Multi-Feature Query by Multiple Examples in Image Databases

Surya Nepal *
Department of Computer Science,
RMIT University, GPOBox 2476V,
Melbourne VIC 3001, Australia
nepal@cs.rmit.edu.au

M.V. Ramakrishna
School of Computer Science and Software
Engineering, Monash University
Caulfield South 3145
rama@csse.monash.edu.au

ABSTRACT

Query by example(QBE) image are supported in Content-Based Image Retrieval (CBIR) systems, such as QBIC and Virage. We are addressing the problem of processing user queries of the form "retrieve images similar to I_1, I_2, ..., I_n based on color and texture". We refer to such a query as MF-QBME (Multi-Feature Query By Multiple Examples). Such user queries are supported in the CHITRA prototype CBIR system we are implementing. Users can define high level concepts such as SUNSET and MOUNTAINS which can later be used in queries such as "retrieve all images that have SUNSET and MOUNTAINS". Answering such queries also requires processing of MF-QBME. We have defined and introduced the concepts of *inclusion property* and *clustered query*. Processing MF-QBME queries involves dealing with multiple points in multiple feature spaces. We define the exact low level semantics MF-QBME, and provide the corresponding processing strategies. The experimental performance results are also presented.

Keywords

CBIR queries, query processing, query optimization, multimedia query processing

1. INTRODUCTION

Advances in storage and processing technologies have made multimedia databases (comprising large collections of images, video and sound) possible. Images are widely used in multimedia collections such as in medical imaging, geographic information systems, and photo galleries. Content Based Image Retrieval (CBIR) systems are required to utilize such collections. Many CBIR systems such as QBIC, Virage, VisualSEEk, NETRA and MARS have been developed recently. In this paper, we are addressing the issues of MF-QBME query processing in CBIR systems. The ideas developed are being implemented in the CHITRA prototype CBIR system we are developing (CHITRA means picture in "Sanskrit").

Unlike exact match in traditional databases, query processing in CBIR systems involves similarity search. The problem is to identify a set of database vectors which are "closest" to the given query feature vectors. The result of a query in traditional databases is always complete and satisfies the user's requirements. However, the CBIR query result is not fixed and depends on the user perception. Similar to traditional databases, efficient query processing in CBIR systems is essential to handle very large image databases.

A common method of querying in CBIR system is "query by example image". The system provides a set of images and features; the user picks relevant images and features, and requests other images with similar features. The query is constructed using corresponding feature vectors. For example, if a user wants to retrieve images of "sunset" from a database, he/she can compose a query by selecting an example image of a sunset. The user then expects to retrieve images of a sunset. Such querying mechanism is supported in CBIR systems such as QBIC, Virage, NETRA, and VisualSEEk. It is logical for users to expect better results by giving more example images of the same concept (sunset). Hence, we decided to support Query By Multiple Example (QBME) images in CHITRA. CBIR systems such as Virage and QBIC do not support such queries.

In this paper, we are addressing the problem of processing MF-QBME queries of the form *"retrieve images similar to I_1, I_2, .., I_n based on color and texture"*. All or none of the example images in the query may be present in the database. Even if all the example images are present in the database, it is quite possible that all the example images are not present in the retrieved set of images by the query processing algorithm. To capture these, we have defined the concept of *inclusion property*. The retrieved result is also dependent on the relative placement of the query images in the feature spaces. We have defined the concept of *clustered query* to describe such a property of the query.

The MF-QBME are implicitly encountered in processing

*The author is now at CSIRO Mathematical and Information Sciences, Locked Bag 17, North Ryde NSW 1670, Australia. Email: Surya.Nepal@cmis.csiro.au

ADVANCES IN DATA MANAGEMENT 2000
Krithi Ramamritham, T.M. Vijayaraman (Editors)
Tata McGraw-Hill Publishing Company Ltd.

concept queries and relevance feedback, as described below.

User Queries: The graphical user interface in CHITRA supports the user to select more than one example image and feature while posing queries [5, 8]. The user can also give images that are not in the database as examples to the system, whereupon the relevant features are extracted on line. Other ways of composing MF-QBME in CHITRA are by selecting inter-image features, inter-image object features, or intra-image object features.

Concept Queries: CHITRA supports queries based on concepts such as "sunset" and "mountains". The user can define concepts using multiple example images and features. Such concepts are stored in the database and the user can later apply them to compose complex concept queries. The user can compose concept queries such as *"retrieve images of sunset"* and *"retrieve images of sunset and mountains"* [10]. Processing such queries requires handling of MF-QBME at the system level.

