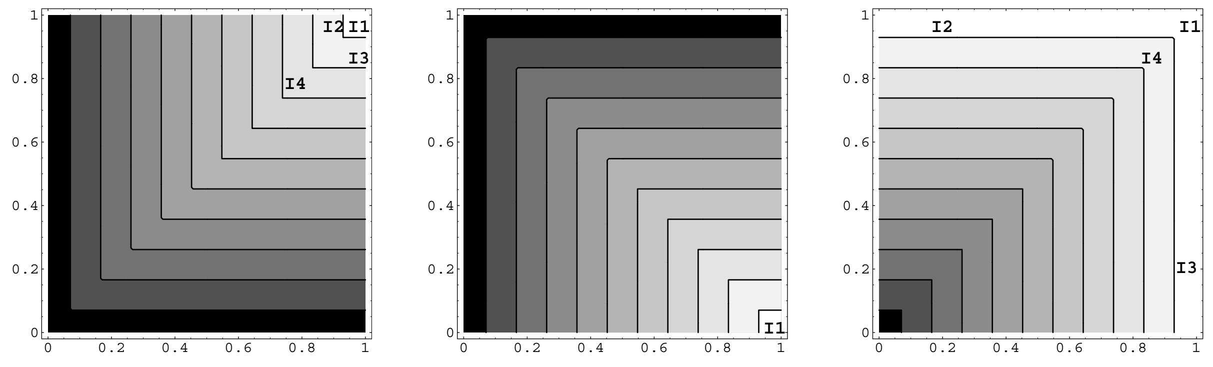
In describing the algorithms the following notation is used.

- 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 figure 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 an implementation.

4.6.1 Conjunctive Query with Positive Sub queries

The algorithm shown in Figure 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.



a) Fuzzy *and* operator b) Fuzzy *and not* operator c) Fuzzy *or* operator

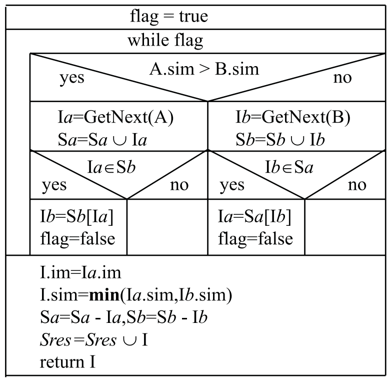
Figure 5: Contour graphs for fuzzy operators. Whiter is higher, darker is lower similarity value.

The operation performed in a binary operator node can be viewed as a function $f(x \in [0,1], y \in [0,1] \rightarrow [0,1]$. As an aid to explain the algorithm, we use contour plots that show the value of $f(x,y)$. These plots depict lines along which the value of f 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 f (degree of membership) are in the white areas, the darker the region, the lower the value. Figure 5a) shows the plot that corresponds to the fuzzy *and* operator.

Imagine an overlay of Figure 4 on top of Figure 5(a). 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 is the larger of a or 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 definitely are 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 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 (πA) and S_b (πB) 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 $\mid S_a \cup S_b \mid$ will never exceed the size of the feature collection. If it did, the intersection would not be empty and results would be produced. 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.

```
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($A$)
      $S_a = S_a \cup I_a$
      if $I_a.image \in S_b$ then ;image already seen in $B$
        $I_b$= image $S_b[I_a.image]$
        exit loop
      end if
    else
    if $I_b.degree > I_a.degree$ then
      $I_b$= GetNext($B$)
      $S_b = S_b \cup I_b$
      if $I_b.image \in S_a$ 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 $S_b$
  $I.image = I_a.image$
  $I.degree = \min(I_a.degree, I_b.degree)$
  $S_a = S_a - I_a, S_b = S_b - I_b, S_{res} = S_{res} \cup I$
  return $I$
```



a) Pseudo code b) Flow graph

Figure 6: Algorithm returning the next best for the fuzzy *and* case.

4.6.2 Conjunctive Query with Negative Sub query

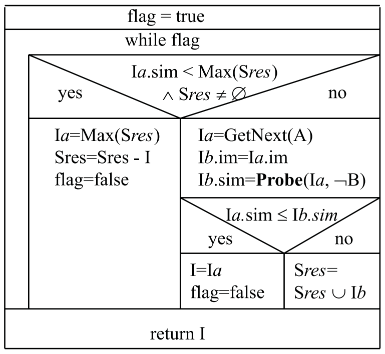
We next present the algorithm for computing the query $Q = A \wedge \neg B$, it is presented in Figure 7. Figure 5b) shows the contour plot that corresponds to this query. A strategy similar to the previous subsection could be used if traversing the stream B in reverse order was possible. This implies a furthest neighbor query that is not supported. The positive term is used to guide the search and the negative sub query 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:

- $S_{aux} \neq \emptyset \wedge \text{Peek}(A).degree \leq \text{MaximumDegree}(S_{aux})$ 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.
- $(S_{aux} = \emptyset \vee \text{Peek}(A).degree > \text{MaximumDegree}(S_{aux})) \wedge \text{Peek}(A).degree \leq \text{Probe}(\text{Peek}(A).id).degree$ 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 sub query. The answer is the best image from A.
- $(S_{aux} = \emptyset \vee \text{Peek}(A).degree > \text{MaximumDegree}(S_{aux})) \wedge \text{Peek}(A).degree > \text{Probe}(\text{Peek}(A).id).degree$

```
Algorithm GetNextAnd_Not_Fuzzy($A$, $B$)
;returns:  next best image in $A$ and not $B$
  while (TRUE)
    $I_a$= Peek ($A$)
    if $S_{aux} \neq \emptyset \wedge I_a.degree <$ 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($I_a$ , $\neg B$)
      if $I_a.degree \leq I_b.degree$ then
        $I = I_a$
        exit loop
      else
        $S_{aux} = S_{aux} \cup I_b$
      end if
    end if
  end while
  $S_{res} = S_{res} \cup I$
  return $I$
```

a) Pseudo code



b) Flow graph

Figure 7: Algorithm returning the next best for the fuzzy *and not* case.

there is no better candidate on hold and the degree of the best image from A is higher than the probe on the negative sub query. 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

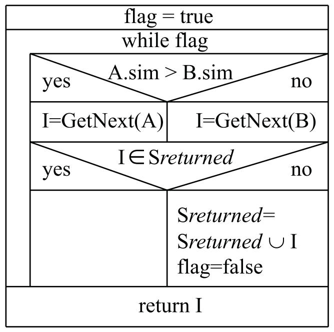
The algorithm shown in Figure 8 computes the set of images ranked on their degree of membership to the query $Q = A \vee B$, given input streams A and B which are ranked based on the degree of membership of images in A and B.

Figure 5c) shows the contour plot for the disjunctive fuzzy operator. By overlaying Figure 4 on Figure 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 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 sub queries and thus can safely ignore the other. Since 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.

```
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 $I.image \in S_{res}$ then
      flag = TRUE
    end if
  end while
  $S_{res} = S_{res} \cup I$
  return I
```

a) Pseudo code



b) Flow graph

Figure 8: Algorithm returning the next best for the fuzzy *or* case.

4.7 Probabilistic Boolean Model

Let $Q(v_1, v_2, \ldots, v_n)$ be a query and I be an image. In the probabilistic Boolean model, the similarity $D(I, v_i)$ between the query variable v_i and the corresponding feature in the image is taken to be the probability of the image I matching the query variable v_i, denoted by $P(v_i|I)$. 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 $P(Q(v_1, v_2, \ldots, v_n)|I)$) which is in turn used to rank the images. To enable computation of $P(Q(v1, v_2, \ldots, v_n)|I)$, an assumption of *independence* is made. That is, we assume that for all variables v_i, v_j following holds:

$$P(v_i \wedge v_j|I) = P(v_i|I) \times P(v_j|I) \tag{12}$$

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 $P(Q|I)$. If Q is a disjunction ($Q = Q_1 \vee Q_2$), following the laws of probability, $P(Q_1 \vee Q_2|I)$ can be estimated as follows:

$$P(Q_1 \vee Q_2|I) = P(Q_1|I) + P(Q_2|I) - P(Q_1 \wedge Q_2|I) \tag{13}$$

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:

$$P(\neg Q_1) = 1 - P(Q_1) \tag{14}$$

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

$$P(Q_1 \wedge Q_2) = P(Q_1) \times P(Q_2) \tag{15}$$

Our retrieval results (see section 5) show that even if query terms are considered as independent, the resulting retrieval performance is quite good. Developing a dependence model and incorporating efficient evaluation techniques is an important extension to our current work.

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].

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

It seems odd to compute the same results using probability when the probability of a feature is derived from the fuzzy interpretation itself. Certainly, the results are very similar given simple queries (one operation only i.e. *and* or *or*), but when more levels are present, the results may vary.

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 $(I.prob)$.

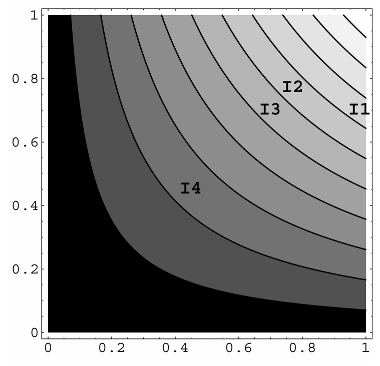
- A and B are assumed to be streams as defined in section 4.4.

