Query Processing in Multimedia Databases

Ramakrishna, M.V.
Monash University, Melbourne
Presented at ACM CIKM 2000

Contributors:
Dr. S. Nepal.

Overview

- Introduction
  Nature of multimedia queries
  Comparison to traditional queries
  Fuzzy Nature of MMQP
- Query Languages
  Elementary CBIR Queries
  Query by Example
- Types of MMQ
- Query By Example(QBME) Images

Introduction

Multimedia data has become prevalant due to the advent of the computing technology: processor speed, memory capacity and communication at low cost, ubiquitous availability (every where).

Fast handling/retrieval of such data has become a necessity.

Similar to traditional data(free text and record structured) of yester years, multimedia data needs to be handled.

Tools/techniques are necessary.

Nature of Traditional Data

The traditional databases deal with exact data.

Given a set of employee records, we need to organise and answer queries such as,

- retrieve all employees with last name = 'smith';
- retrieve all employees whose salary $>$ 50000 and $<$ 60000;

There is no ambiguity,

- in the data,
- in the query,
- and in what is to be retrieved and presented to the user.

Nature of Multimedia Data

Multimedia data is most often 'fuzzy'.

Given a set of images (image database), we need to `retrieve images similar to` I_1 `based on colour'';`
`retrieve images with 'Bill Clinton' in it;`

Given a set of share prices, need to "retrieve share price of `pronounced`('qantas');"
(as supported by stock broker *commonwealth securities* telephone stock quote facility)
Here the data is exact, but the *actual query* is fuzzy.

Fuzzy Queries and Results

Query languages which enable fuzzy specification of user requirement have been developed - beyond the scope of this tutorial.

The result required is usually a set of objects, sorted on the similarity with the query object.

How good is a result returned by the system?

The "quality" of result is subjective: dependent on how satisfied is a user.

User Satisfaction

The quality of the result is measured in terms of *recall* and *precision*.

$$Precision = \frac{number of relevant objects in the result}{total number of objects in the result}.$$

$$Recall = \frac{number of relevant objects retrieved}{total number of relevant objects in the database}.$$

Evaluation of Query Processing Algorithms

How to evaluate query processing algorithms, similarity measures, features etc?

Traditional databases: I/O, CPU and memory costs.

Multimedia: In addition, user satisfaction with the system output need be considered.

The question is, how to combine the two (Execution Cost and Retrieval Performance) to arrive at a useful measure?

The Evaluation Measure

E_c: Execution Cost
(disk accesses, CPU time, ...).

R_p: Retrieval Performance
$R_p = f(p)$.

The Effective Cost P_c is defined as:

$$\boldsymbol{P_c = E_c \odot R_p}$$

where $\odot$ is the combining function.

Non-Linear(CBIR) Model:

To better reflect the reality of CBIR applications, we define,

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

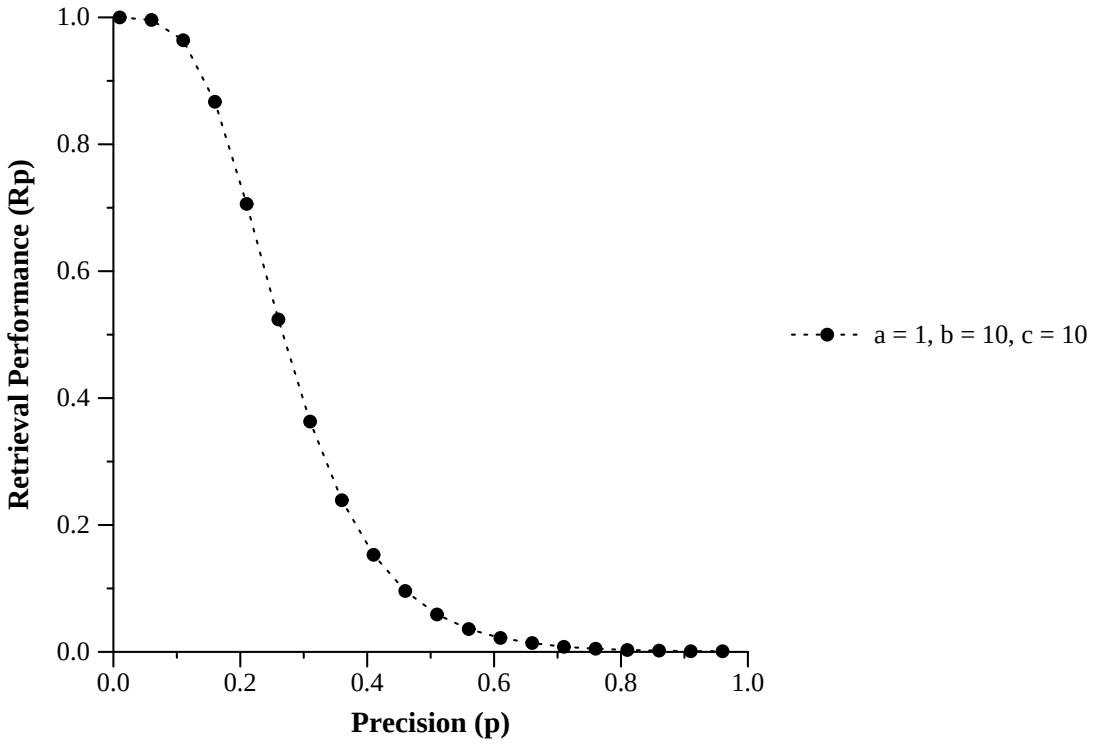
where K is the scale factor, and R_p is a double exponential.

$$\boldsymbol{R_p = (1 - R_{max} e^{(-R_{min} e^{(-R_{ch} p)})})}$$

where R_{max}, R_{min} and R_{ch} are user-defined parameters.
R_p is constrained to be between 0 and 1.0.

This cost measure enables a user to avoid penalizing an algorithm giving a precision of 0.9 as against an algorithm giving perfect precision. At the same time, algorithms giving too low precisions are heavily penalized.

Plot of Retrieval Performance



R_{max}: determines R_p when precision is 1.0.

R_{min}: determines the maximum penalty (when $R_p = 0$).

R_{ch}: determines the range of p when the penalty changes rapidly.

An Example

Alg	Disk Acc	Prec	P_c *(Linear Mod)*		P_c *(CBIR Mod)*	
			$R_p = (1-p)$		$R_p = 1 - ae^{-be^c}$	
			$W_r = 6$	$W_r = 100$	K $= 1$	K $= 10$
A_1	100	1.0	100	**300**	100	100
A_2	50	0.6	120	2000	**55**	100
A_3	80	0.9	**48**	8000	80	**80**

Types of Multimedia queries

Our discussion will be in terms of processing the following CBIR queries.

Q_1: "retrieve images similar to I_1 based on colour".
Q_2: "retrieve images similar to I_1 based on colour AND texture".
Q_3: "retrieve images similar to $I_1, I_2, .., I_k$ based on colour".
Q_4: "retrieve images similar to $I_1, I_2, .., I_k$ based on colour AND texture".

Processing Simple CBIR Queries

Q_1: "retrieve images similar to I_1 based on colour".

This would be processed as follows:

1. Extract color feature vector f_1 of I_1.
2. Pose a KNN (or region) query to the color index.
3. Display the images retrieved ordered on similarity with f_1.

Evaluation of combining functions

Combining functions are necessary when processing complex queries like:
Q_2: "retrieve images similar to I_1 based on colour AND texture".

How can we efficiently evaluate the combining function AND?

Fagin has proposed an algorithm for evalution of combining functions, based on fuzzy logic for multi-database systems [2, 1].

A simple algorithm to evaluate query Q_2 is as follows.