Relevance Feedback: Text retrieval systems use relevance feedback where users can select the relevant documents and refine the original query. Experimental evidence suggests that relevance feedback improves the query results. Relevance feedback techniques are also used in CBIR systems to refine the user's queries, and is supported in CHITRA [7]. The user can pose a query to the system and retrieve the similar images. If the user is not satisfied with the retrieved images, he/she can select the images from the retrieved set that are meaningful and give to the system as feedback. CBIR systems such as QBIC and Virage also support relevance feedback with a single image. MARS supports relevance feedback with multiple example images [15]. Handling relevance feedback requires processing of MF-QBME at the system level.

The rest of the paper is organized as follows. Various CBIR query processing strategies are reviewed in Section 2. In Section 3 we introduce and define the *inclusion property* and *clustered query* and discuss the processing of MF-QBME queries. Experimental results are reported in Section 4 followed by conclusions.

2. BACKGROUND

The query processing literature of our interest can be summarized in terms of the four queries shown in Figure 1.

2.1 Processing of Single Example Queries

We refer to a query containing a single feature vector such as Q_1 as a *simple query*. Processing such queries involves retrieving the k nearest images in the feature space, where nearness is defined by the similarity measure using search algorithms [16].

We call queries such as Q_2 involving more than one feature as complex queries. The processing Q_2 involves retrieval from indexes on color and texture features, and combining them appropriately. Such conjunctions of atomic queries forms an important class of CBIR queries. Fagin defined the semantics of such queries using fuzzy logic and has proposed an algorithm to combine the results of atomic queries [3]. Fagin's algorithm has been implemented in the IBM's Garlic project to combine multiple systems within the same framework [2]. Nepal and Ramakrishna have carried out an analysis of Fagin's algorithm, and proposed a multi-step algorithm [9]. The multi-step algorithm performs better than Fagin's algorithm for fuzzy combining functions which satisfy certain bounding properties. Ortega et al. proposed a complex query processing algorithm based on a demand driven approach [12]. Their algorithm retrieves one element at a time for the atomic queries instead of retrieving k elements as in Fagin's approach. Thus, the number of elements that are retrieved for each atomic query depends on the combining functions used.

2.2 Processing of Single Feature QBME (Q_3)

Single Feature QBME are supported in CBIR systems such as MARS [15] and MindReader [4]. Porkaew et al. present two approaches of processing such queries [15]: Centroid Expansion Search (CES) and Multiple Expansion Search MES). In MES, the k nearest neighbors for n query points are determined by iteratively retrieving next nearest neighbors for each query point. Thus, in order to obtain the top k images, MES incrementally evaluates the nearest neighbor for each of the example images until there are at least k images in the result. Efficient strategies of implementing MES overcoming the problem of making n k-nearest neighbor calls are also proposed in [15]. The key idea is to traverse a multidimensional index structure such that the best k objects are retrieved from the data collection without having to explicitly execute k nearest neighbor queries for each point in the query.

Query point movement attempts to move the query representation in the direction where relevant objects are located [13]. At any instance, the query point is represented by a single point in each feature space. When the user uses multiple examples to construct the query, the centroid is used as the single point query. When the user interacts with the system, he/she marks the relevant images and poses the relevance feedback query. Then, the weighted centroid of the relevant images is used as the single query point. The user can also provide different levels of relevance. The weights are obtained from the relevance levels provided by the user.

Ishikawa et al. proposed a method for handling multiple example queries in a single feature space [4]. This method tries to find an appropriate distance function and a query point using the examples returned as feedback by the user.

3. MULTI-FEATURE QBME

We first introduce some functions and properties for our discussion on the semantics and processing of MF-QBME. We assume that the system has all the relevant features extracted and stored in appropriate index structures. The system supports k-nearest neighbor search, range search, and the following function calls.

GetNextK(f_v, k) : The subsystem returns the most similar k elements based on the similarity of those elements with query feature vector f_v. When an algorithm makes a $GetNextK()$ call for the first time, the

Q_1 (single feature single example) : *''Retrieve images similar to I_1 based on color''.*
Q_2 (multi-feature single example) : *''Retrieve images similar to I_1 based on color and texture''.*
Q_3 (single feature QBME) : *''Retrieve images similar to $I_1, .., I_n$ based on color''.*
Q_4 (multi-feature QBME) : *''Retrieve images similar to $I_1, .., I_n$ based on color and texture''.*

Figure 1: Four types of queries

subsystem returns the top k elements based on the similarity. The subsystem then starts to yield elements from $k+1$ most similar images in the subsequent calls.

