# AN XML BASED CONTENT-BASED IMAGE RETRIEVAL SYSTEM WITH MPEG-7 DESCRIPTORS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

SERDAR ARSLAN

IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

DECEMBER 2004

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Adnan Yazıcı
Supervisor

**Examining Committee Members**

| | |
|---|---|
| Assoc. Prof. Dr Gözde Bozdağı Akar | (METU, EEE) |
| Prof. Dr. Adnan Yazıcı | (METU, CENG) |
| Asst. Prof. Dr. Halit Oğuztüzün | (METU, CENG) |
| Assoc. Prof. Dr. Nihan Kesim Çiçekli | (METU, CENG) |
| Yakup Yıldırım | (HAVELSAN) |

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Serdar Arslan

Signature :

# ABSTRACT

AN XML BASED CONTENT-BASED IMAGE RETRIEVAL SYSTEM
WITH MPEG-7 DESCRIPTORS

Recently, very large collections of images and videos have grown rapidly. In parallel with this growth, content-based retrieval and querying the indexed collections are required to access visual information. Three main components of the visual information are color, texture and shape. In this thesis, an XML based content-based image retrieval system is presented that combines three visual descriptors of MPEG-7 and measures similarity of images by applying a distance function. An XML database is used for storing these three descriptors. The system is also extended to support high dimensional indexing for efficient search and retrieval from its XML database. To do this, an index structure, called M-Tree, is implemented which uses weighted Euclidean distance function for similarity measure. Ordered Weighted Aggregation (OWA) operators are used to define the weights of the distance function and to combine three features' distance functions into one. The system supports nearest neighbor queries and three types of fuzzy queries; feature-based, image-based and color-based queries. Also it is shown through experimental results and analysis of retrieval effectiveness of querying that the content-based retrieval system is effective in terms of retrieval and scalability.

*Keywords:* Content-Based Image Retrieval, MPEG-7 Descriptors, Color Layout, Dominant Color, Edge Histogram, M-Tree, Ordered Weighted Aggregation, XML Database

# ÖZ

MPEG-7 TANIMLAYICILARI İLE XML TABANLI İÇERİK-TABANLI GÖRÜNTÜ ERİŞİM SİSTEMİ

Son zamanlarda, çok büyük görüntü ve video veritabanları ortaya çıkmıştır. Bu büyümeye paralel olarak, görsel bilgiye erişebilmek için içerik-tabanlı erişim ve indekslenmiş veritabanları üzerinde arama yapabilme ihtiyaçları doğmaktadır. Görsel bilginin üç ana bileşeni renk, doku ve biçimdir. Bu tezde, MPEG-7 'nin üç görsel tanımlayıcısını birleştiren ve görüntülere uzaklık fonksiyonunu uygulayarak aralarındaki benzerliği ölçen XML tabanlı bir içerik-tabanlı görüntü erişim sistemi sunulmaktadır. Bu üç görsel tanımlayıcıyı tutmak için bir XML veritabanı kullanılmaktadır. Sistem aynı zamanda XML veritabanı üzerinde etkin arama ve erişim için cok-boyutlu indekslemeyi desteklemektedir. Bu indeksleme, M-Tree adı verilen indeks yapısı ile geliştirilmiştir ve M-Tree benzerlik ölçümü için ağırlıklı Euclidean uzaklık fonksiyonunu kullanmaktadır. Uzaklık fonksiyonunun ağırlıklarını hesaplamak ve üç görsel özelliğin uzaklık fonksiyonlarını bir uzaklık fonksiyonuna birleştirmek için Sıralı Ağırlıklı Toplam operatörleri kullanılmıştır. Sistem en yakın komşuluk sorgularını ve 3 tip bulanık sorguları sağlamaktadır: özellik tabanlı, görüntü tabanlı ve renk tabanlı sorgular. Ayrıca deney sonuçları ve sorguların sonuç etkinlikleri bu içerik-tabanlı erişim sisteminin erişim ve ölçeklenebilirlik bakımından etkin olduğunu göstermektedir.