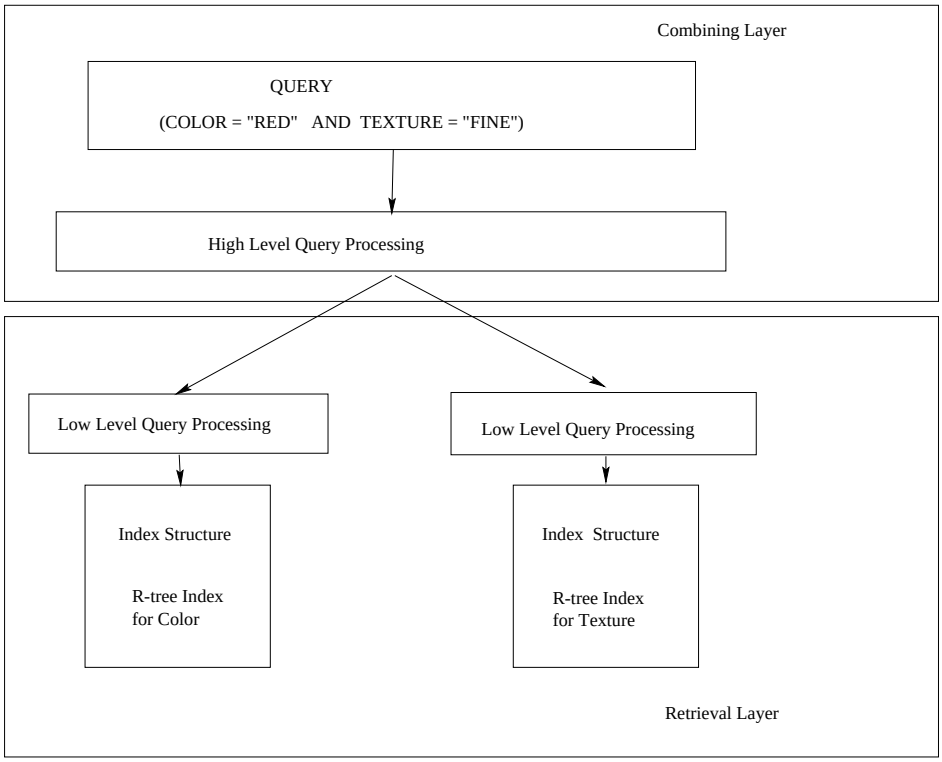
1. Obtain a sorted list of images based on colour similarity.

2. Obtain a sorted list of images based on texture similarity.

3. Obtain the final sorted list by taking the minimum of the similarity values from those two sorted lists.

4. Display the top k images on the screen. When the user requests "next k", the system displays a further k images from the sorted list.

Query processing system



Motivation for Our Algorithm

Consider the data set:

Colour = "red"		——	texture = "fine"	
objid	Grade		objid	Grade
01	0.9		04	0.5
02	0.8		03	0.45
03	0.7		05	0.4
04	0.5		02	0.3
05	0.1		01	0.2

The result of the query:

colour="red" AND texture = "fine"

is:

objid	Grade
04	0.5
03	0.45
02	0.3
01	0.2
05	0.1

The total number of accesses required using Fagin's approach for $(k = 2)$ is 10 (8 sorted accesses and 2 random accesses).

However, using multiple steps and exploiting the properties of the min function, we can reduce the total number of accesses to 8
(4 sorted accesses and 4 random accesses).

The Multi-step Algorithm

1. For each feature, say colour, in the query, request the subsystem col to return the next image x.

2. For each image x returned, do random access to other subsystems (find $\mu_{tex}(x)$).

3. Compute the threshold, $t_h = min(\mu_{col}(x_i), .., \mu_{tex}(x_m))$.

4. Compute the grade $\mu_Q(x) = t(\mu_{col}(x), .., \mu_{tex}(x))$ for each image. Update $Y = \{x \mid \mu_Q(x) \geq t_h\}$.