GetSim(I, f_v) : The subsystem returns the similarity of image I with query feature vector f_v. As against the above two functions, which requires the use of the feature index, $GetSim()$ only needs to use direct access.

3.1 Some useful properties

Some or all of the example images in the MF-QBME may not be present in the database. It is also quite possible that, even if all the example images are in the database, the result may not include some or all the example images. If the user does not see the example images in the query results even though they are in the database, the system looks unfriendly and the user will be annoyed. We define some properties to capture these. Let Q_s be the set of example images $Q_s = \{I_1, .., I_n\}$, and DB the set of all images in the database. The query Q_4 becomes,
Q_4: retrieve k images similar to Q_s based on color and texture.

DEFINITION 1. **inclusive query:** *A query such as Q_4 is said to be inclusive if all the example images are in the database.*

$$Q_s \subset DB$$

DEFINITION 2. **inclusive algorithm:** *If for every inclusive query Q, and $k \geq n$, the result returned by an algorithm contains Q_s then the query processing algorithm is said to be inclusive.*

$$\textit{For every query } Q \textit{ whose } Q_s \subset DB, \quad NN_Q^k \supset Q_s$$

where NN_Q^k is the result returned by the algorithm (set of k most similar images).

For the systems to be user-friendly and provide greater user satisfaction, the query processing algorithms should be inclusive. The performance of query processing algorithms depends on the distribution of query points in the corresponding feature spaces. For example, if the user wants to retrieve snow-covered mountain images, he/she specifies one or more such mountain as examples. For some concepts, the feature vectors of all the images may be close to each other. In other concepts, they may be on a straight line for example as illustrated in Figure 2(b) for hypothetical two dimensional feature vectors. The choice of an algorithm is dependent on these properties. Some algorithms in the literature [15, 14] appear to assume that the distribution will be similar to that in Figure 2(a). We define k-clustered query to capture this property of the feature spaces.

DEFINITION 3. **k-clustered query:** *A query such as Q_4 is said to be k-clustered query if Q_s is contained in the $(n+k)$ nearest neighbor in the database of every example image I_r.*

$$\forall I_r \in Q_s : NN_{I_r}^{n+k} \supset Q_s$$

where $NN_{I_r}^{n+k}$ is a set of $(n+k)$ most similar images with image I_r.

3.2 Semantics and Processing Algorithms

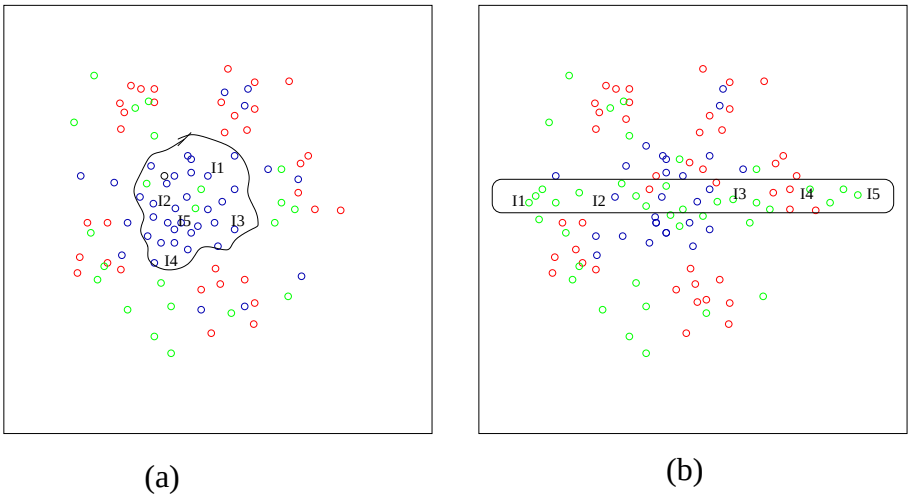
The query Q_4 is illustrated for a (hypothetical) two dimensional feature space in Figure 3. There are five query points, one corresponding to each example image, in each of the two feature spaces. The closeness of the features of example images depends on the nature of the feature values. We define two possible semantics of MF-QBME queries and propose the corresponding processing strategies.