- 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 figure 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.

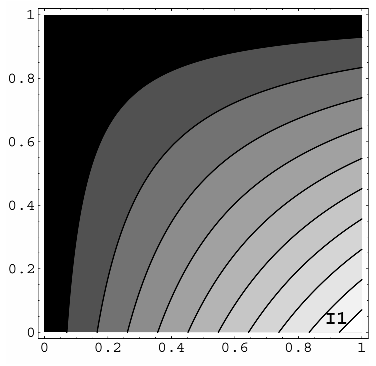
The following three subsections describe the algorithms used to implement the above shown operations in an efficient manner. For clarity purposes, when describing the algorithms below we omit some critical error and boundary checking which needs to be considered in an implementation.

4.8.1 Conjunctive Query with Positive Sub queries

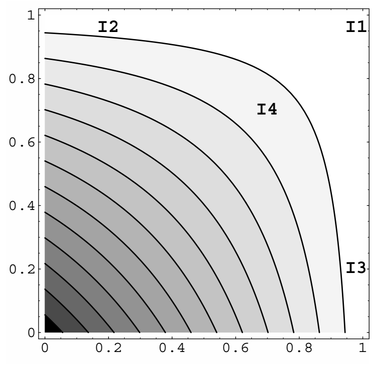
The algorithm in figure 10 computes the set of images ranked on their probability of match to the query $Q = A \wedge B$, given input streams A and B which are ranked based on their matching probability of images in A and B.



a) Probabilistic *and* operator



b) Probabilistic *and not* operator



c) Probabilistic *or* operator

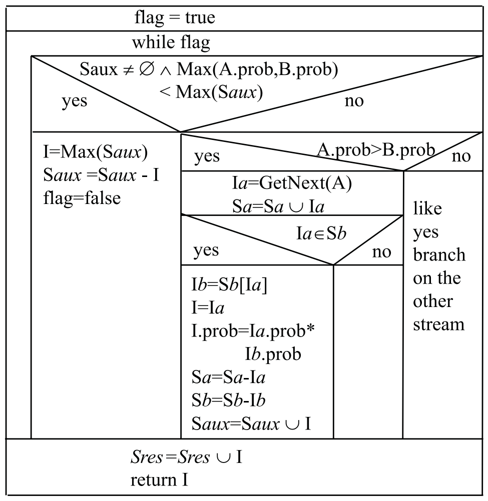
Figure 9: Contour graphs for probabilistic operators. Whiter is higher, darker is lower similarity value.

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 figure 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 an 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 figure 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.

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

```
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 $S_{aux} \neq \emptyset \wedge$ 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($A$)
        $S_a$= $S_a \cup I_a$
        if $I_a \in S_b$ then
          $I_b$= image from $S_b$ equivalent to $I_a$
          $I$= $I_a$
          $I$.prob = $I_a$.prob $\times$ $I_b$.prob
          $S_a$= $S_a$-$I_a$, $S_b$= $S_b$-$I_b$, $S_{aux}$ = $S_{aux} \cup I$
        end if
      else
        ; symmetric code to then branch
      end if
    end if
  end while
  $S_{res}$= $S_{res} \cup I$
  return $I$
```

a) Pseudo code



b) Flow graph

Figure 10: Algorithm that implements the *and* operator for the probabilistic case.

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.prob$_A$ and I_k.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:

$$\begin{array}{ll} \text{Peek}(A)\text{.prob} \times I_k\text{.prob}_B & \text{if } I_k\text{.prob}_B \text{ is known, or} \\ \text{Peek}(B)\text{.prob} \times I_k\text{.prob}_A & \text{if } I_k\text{.prob}_A \text{ is known} \end{array} \qquad (16)$$

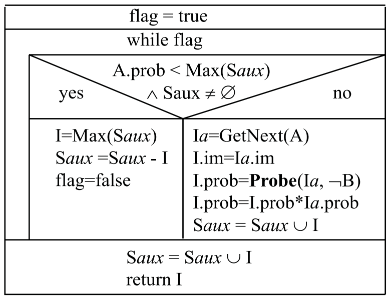
This more sophisticated criteria is not incorporated in figure 10, instead the simpler criteria described above is included.