5. Repeat steps 2 to 5 until the set Y has k images.

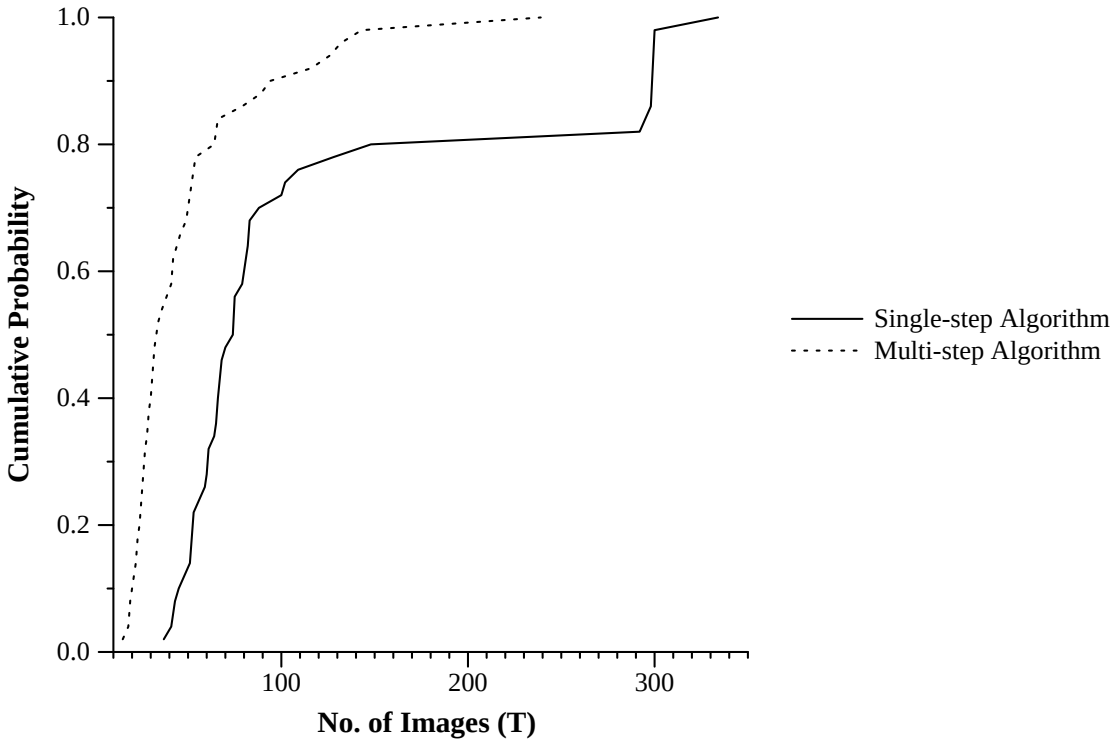
6. Output the graded set $Y = \{(x, \mu_Q(x)\}$.

Experimental Results

Colour and Texture Features

We performed an experiment with an image database, $N = 1000$, using Colour histogram and Gabor texture features.

Probability distribution of number of images accessed for $m = 2$ and $k = 10$:



With $N = 1000$, $k = 10$ and $m = 5$, we observed the following results:

Cost	Fagin's (Single-step) Algorithm	Multi-step Algorithm
Expected	288.35	191.57
Sorted	1441.77	957.88
Random	2058.63	1750.65
Total	3500.40	2708.53

Summary of the section

- A new multi-step algorithm for evaluating combining functions such as min.
- We are working on, other combining functions, and a more practical cost model considering index characteristics etc.

Queries By Multiple Example (QBME) Images

Q_3: "retrieve images similar to $I_1, I_2, ..., I_k$ based on colour"

Q_4: "retrieve images similar to $I_1, I_2, ..., I_k$ based on colour AND texture"

QBME is encountered when the user:

- explicitly poses such a query,
- provides relevance feedback,
- poses a concept query such as: "$Sunset$ AND $Mountain$"

Single Feature QBME

There are two broad approaches to process QBME.