DEFINITION 4. *A* **Multi-Feature Multiple Example queries** *$MFQBME =< m, n, Q_s, F, d >$ consists of images $Q_s = \{I_1, I_2, .., I_n\}$, features $F = \{f_1, .., f_i, .., f_m\}$, and a distance function d that given two images, returns the distance between them based on feature f_i. We say $m \times n$ is the size of the $MFQBME$.*

The distance function d determines the distance between two images on the feature space defined by f_i. Let $I_{i_{f_j}}$ be a feature vector for the example image i in the feature space f_j. The distance between a database image I_{db} in the feature space from the given example image is given by $d(I_{i_{f_j}}, I_{db_f})$. The distance of a database image I_{db} from the Q_s is then given by,

$$D(I_{db}, Q_s) = G(d_{11}, d_{12}, ..., d_{mn})$$

where $d_{11}, .., d_{mn}$ are the distances of the database image I_{db} with features $I_{1_{f_1}}, .., I_{n_{f_m}}$, respectively. The semantics is based on the combining function G. There are many possible ways of defining the semantics for the MF-QBME. We use the fuzzy framework to evaluate the query. We have identified two ways of defining semantics for the MF-QBME.



(a) (b)

Figure 2: An illustration of clustering property in a hypothetical two dimensional space

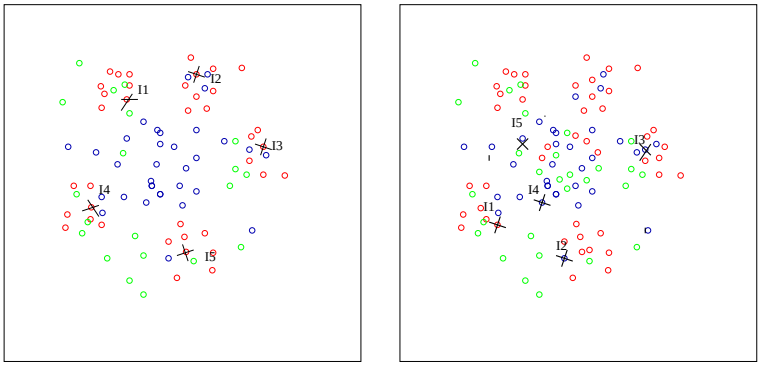


Figure 3: An illustration of multi-feature multiple example query in a hypothetical two dimensional space

Feature Priority (OR-AND) Approach

In this approach, the given query is decomposed into single feature multiple example queries. Each is then processed using the appropriate above mentioned algorithms. The results are combined using an algorithm such as the multi-step algorithm. The semantic tree for the query is shown in Figure 5. This approach first performs the OR operation on the feature and then AND operation, hence its name. This query tree can be processed using the data driven algorithm such as that proposed in [12]. In describing the algorithm, we use standard fuzzy functions Min and Max to evaluate AND and OR operators, for simplicity. However, the algorithm remains the same for any other combining function [3]. The AND and OR operators in our framework are evaluated as follows.

- AND - maximum of all distances or minimum of all similarity values.
- OR - minimum of all distances or maximum of all similarity values.

Semantics

The low level semantics of the query *retrieve k images most similar to I_1,..,I_n based on features f_1,..,f_m,* is interpreted as follows.

For each database image I compute the final grade $G(I)$ as,

$G(I) = (d_{11} \vee .. \vee d_{n1}) \wedge .. \wedge (d_{1m} \vee .. \vee d_{nm})$

where d_{ij} is the distance of database image I with example image I_i based on feature f_j. The symbols $\wedge$ and $\vee$ are AND and OR operators, respectively. The required result is then the sorted list of images based on $G(I)$.

Processing:

The query semantic tree for Q_4 based on the feature priority approach is shown in Figure 5. $ColSim()$ and $TexSim()$ return lists of database images based on the similarity values with color and texture features, respectively. The AND and OR operators are used to combine the results. The query semantic tree is then processed using the algorithm shown in Figure 4.

1. The leaf nodes are processed using a k-nearest neighbor search algorithm to retrieve most similar k images using the $GetnextK()$ function.

2. The OR and AND nodes are processed as follows.

   - For each node i, request the node i to return the next image x. Thus, each node i outputs the graded set of pairs $(x, \mu_i(x))$, where x is an image in the database and $\mu_i(x)$ is the similarity value of x under i.
   - For each image x returned by the node i in the current iteration, do random access to j using the $GetSim()$ function, $i \neq j$, to find $\mu_j(x)$.
   - Compute the threshold grade, t_h for this iteration, $t_h = t(\mu_{q_i}(x_i), .., \mu_{q_m}(x_m))$. The function t is Min for AND node and Max for OR node. Here x_i is the element retrieved by node i in the current iteration.
   - Compute the grade $\mu_Q(x) = t(\mu_{q_i}(x), .., \mu_{q_m}(x))$ for each image x retrieved in this iteration. Update Y, the set containing all images that have grade $\mu_Q(x) \geq t_h$.
   - Repeat the above iteration until the set Y has k images.
   - Output the graded set $\{(x, \mu_Q(x) | x \in Y\}$.