4.8.2 Conjunctive Query with Negative Sub query

We next develop the algorithm for computing the query $Q = A \wedge \neg B$, it is shown in figure 11. The algorithm is different compared to the one developed for the conjunctive query with no negative sub query. As described for the fuzzy model, a similar method to the conjunctive query with only positive sub queries 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

```
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 $S_{aux} \neq \emptyset \wedge I_a$.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($I, \neg B$)
      $I$.prob = $I$.prob $\times$ $I_b$.prob
      $S_{aux}$ = $S_{aux}$ $\cup$ $I$
    end if
  end while
  $S_{res}$= $S_{res}$$\cup$ $I$
  return $I$
```

a) Pseudo code



b) Flow graph

Figure 11: Algorithm that implements the *and not* operator for the probabilistic case.

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 sub query $\neg B$ 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.

4.8.3 Disjunctive Query

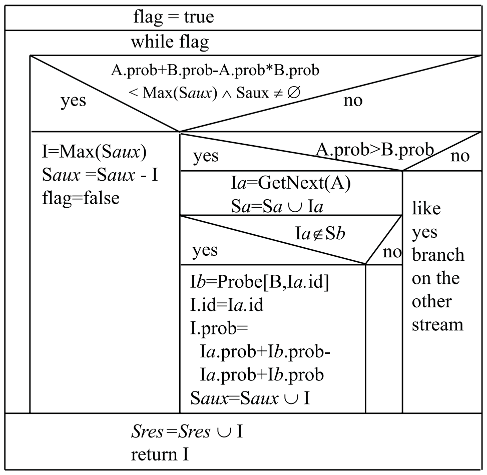
Finally, to compute a disjunctive query node, we need the algorithm shown in figure 12. Disjunctive queries are hard to compute in this case. Consider figure 9c), images I_1, I_2 and I_3 have very similar probabilities. In the fuzzy case, the iso similarity curves were parallel to the axes and we could 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,

- the final probability does depend on all the query terms while in the fuzzy model, only the best one is relevant
- 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.

```
Algorithm GetNextOr_Probability($A$, $B$)
;returns:  next best image in $A$ or $B$
  flag = TRUE
  while (flag)
    $I_a = \texttt{Peek}(A), I_b = \texttt{Peek}(B)$
    if $S_{aux} \neq \emptyset \wedge I_a.\texttt{prob} + I_b.\texttt{prob} - I_a.\texttt{prob} \times I_b.\texttt{prob} \leq$
          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($A$)
        $S_a$= $S_a \cup I_a$
        if $I_a \notin S_b$ then ; do a probe
          $I_b = \texttt{Probe}(B, I_a.\texttt{id})$
          $I.id = I_a.id$
          $I.\texttt{prob} = I_a.\texttt{prob} + I_b.\texttt{prob} - I_a.\texttt{prob} \times I_b.\texttt{prob}$
          $S_{aux} = S_{aux} \cup I$
        end if
      else
        ; symmetric code to then branch
      end if
    end if
  end while
  $S_{res}$= $S_{res} \cup I$
  return $I$
```



a) Pseudo code b) Flow graph

Figure 12: Algorithm that implements the *or* operator for the probabilistic case.

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 sub queries. So far, only algorithms based on negation have required this and then only for the negation operator. If probing on sub queries 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 is known, an upper bound on the final probability can be established by:

$$\begin{aligned} upper(I_k) &= \text{Peek}(A).\text{prob} + I_k.\text{prob}_B - \text{Peek}(A).\text{prob} \times I_k.\text{prob}_B \quad \text{if } I_k.\text{prob}_B \text{ is known, or} \\ upper(I_k) &= \text{Peek}(B).\text{prob} + I_k.\text{prob}_A - \text{Peek}(B).\text{prob} \times I_k.\text{prob}_A \quad \text{if } I_k.\text{prob}_A \text{ is known} \end{aligned} \tag{17}$$

And the known probability component is a lower bound. 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 images can exceed it, and no combination of any images left in A and B can exceed $lower(I_k)$.