- Effective Nearest Neighbour (ENN)
- K Nearest Neighbour (KNN)

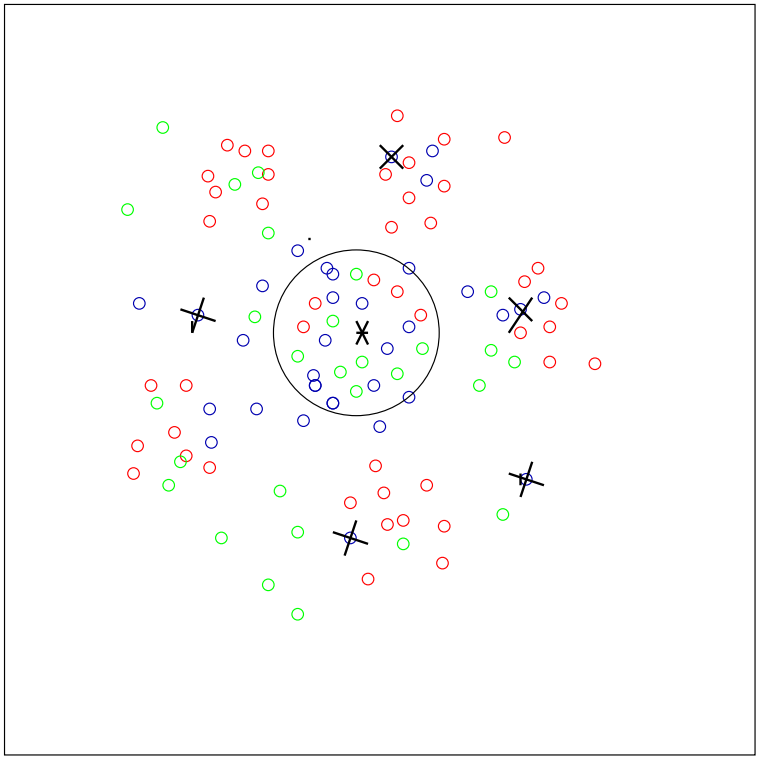
In the ENN approach, a single effective query point is first computed and then the similarity search is made.

In the KNN approach, K different nearest neighbour searches are made.

The ENN Approach

The effective query point (such as the centroid) is computed using one of several techniques.

The following diagram shows the basic idea of ENN approach.



Different "effective" queries

Query Point Movement (QPM):
This method is used in the MARS system developed by Mehrotra, Ortega et.al.

First, the effective query point is computed as the centroid of the given queries.

Then, using feedback from the user, the query point is moved towards good examples and away from the bad ones.

Query Reweighting:
The effective point is computed by iteratively giving different weights to the examples based on user feedback.

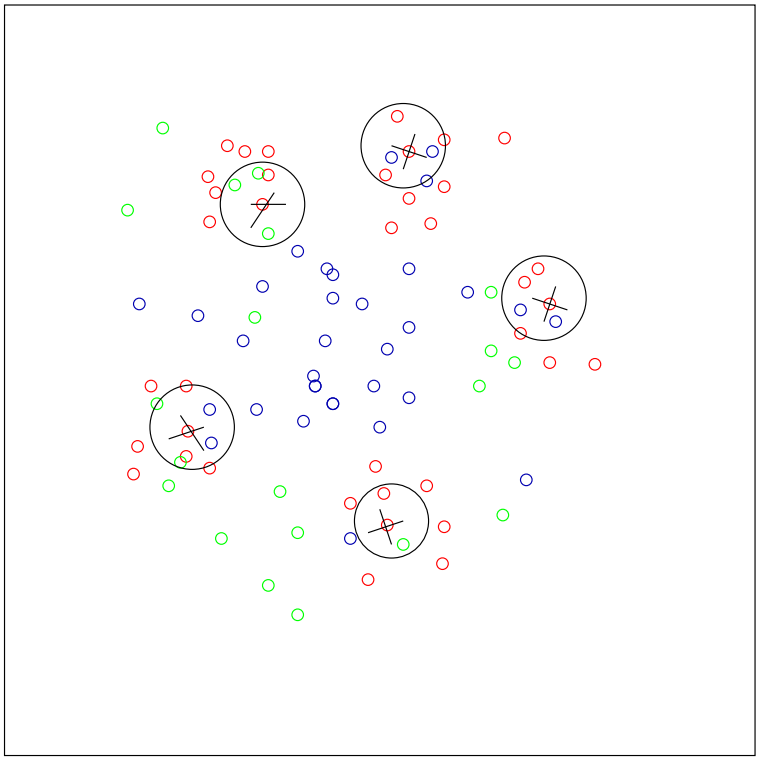
Generalised Euclidean Distance:
The distance function used in MindReader developed by Ishikawa et. al. distills information from the examples into a "best query point" and coefficients of an implied distance function.