Figure 4: An algorithm to process the query tree

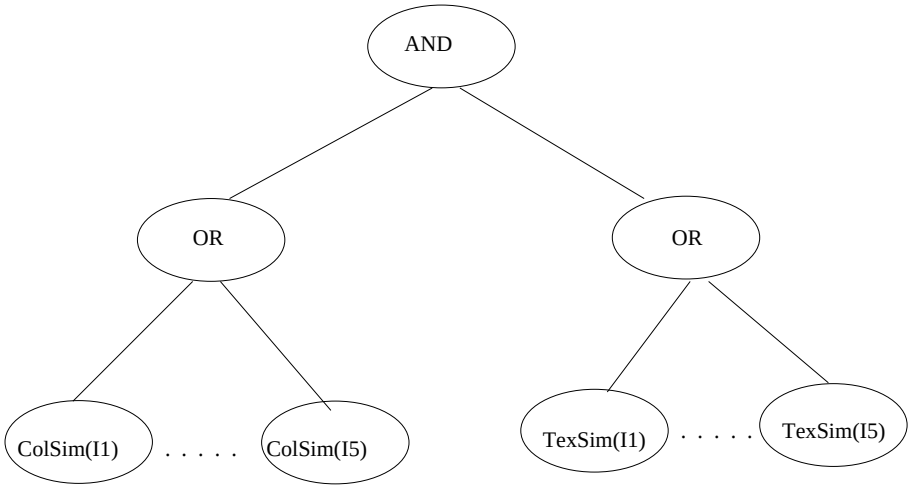


Figure 5: The query tree for Q_4 based on feature priority (OR-AND) approach

Image Priority (AND-OR) Approach

In this approach, the given query is decomposed into multi-feature single example sub-queries. Each sub-query is then processed using algorithms mentioned in the last section. The results are then combined using the combining functions (such as, Max for fuzzy OR). Figure 6 shows the tree representation of the query semantics. This algorithm first performs the AND operation and then OR. Here again, the algorithm is given in terms of fuzzy evaluation functions Min and Max for AND and OR operators. But the algorithm is valid for any other functions.

Semantics

The low level semantics of the query *retrieve k images most similar to* $I_1,..,I_n$ *based on features* $f_1,..,f_m$, is interpreted as follows.

For each database image I compute the final grade $G(I)$ as,

$G(I) = (d_{11} \wedge .. \wedge d_{1m}) \vee .. \vee (d_{n1} \wedge .. \wedge d_{nm})$,

where d_{ij} is the similarity of I with example image I_i based on feature f_j. The symbols $\wedge$ and $\vee$ represent AND and OR operations, respectively. The required result is then a sorted list of images based on $G(I)$.

Processing

The query semantic tree for Q_4 based on the image priority approach is shown in Figure 5. $ColSim()$ and $TexSim()$ return lists of database images based on color and texture similarity, respectively. The AND and OR operators are used to combine the results. The query semantic tree is then processed using the algorithm shown in Figure 4.

4. EXPERIMENTAL EVALUATION

The execution cost of the algorithms should be measured in terms of number of disk page accesses, memory uses, CPU processing costs etc. We use the number of objects accessed to measure the execution cost E_c, which serves our purpose of comparing algorithms [3]. This cost model is **not realistic** since the number of disk accesses required for sorted access are in general higher than that for direct access. Our aim here is to compare the relative performance of proposed query processing algorithms, independent of underlying index structures. This also avoids the effects of a particular index structure being used on the execution cost.

The *sorted access cost* is the number of elements retrieved by the subsystems in response to $GetNextK()$ call. For example, if each of the subsystems retrieves 50 images in response to each of the five $GetNextK()$ calls, then the sorted access cost of the algorithm is 250. Each call to $GetSim()$ to find the similarity value of an image incurs a unit *direct access cost*. The *execution cost* is the total number of objects accessed from the system.

The retrieval performance of MF-QBME query processing algorithms is measured in terms of precision/recall values calculated as follows.

Precision is the ratio of the number of relevant images retrieved to the total number of images retrieved. The ideal situation corresponds to 100% precision, when all retrieved images are relevant.

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

Recall is the ratio of the number of relevant images retrieved to the total number of relevant images. We can achieve ideal recall (100%) by retrieving all the images from the database, but the corresponding precision will be poor.

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

Some features give better retrieval performance for a given collection than others. As our aim is not to achieve the best precision/recall values, but to measure the relative performance of the algorithms. The experimental results should be **judged in terms of relative values** rather than the absolute.