4.9 Comparison of algorithms to other work

Recently, [11] 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 being developed at the IBM Almaden Research Center [9]. A function F is monotonic if $F(x_1, \ldots, x_m) \leq F(x'_1, \ldots, x'_m)$ if $x_i \leq x'_i$ 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 this algorithm, each stream outputs data items in sorted order based on degree of membership until there is a set of L of at at least k objects ($\mid L \mid = k$) such that each stream has output all the members in L. Since the terminating condition is based on the number of items in the intersection set L, the algorithm does not guarantee that L is the answer set. Hence the above step is followed by a probing process (random access): for each item output by any of the streams, each stream is probed to retrieve the membership value (unless the item was already output by that sream). Then the membership value μ_Q of each item with respect to the query is computed by applying the scoring function and followed by sorting them based on μ_Q and the top k are returned. On the other hand, since the terminating condition in MARS is based on the degree of membership of the retrieved items from each stream, MARS guarantees that the intersection set generated is the final answer set (see section 4.6.1). Hence MARS, unlike Garlic, does not need to perform the random access for *each* data item retrieved from *each* stream. According to the performance cost model proposed in [11], the total database access cost due to random access can be much higher compared to the total cost due to sorted access. Specifically, the worst case total cost of random access is the number of input streams times the total cost of sorted access. Furthermore, the final answers in MARS are generated one by one in ranked order. Thus MARS follows a *demand-driven data flow* approach [16], i.e. a data item is never produced until it is demanded. So the wait time of intermediate answer items in a temporary file or buffer between operators in the query tree is minimized. This model is efficient in its time-space product memory costs [16]. 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 [15].

Another approach to optimizing query processing over multimedia repositories has been proposed in [6]. It presents a strategy to optimize queries when users specify thresholds on the grade of match of acceptable objects as filter conditions. It uses the results in [11] 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 [11]. 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 datasets to measure the performance of the retrieval models and query processing algorithms used in MARS. We present the results of our experiments in this section. We first 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 used the following two metrics to measure the retrieval performance: *precision* and *recall* [36, 3]. 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, there exists two subsets of documents in the collection. One subset is the set of relevant documents i.e. for each query, it is possible to partition the collection into relevant and non relevant documents based on the user's criteria of relevance. 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%) means that all retrieved images are relevant.

$$\frac{|relevant \bigcap retrieved|}{|retrieved|} \tag{18}$$

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

$$\frac{|relevant \bigcap retrieved|}{|relevant|} \tag{19}$$

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 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 can be done easily in MARS since the query processing algorithms have been implemented as a pipeline in MARS.

5.2 Description of data sets used

We have conducted experiments on the following two datasets. The first dataset comprises of 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 groups ranged in their cardinality from 9 to 33 images.

The other dataset comprises of video sequences. These have been segmented into individual frames for a total of about 65000 images. To describe the data set and the query set, we introduce a few definitions. A *shot* is defined as a set of temporally contiguous frames having similar visual content. A *scene*, on the other hand, is defined as a collection of shots that have a single semantic content. Thus, while shots have distinct physical boundaries, scenes have semantic boundaries. The shots in a scene commonly come from few different sources, typically two to four camera sources. We define a *group* of shots within a scene as those from the same source. Note that the shots in a group may not be temporally contiguous. Thus a scene is a sequence of interleaved groups of shots where each group is from the same source. The relevant set of frames for a query frame F_Q is defined to be all frames constituting a shot present in the same group as the shot which F_Q belong to. As in the Fowler collection, five groups of shots were identified for each scene and several queries were constructed and executed for each shot.

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. Figure 13(a) 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. Note that as explained in section 4.2, higher weights indicate less emphasis while lower weights imply more emphasis.

We also conducted experiments to observe the impact of the retrieval model used to evaluate the queries. We observed that the fuzzy and probabilistic interpretation of the same query yields different results. Figure 13(b) 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.

Figure 13(c) 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 Figure 13(d). In general, the probabilistic model outperforms the fuzzy model.

5.3.2 Video collection

We performed a series of experiments over video data. We executed a video query with two terms to compare the performance of the two retrieval models. The result is shown in Figure 13(e). The result shows that the probabilistic consistently outperforms the fuzzy models. Figure 13f) shows the result of a different video query. Again, the probabilistic interpretation shows better performance than the fuzzy interpretation.

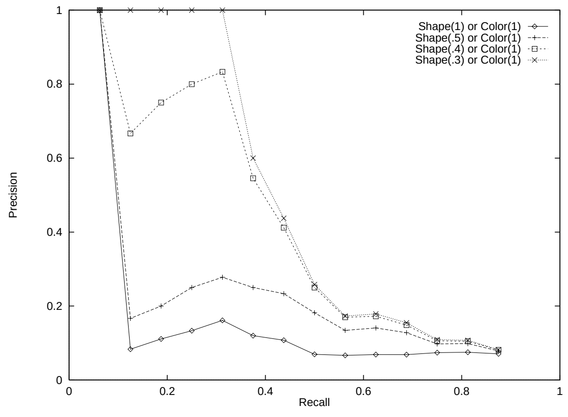
5.4 Analysis of data

Note the graphs shown are not always monotonic. The precision is expected to monotonically decrease as 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 non relevant 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.