*Anahtar Sözcükler:* İçerik-Tabanlı Görüntü Erişimi, MPEG-7 Tanımlayıcıları, Renk Planı, Baskın Renk, Kenar Dağılımı, M-Tree, Sıralı Ağırlıklı Toplam, XML Veritabanı

# ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor, Prof. Dr. Adnan Yazıcı for his instructive comments in the supervision of the thesis. And I also thank my family for endless love and support, METU Computer Center staff for the technical infrastructure they supply, and all other friends who helped in producing this thesis.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVATIONS

| | |
|---|---|
| ADL | Alexandria Digital Library |
| ANMRR | Average Normalized Modified Retrieval Rank. |
| AV | Audio-Visual. |
| AVR | Average Rank. |
| BR | Bounding Rectangle |
| CBIR | Content-Based Image Retrieval |
| CL | Color Layout |
| CSs | Coding Schemes |
| DC | Dominant Color |
| Ds | Descriptors |
| DSs | Description Schemes |
| DDL | Description Definition Language |
| EH | Edge Histogram |
| GiST | The Generalized Search Tree |
| GUI | Graphical User Interface |
| JSP | Java Server Page |
| k-NN | k Nearest Neighbor |
| MRR | Modified Retrieval Rank |
| MPEG | Moving Pictures Experts Group |
| NMRR | Normalized Modified Retrieval Rank |
| OWA | Ordered Weighted Aggregation |
| QBE | Query By Example |
| QBF | Query By Feature |
| QBIC | Query By Image Content |
| SP | Spatial Point |
| SQL | Structural Query Language |
| XML | Extensible Markup Language |
| XM | Experimentation Model |

# CHAPTER 1

# INTRODUCTION

"A picture is worth ten thousand words."

*—A Chinese proverb*

## 1.1 Motivation

The tremendous growth in the amount of multimedia is driving the need for more effective methods for storing, searching and retrieving digital images, video and audio data. The visual content of images can be categorized as follows: spatial, semantic, and low-level [1]. The spatial content of an image is the relative positioning of the objects in the image. The actual meaning of the image that a user captures when he/she looks at the image forms the semantic content of the image. The low-level content is formed by the low-level features such as color, shape, and texture. For indexing the images based on these low-level features, various methods exist in the literature.

Research in Content-Based Image Retrieval (CBIR) today is now concentrating on deeper problems, and can be seen as a lively discipline of computer vision, databases, and information retrieval [2].

In general, most CBIR systems suffer from several drawbacks [3]: First, feature extraction is very expensive process. Since low-level features such as color, shape, and texture are very complicated for extraction, CBIR systems should improve efficiency of this process. Second, the quality of results tends to be low. Third,

querying performance with often long reply times is unsatisfactory. Finally, user interfaces are much too complicated for average users.

The content-based image retrieval system proposed in this thesis includes the following features:

- Efficient extraction of low-level features: Low-level features (color, texture and shape features) need very complex extraction process, so a qualified CBIR system should improve the performance of feature extraction. Many researchers have used several methods to extract audio-visual features up to now, and these features were formed in various formats. However, the necessity arises for a common format, which is able to represent the audio-visual content. As a consequence, MPEG-7, formally known as Multimedia Content Description Interface is introduced as an ISO/IEC standard by MPEG (Moving Pictures Experts Group) [10] and MPEG-7 focuses on *description* of multimedia content. The key issue here is that MPEG-7 does not standardize the way to obtain these descriptions or how to use them, but only standardizes the descriptions and the way of structuring them. The emerging MPEG-7 multimedia content description standard promises to further improve content-based searching by providing a rich set of standardized tools for describing multimedia content in XML. The MPEG-7 standard enables fast and effective content-based searching by defining descriptors for color, texture, shape and other features.

- Satisfactory querying performance—CBIR systems use distance functions to calculate the dissimilarity between a search image and database images. This process is often very slow and reply times in the range of minutes may occur for large databases. Since multimedia data usually have high-dimensional properties, for example, an image might have multi-dimensional features, such as texture, color, and shape, it is very important for an indexing technique that can support execution of high-dimensional similarity queries to be invented for

multimedia databases. M-tree [15] is such a high-dimensional and distance-based index structure based on *Metric Space*.