4.1 Experimental Details

There is a lack of a standard benchmark for testing CBIR query processing algorithms. In accordance with the consensus reached at the SIGIR'98 workshop, we used the Corel collection of images [17]. A collection of 1000 images was chosen from two different volumes of Corel photo CDs: *travel destinations* and *natural scenes*. We used the following high level concepts for our query images: *buildings, fishes, flowers, flowerbeds, greenbeds, mountains, people, plants, sea, and sunset*. Each image was prejudged to be relevant or not for each of the query (so as the experimental results are unbiased). This process led to the partitioning of the image collection into the query concepts. We considered two global features: color and texture. Following Carson and Ogle's experimental results on human perception of colors, we used 13 dimensional color feature vectors [1]. For texture we used 16 dimensional Gabor texture feature vectors generated using four different scales and two orientations [6]. The feature vectors for all the images were extracted and stored appropriately.

To evaluate the retrieval performance of MF-QBME queries, we posed queries of the form Q_4. For each concept, we posed a query using a randomly selected set of five example images. In each case, the recall/precision values were computed based on the system response and the relevance judgment, as defined above. To measure the relative retrieval performance, we plot precision values at various intervals of recall for the two different algorithms. The retrieval performance of these algorithms for one set of queries (with sunset example images) is shown in Figure 7. We observe that the image priority (AND-OR) approach gives better retrieval performance for this type of query.

4.2 Evaluation

Tables 1 shows the recall/precision results of the experiments with query Q_4, $k = 20$. The execution costs shown are the observed number of images accessed in each case. They also show the effective cost P_c, computed using the CBIR cost model [11]. We use E_{c_i} to denote the execution cost incurred by an algorithm for retrieving i images. Let the corresponding retrieval performance be R_{p_i}. The evalu-

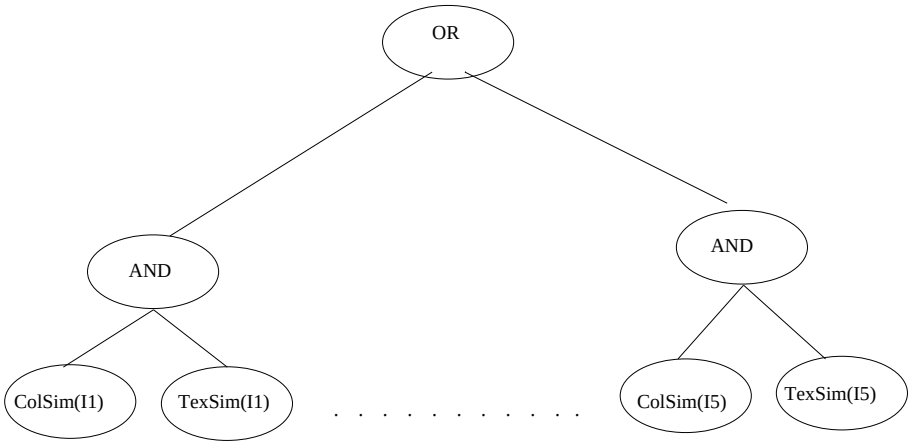


Figure 6: The query tree for Q_4 based on image priority (AND-OR) approach

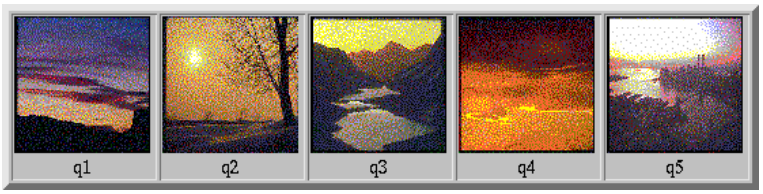


Figure 8: An example query Q_4: "retrieve images similar to $q_1, .., q_5$ based on color and texture".

Figure 9: The best 20 images retrieved by the OR-AND algorithm in response to the query Q_4

Figure 10: The best 20 images returned by the AND-OR algorithm in response to the query Q_4