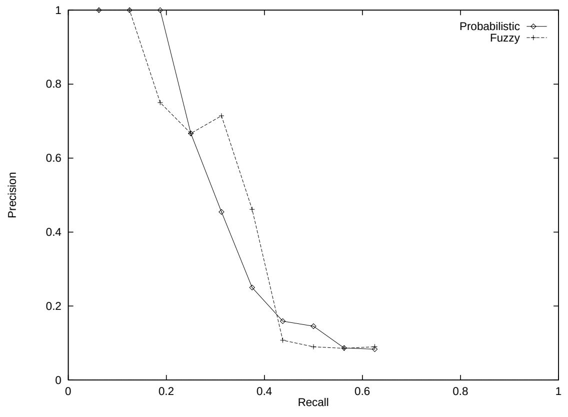
We observe from Figure 13(a) that the weighting of features can improve performance dramatically . The weights for the queries were determined subjectively and several combinations were tried. Automatic learning of these weights in MARS is an interesting extension of this work. We also observed (from Figure 13(c)) that complex weighting strategies may not always improve performance significantly. Determining when weighting can have significant effect is not obvious. We observed that the probabilistic model is superior to the fuzzy model for video queries. The probabilistic model is expected to perform better since the results of probabilistic operations involves on all its operands while the results of the min and max operations in the fuzzy model involves only one of the operands and hence carries less information. However, the distinction is not so clear for the Fowler dataset. This can be accounted to the small size of the dataset (286 images) and probably the clear distinction can only be made large enough datasets i.e. when the size of the dataset is far greater than the sizes of the relevant and retrieved sets.

6 Related Work

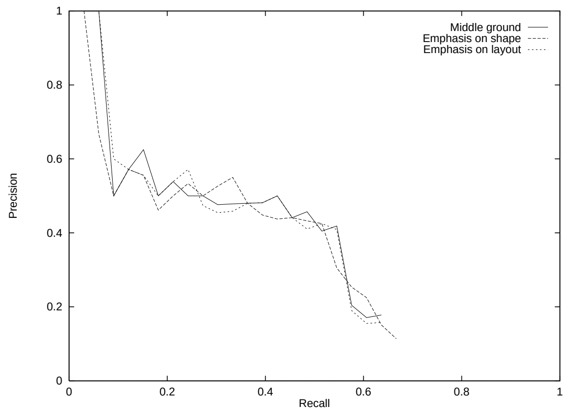
Content-based retrieval of images is an active area of research being pursued independently by many research teams. Similar to MARS, most existing content-based image retrieval systems also extract low-level image features like color, texture, shape, and structure [12, 44, 26, 14, 28, 25, 29, 38]. However, 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.



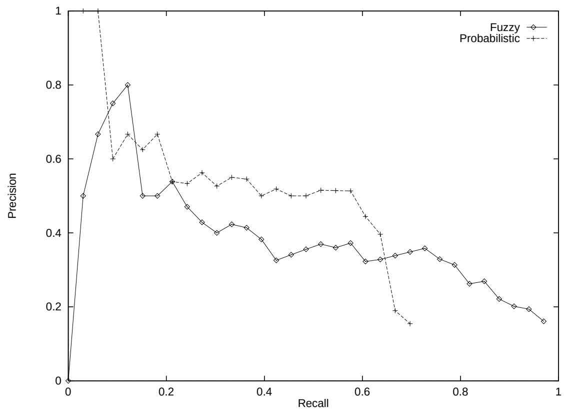
a) Effects of varying the weighting on a query



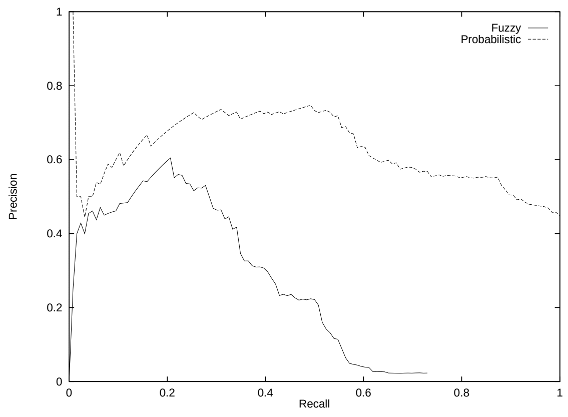
b) Fuzzy vs. Probabilistic performance for same simple query