The standard Euclidean distance has circles for isosurfaces, while the weighted Euclidean distance has ellipses aligned with the coordinate axis.
The new function describes ellipses that are not necessarily aligned with the coordinate axis.

The KNN Approach

In this method, the nearest neighbours for each of the given K query points is determined by posing nearest neighbour queries to the feature index with $f_1 \cdots f_k$.

The index returns a sorted list of pairs $<image\_id,\ similarity\_value>$ which are sorted and output.



KNN in two-dimensional feature space.

In MARS, this technique is called MES and has been used in four ways:

1. Single search

2. Balanced search (the most effective)

3. Weighted search

4. Inverse weighted search

Experimental Results

To have experimentally evaluated the above two approaches, using the Corel collection of images for query Q_3.

We manually classified images into: buildings, fishes, flowers, flower-beds, green-beds, mountains, people, plants, sea, sunset, all others.

We considered two global features: colour and texture.

Following Carson and Ogle's experimental results on human perception of colours, we used the 13 dimensional colour feature vectors.

For texture we used 16 dimensional Gabor texture feature vectors.

Results

Effective cost of KNN and ENN based on **colour**

Queries			*EC* (E_c)		*Eff Cost*(P_c)			
	KNN	ENN	KNN	ENN	KNN		ENN	
	p/R_p	p/R_p			$K=1$	$=10$	$=1$	$=10$
Flowers	$0.35/0.25$	$0.05/1.0$	50	20	62.5	**175**	**40**	220
Mount.	$0.30/0.30$	$0.20/0.70$	45	20	58.5	180	**34**	**160**
Sunset	$0.40/0.18$	$0.10/1.00$	50	20	84	**140**	**40**	220

Observations

The retrieval performance of KNN is better than ENN.

The execution cost of ENN is better than KNN.

Based on the user's preferences over retrieval performance, the overall performance is evaluated using proposed CBIR query evaluation model.

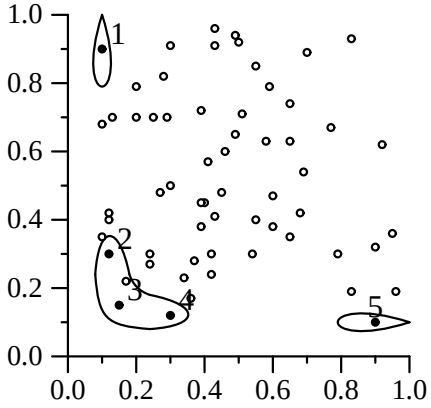
The choice of the algorithms based on the overall performance depends on the users' preference over recall/precision values.

New Taxonomy

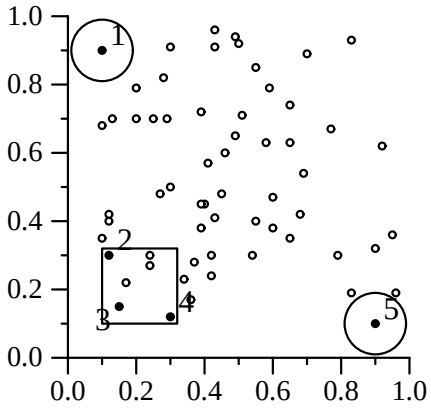
1. Single Cohesive Region(SCR): The region of interest is one continuous space.

2. Multiple Cohesive Region(MCR): The region of interest is the union of more than one continuous space, each described above.