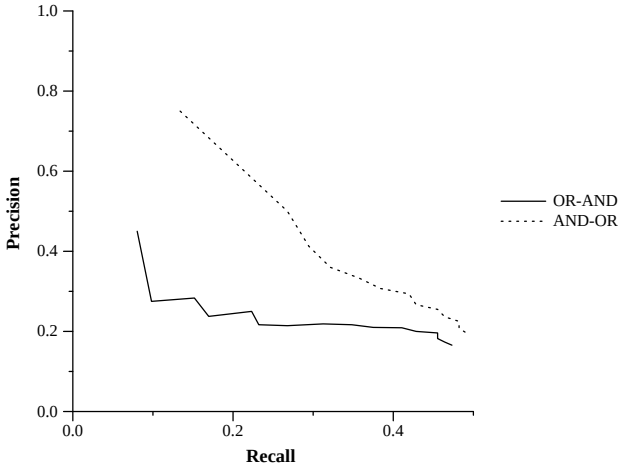


Figure 7: Performance of $MF-QBME$ processing algorithms for sunset examples

ation measure P_{c_i} we have proposed is a combination of E_{c_i} and R_{p_i} with the appropriate combining function given by,

$$P_{c_i} = E_{c_i} \odot R_{p_i} \tag{1}$$

We use multiplication $\times$ with suitable scale factor K as the combining function.

$$P_c = E_c \times (1 + K \times R_p) \tag{2}$$

The retrieval performance R_p is defined as a negative double-exponential function of precision (p).

$$R_p = \begin{cases} (1 - R_{max} \times \exp^{(-R_{min} \times \exp^{(-R_{ch} \times p)})}) \\ \text{if } 0 \geq (R_{max} \times \exp^{(-R_{min} \times \exp^{(-R_{ch} \times p)})}) \leq 1 \\ 0 \\ \text{if } (R_{max} \times \exp^{(-R_{min} \times \exp^{(-R_{ch} \times p)})}) > 1 \end{cases}$$

where p is the precision value, R_{max}, R_{min} and R_{ch} are the parameters that can be chosen by the users (and hence it is a parametric approach).

We explain the results shown in the last row of the table. To retrieve the 20 best images from the database of color and texture features, the *OR-AND* algorithm gives a precision value of 0.45 (as shown in Figure 7) and incurs an execution cost of 436 accesses. The corresponding retrieval performance R_p is calculated based on the CBIR model for $p = 0.45$ as 0.11 [11]. The effective cost P_c for $K = 1$ is then given as,
$P_c = E_c \times (1 + K \times R_p)$
$P_c = 436 \times (1 + 1 \times 0.11)$
$P_c = 483.$

Queries			*ExecutionCost (E_c)*		*Effective Cost (P_c)*			
	OR-AND	AND-OR	OR-AND	AND-OR	OR-AND		AND-OR	
	p/R_p	p/R_p			$K=1$	$K=100$	$K=1$	$K=100$
Buildings	0.45/0.11	0.40/0.17	346	1358	**384**	**4152**	1588	24444
Fish	0.25/0.56	0.25/0.56	418	1036	**652**	**23826**	1616	59052
Flowers	0.30/0.39	0.40/0.17	399	1088	**554**	**15960**	1272	19584
Flowerbeds	0.40/0.17	0.35/0.26	619	1444	**724**	**11142**	1819	38988
Greenbeds	0.35/0.26	0.35/0.26	420	1366	**529**	**11340**	1721	36882
Mountains	0.30/0.39	0.30/0.39	550	1272	**764**	**22000**	1768	50880
People	0.25/0.56	0.30/0.39	241	1479	**375**	**13737**	2055	59160
Plant	0.60/0.03	0.60/0.03	247	760	**254**	**988**	782	3040
Sea	0.25/0.56	0.25/0.56	485	1539	**756**	**27645**	2400	87723
Sunset	0.45/0.11	0.75/0.00	436	1166	**483**	5232	1166	**1166**

Table 1: Effective cost of $QBME_{mf}$ based on color and texture .

The corresponding values for the *AND-OR* algorithm are, $R_p = 0.00$ and $P_c = 1166$. Thus, we conclude that *OR-AND* is the better algorithm. If we change the value of K to 100, the effective P_c for *OR-AND* becomes,
$P_c = 436 \times (1 + 100 \times 0.11)$
$P_c = 5232$

which is worse than the *AND-OR* cost of 1166. Note that the values of P_c for the AND-OR algorithm remain the same, independent of K. This is due to the high value of precision (0.75), which makes $R_p = 0$. The user is fully satisfied with the precision, and there is no penalty of lower precision to the execution cost. The higher value of K penalizes the poor retrieval performance more heavily. The query ("sunset") and the results obtained for both AND-OR and OR-AND algorithms are shown in Figures 8,9, and 10.

5. CONCLUSIONS

Much of recent research has focused on processing simple CBIR queries using high dimensional index structures. In this paper, we described the motivation for the support of MF-QBME queries in CHITRA. We defined the two possible semantics of such queries using fuzzy logic, and provided the corresponding processing algorithms. Based on the experimental results, we draw the following conclusions.