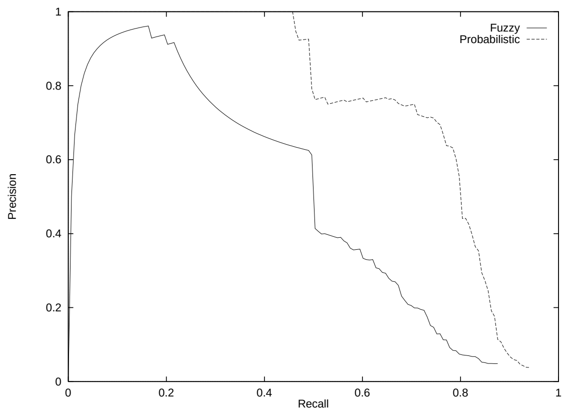
c) Complex query with different weights



d) Fuzzy vs. probabilistic for same complex query



e) Video query. Comparing probability and fuzzy models



f) Video query. Comparing probability and fuzzy models

Figure 13: Experimental result graphs

Some commercial systems have been developed. QBIC [14], 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 [44], 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 multi-dimensional 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 [18].

Photobook [29] is a set of interactive tools for browsing and searching images developed at the MIT Media Lab. Photobook consists of three sub-books, from which shape, texture, and face features are extracted respectively. Users can then query based on corresponding features in each of the three sub-books. In its more recent version of Photobook, FourEyes, Picard et al. proposed to include human in the image annotation and retrieval loop [27]. 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 [21] the authors 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 interpretation 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 our group at the University of Illinois [26]. 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 [36, 3, 47]. 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. Finally, we are also exploring the use of relevance feedback techniques in our extended boolean model.

References

[1] Jeffrey R. Bach, Charles Fuller, Amarnath Gupta, Arun Hampapur, Bradley Horowitz, Rich Humphrey, Ramesh Jain, and Chiao fe Shu. The virage image search engine: An open framework for image management. In *SPIE Storage and Retrieval for Still Image and Video Databases IV*.

[2] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of ACM SIGMOD*, May 1990.

[3] J. P. Callan, W. B. Croft, and S. M. Harding. The inquery retrieval system. In *In Proceedings of the Third International Conference on Database and Expert Systems Applications*, Valencia, Spain, 1992.

[4] Kaushik Chakrabarti, Yong Rui, Sharad Mehrotra, and Thomas S. Huang. Dynamic clustering approach to indexing high dimensional multimedia data. *Technical Report TR-MARS-97-10, Department of Computer Science, University of Illino is*, 1997.

[5] Tianhorng Chang and C.-C. Jay Kuo. Texture analysis and classification with tree-structured wavelet transform. *IEEE Trans. Image Proc.*, 2(4):429–441, October 1993.

[6] Surajit Chaudhari and Luis Gravano. Optimizing queries over multimedia repositories. *Proc. of SIGMOD*, 1996.

[7] T. Chiueh. Content-based image indexing. *Proc. of VLDB*, 1994.

[8] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. *Proc. of VLDB*, 1997.

[9] W. F. Cody et. al. Querying multimedia data from multimedia repositories by content: the garlic project. *Proc. of Visual Database Systems (VDB-3)*, 1995.

[10] G. Evangelidis, D. Lomet, and B. Salzberg. The hb^{π}-tree: A modified hb-tree supporting concurrency, recovery and node consolidation. In *Proceedings of VLDB*, 1995.

[11] Ronald Fagin. Combining fuzzy information from multiple systems. *Proc. of the 15th ACM Symp. on PODS*, 1996.

[12] C. Faloutsos, M. Flocker, W. Niblack, D. Petkovic, W. Equitz, and R. Barber. Efficient and effective querying by image content. Technical Report RJ 9453 (83074), IBM Research Report, Aug. 1993.

[13] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. *Proceedings of SIGMOD*, 1994.

[14] M. Flickner et al. Query by image and video content: The qbic system. *IEEE Computer*, September 1995.

[15] Goetz Graefe. Encapsulation of parallelism in the volcano query processing system. *Proc. of ACM SIGMOD*, 1990.

[16] Goetz Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys, Vol. 25, No. 2, 1993*, 1996.

[17] M. H. Gross, R. Koch, L. Lippert, and A. Dreger. Multiscale image texture analysis in wavelet spaces. In *Proc. IEEE Int. Conf. on Image Proc.*, 1994.

[18] Amarnath Gupta and Ramesh Jain. Visual information retrieval. *Communications of the ACM*, 40(5), 1997.

[19] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Conf., pp. 47–57.*, 1984.

[20] Robert M. Haralick, K. Shanmugam, and Its'hak Dinstein. Texture features for image classification. *IEEE Trans. on Sys, Man, and Cyb*, SMC-3(6), 1973.

[21] Anil K. Jain and Aditya Vailaya. Image retrieval using color and shape. *Pattern Recognition*, 29(8), 1996.

[22] Amlan Kundu and Jia-Lin Chen. Texture classification using qmf bank-based subband decomposition. *CVGIP: Graphical Models and Image Processing*, 54(5):369–384, September 1992.