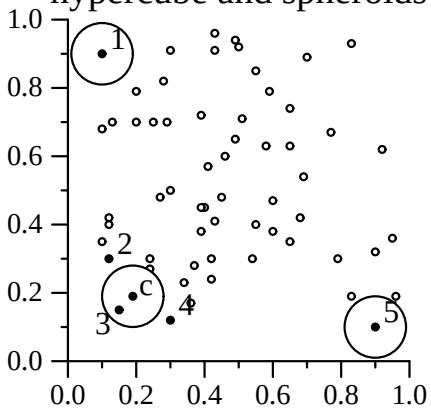
The figure shows multiple regions with different shapes of the result regions.



General



hypercube and spheroids



spheroids

QBME with Multiple Features

Such queries contain multiple query points in multiple feature spaces.

We must address the issues of:

- defining the query semantics.
- processing the query.

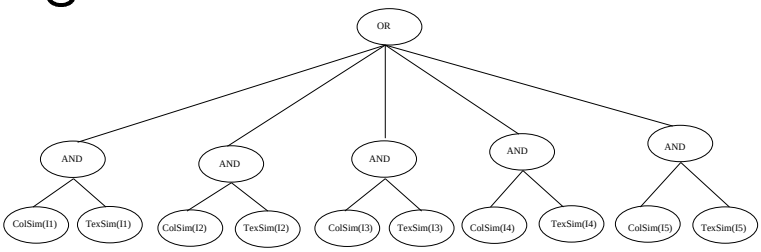
We present two ways of defining semantics and propose query processing algorithms.

Image Priority (AND-OR) Approach

In this approach, the given query is decomposed into single-example, multiple feature queries.

These are processed using the multi-step query processing algorithm.

The results are then combined using combining functions.

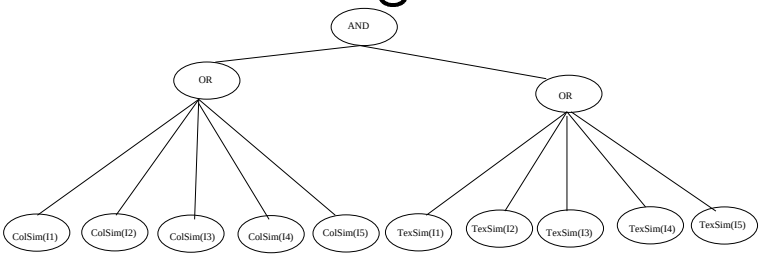


Feature Priority (OR-AND) Approach

In this approach, the given query is decomposed into separate single-feature QBME queries.

These queries are processed individually using the previously-described algorithms.

The results are then combined using appropriate combining functions.



Processing

The actual processing in both approaches is the same:

- The leaf nodes are processed using the KNN method to retrieve the k most similar images.
- The AND nodes are processed using the Multi-step algorithm.
- The OR nodes are processed using the Max function on the child nodes' results.
- The above steps are repeated until k results are obtained, when they are displayed.

Effective cost of QBME

Queries			*Execution Cost (E_c)*		*Effective Cos*		
	OR-AND p/R_p	AND-OR p/R_p	OR-AND	AND-OR	OR-AND		
					$K=1$	$K=10$	K
Flowers	$0.30/0.30$	$0.40/0.18$	399	1088	**518.7**	**12369**	12
Mountains	$0.30/0.30$	$0.30/0.30$	550	1272	**715**	**17050**	16
Sunset	$0.45/0.08$	$0.80/0.00$	436	1166	**460.08**	3924	11

Summary

In this section we discussed,

1. Approaches to processing single-feature QBME:
   - Effective Nearest Neighbours (ENN)
     - Query Reweighting
     - Query-Point Movement
     - The generalised Ellipsoid Distance Function.
   - K Nearest Neighbours (KNN)
2. Processing multiple-feature QBME:
   - Image Priority Method (AND-OR).
   - Feature Priority Method (OR-AND).

References

[1] 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), Lausanne, Switzerland, March 1995.

[2] Ronald Fagin. Combining fuzzy information from multiple systems. Proc. Fifteenth ACM Symp. on Principles of Database Systems, pages 216–226, Montreal, 1996. URL: http://www.almaden.ibm.com/cs/people/fagin/.