- The choice of an algorithm depends on the distribution of data points and query points in a particular feature space. The characterisation of the feature space in relation to query points is left as an open problem.
- In general, the AND-OR algorithm gives a better retrieval performance than OR-AND for low recall values. For higher recall values, the difference in retrieval performance is not very significant.
- The execution cost of the OR-AND approach to process MF-QBME queries is much lower than that of AND-OR approach.
- Based on the overall performance, we observed that it is always better to use the OR-AND approach than the AND-OR approach for processing MF-QBME queries.
- Both AND-OR and OR-AND algorithms are inclusive.

6. REFERENCES

[1] Chad Carson and Virginia E. Ogle. Storage and retrieval of feature data for a very large online image collection. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 19(4):19–27, December 1996.

[2] W. F. Cody, L. M. Haas, W. Niblack, M. Arya, M. J. Carey, R. Fagin, M. Flickner, D. Lee, D. Petkovic, P. M. Schwarz, J. Thomas, M. Tork Roth, and J. H. Williams and. Querying multimedia data from multiple repositories by content: the Garlic project. Third Working Conference on Visual Database Systems (VDB-3), pages 17 – 35, Lausanne, Switzerland, March 1995.

[3] Ronald Fagin. Combining fuzzy information from multiple systems. Proc. Fifteenth ACM Symp. on Principles of Database Systems, pages 216–226, Montreal, 1996.

[4] Yoshiharu Ishikawa, Ravishankar Subramanya, and Christos Faloutsos. MindReader: Querying databases through multiple examples. In Ashish Gupta, Oded Shmueli, and Jennifer Widom, editors, *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 218–227. Morgan Kaufmann, 1998.

[5] M. Kerr and M.V. Ramakrishna. Design of a graphical query mechanism for the CHITRA CBIR system. Technical report, CS Department, RMIT University, 1999.

[6] W. Y. Ma and B. S. Manjunath. Texture features and learning similarity. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 425 – 430, 1996.

[7] Me.E.J.Wood, N.W.Campbell, and B.T.Thomas. Interactive refinement by relevance feedback in content-based digital image retrieval. In *Proceedings of ACM Multimedia 98*, pages 13–20, Bristol, England, September 12-16, 1998.

[8] Surya Nepal and M.V.Ramakrishna. A generalized test bed for image databases. In *10th International Conference of the Information Resources Management Association*, pages 926–928, Hershey, Pennsylvania, USA, May 1999.

[9] Surya Nepal and M.V.Ramakrishna. Query processing issues in image(multimedia) databases. In *Fifteenth International Conference on Data Engineering (ICDE)*, pages 22–29, March 23-26, Sydney, Australia, 1999.

[10] Surya Nepal, M.V.Ramakrishna, and J.A.Thom. Four layer schema for image data modelling. In Chris McDonald, editor, *Australian Computer Science Communications, Vol 20, No 2, Proceedings of the 9th Australasian Database Conference, ADC'98*, pages 189–200, 2-3 February, Perth, Australia, 1998.

[11] Surya Nepal and M.V. Ramakrishna. An evaluation measure for query processing in CBIR systems. In *11th International Conference of the Information Resources Management Association*, Anchorage, Alaska, USA, May, 21-24 2000. To appear.

[12] M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrota, and T. Huang. Supporting ranked boolean similarity queries in MARS. *IEEE Transaction on Knowledge and Data Engineering*, 10(6):905–925, 1998.

[13] Michael Ortega, Yong Rui, Kaushik Chakrabarti, Sharad Mehrotra, and Thomas S. Huang. Supporting similarity queries in MARS. In *Proc. of ACM Conf.on Multimedia*, pages 403 – 413, 1997.

[14] Kriengkrai Porkaew, Kaushik Chakrabarti, and Sharad Mehrotra. Query refinement for content based multimedia retrieval in MARS. In *IEEE Int. Conference on Multimedia Computing and Systems (ICMCS)*, volume II, pages 25–36, Centro Aftari, Florence, Italy, 1999.

[15] Kriengkrai Porkaew, Sharad Mehrotra, Michael Ortega, and Kaushik Chakrabarti. Similarity search using multiple examples in MARS. In *International Conference on Visual Information Systems, VISUAL'99*, pages 68–75, Amsterdam, the Netherlands, 1999.

[16] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *Proceedings of ACM - SIGMOD Intl. Conference on Management of Data*, pages 71–79, June 1995.

[17] Rohini K. Srihari, Zhongfei Zhang, R. Manmatha, and S. Ravela. Multimedia indexing and retrieval. In *21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR98)*, Melbourne, Australia, August 24 - 28, 1998. Workshop.