[23] Andrew Laine and Jian Fan. Texture classification by wavelet packet signatures. *IEEE Trans. Patt. Recog. and Mach. Intell.*, 15(11):1186–1191, 1993.

[24] H. V. Lin, K.and Jagadish and C. Faloutsos. The TV-tree - an index stucture for high dimensional data. In *VLDB Journal*, 1994.

[25] B.S. Manjunath and W.Y. Ma. Texture features for browsing and retrieval of image data. Technical Report TR-95-06, CIPR, July 1995.

[26] Sharad Mehrotra, Kaushik Chakrabarti, Michael Ortega, Yong Rui, and Thomas S. Huang. Towards extending information retrieval techniques for multimedia retrieval. In *3rd International Workshop on Multimedia Information Systems, Como, Italy*, 1997.

[27] T. P. Minka and R. W. Picard. Interactive learning using a "society of models". Technical Report 349, MIT Media Lab, 1996.

[28] Makoto Miyahara et al. Mathematical transform of (r,g,b) color data to munsell (h,v,c) color data. In *SPIE Visual Communications and Image Processing'88*, volume 1001, 1988. 650.

[29] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook: Tools for content-based manipulation of image databases. In *Proc. Storage and Retrieval for Image and Video Databases II*, volume 2(185), pages 34–47, Bellingham, Wash, 1994. SPIE.

[30] J. T. Robinson. The k-d-b-tree: A search structure for large multidimensional dynamic indexes. In *Proc. ACM SIGMOD*, 1981.

[31] Yong Rui, Thomas S. Huang, and Sharad Mehrotra. Content-based image retrieval with relevance feedback in mars. *IEEE Proc. of Int. Conf. on Image Processing (ICIP)*, 1997.

[32] Yong Rui, Thomas S. Huang, Sharad Mehrotra, and Michael Ortega. Automatic matching tool selection via relevance feedback in mars. *to appear in the 2nd Int. Conf. on Visual Information Systems*, 1997.

[33] Yong Rui, Thomas S. Huang, Sharad Mehrotra, and Michael Ortega. A relevance feedback architecture in content-based multimedia information retrieval systems. In *Proceedings of the IEEE Workshop on Content-based Access of Image and Video Libraries.* IEEE, 1997.

[34] Yong Rui, Alfred C. She, and Thomas S. Huang. Modified fourier descriptors for shape representation – a practical approach. In *Proceeding of First International Workshop on Image Databases and Multi Media Search*, 1996. Amsterdam, The Netherlands.

[35] G. Salton, Edward Fox, and E. Voorhees. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, November 1983.

[36] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* McGraw Hill Computer Science Series, 1983.

[37] T. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-tree: A dynamic index for multi-dimensional objects. In *Proc. VLDB*, 1987.

[38] John R. Smith and Shih-Fu Chang. Tools and techniques for color image retrieval. In *IS & T/SPIE proceedings*, volume 2670, 1994. Storage & Retrieval for Image and Video Databases IV.

[39] John R. Smith and Shih-Fu Chang. Transform features for texture classification and discrimination in large image databases. In *Proc. IEEE Int. Conf. on Image Proc.*, 1994.

[40] John R. Smith and Shih-Fu Chang. Single color extraction and image query. In *Proc. IEEE Int. Conf. on Image Proc.*, 1995.

[41] John R. Smith and Shih-Fu Chang. Automated binary texture feature sets for image retrieval. In *Proc ICASSP-96*, Atlanta, GA, 1996.

[42] Markus Stricker and Markus Orengo. Similarity of color images. In *Proc. SPIE Conf. on Vis. Commun. and Image Proc.*, 1995.

[43] Michael Swain and Dana Ballard. Color indexing. *International Journal of Computer Vision*, 7(1), 1991.

[44] Hideyuki Tamura et al. Texture features corresponding to visual perception. *IEEE Trans. Systems, Man, and Cybernetics*, SMC-8(6), June 1978.

[45] Hideyuki Tamura, Shunji Mori, and Takashi Yamawaki. Texture features corresponding to visual perception. *IEEE Trans. on Sys, Man, and Cyb*, SMC-8(6), 1978.

[46] K. S. Thyagarajan, Tom Nguyen, and Charles Persons. A maximum likelihood approach to texture classification using wavelet transform. In *Proc. IEEE Int. Conf. on Image Proc.*, 1994.

[47] C. J. Van Rijsbergen. *Information Retrieval.* London: Butterworths, 1979.

[48] D. White and R. Jain. Similarity indexing with the ss-tree. *Proc. of ICDE*, 1995.