- Satisfactory result quality—By using only general features for all types of images and asking the user to choose features leads to low quality retrieval results. Multi-features should be combined to improve the query performance.

## 1.2 Contributions

In this work, we propose an XML-based CBIR system with MPEG-7 Content Descriptors. This CBIR system consists of three modules:

### 1.2.1 Feature Extraction Module

In the multimedia processing, only the description of content is in the scope of MPEG-7 [10], not how a description is produced or consumed. Further, the descriptions are not required to allow interoperability. This leaves space for industrial and academic competition in developing new, more powerful methods for multimedia content analysis, better search engines and user applications.

For extracting low-level features from images, we use MPEG-7 reference software (XM) [11]. MPEG-7 aims at setting up a framework for describing all aspects of multimedia contents. It focuses mainly on setting up the standard low-level descriptors set and high-level abstract descriptions set. MPEG-7 XM includes low-level feature extraction methods and stores them in XML format. In these feature extraction methods, *Dominant Color (DC), Color Layout (CL)* and *Edge Histogram (EH)* features are used in this study to describe image contents.

### 1.2.2 Image Database and Indexing

Since MPEG-7 XM extracts low-level features and stores these features in XML format, we use Berkeley XML DB [14] as our image database management system. Normally, image descriptors are represented by multi-dimensional vectors, which are often used to calculate the descriptor distance in the feature space for measuring the similarity of two images. When the number of images in the database is small, a sequential linear search can provide a reasonable performance. However, with large-scale image databases, indexing support for similarity-based queries becomes necessary.

Because of using multi-dimensional features, we need an efficient access method over image database and we use M-Tree [15] [42] for this purpose. Since M-Tree is a distance-based tree structure, we need an efficient distance function to make the evaluation of similarity of images and query results better. So Euclidean Distance function [4] is used as similarity measure. In general, the CBIR systems support the combinations of features for efficient indexing and querying.

But most of these systems combine these features by associating weights to individual features. Main problem here is that the same weights are associated with the same features for all images in database and sum of these weighted features are used to build an index structure. However, when comparing two specific images, one feature can be more distinctive than the others; so that feature must be associated with higher weights. When comparing other two images, that feature may be less distinctive than the other features and for this case that feature must be associated with a lower weight. For this purpose, in this system we use *Ordered Weighted Aggregation* (OWA) [20] operators to associate variable weights with three low-level features (DC, CL, EH) and calculate a combined distance for constructing M-Tree structure.

### 1.2.3 Query Module

Image objects may have a complex inherent structure. Content-based retrieval of images is on a number of content descriptors, including color, texture, shape, relative location of image objects and regions, spatial layout, etc. To query image contents, unlike traditional SQL queries, users are usually not able to precisely characterize the objects in queries. More importantly, images with slight differences look the same from the viewpoint of users. That's why image query system should support approximate similarity search.

In traditional image retrieval systems, the query languages only deal with exact-match queries. This might be sufficient to deal with queries for metadata and annotations of multimedia data. These queries are definitely important. However, content-based information retrieval requires non-exact match (fuzzy) queries. A query is fuzzy if the properties of objects being queried cannot be certain (like red ball) or the comparison operators in the query cannot provide exact matches. Systems allow queries to be more or less satisfied by using fuzzy query paradigms. Then, the results of a query are ranked according to their degree of satisfaction [9].

There could be many ways for users to query images:

- Query by example (QBE): Users choose an image already displayed and ask for images similar to the selected one.

- Direct query: Users specified their desired image features directly.

- Query by sketch: Users roughly sketch the shapes they wish to retrieve.

- Query by painting (or query by color): Users paint a simple color image as the query specification, and those images with similar colors in the same spatial arrangement are retrieved.

Query by example (QBE) is a common retrieval paradigm in content-based image retrieval applications [8]. In a query-by-example CBIR system, the query image is usually used as a seed to retrieve similar images from the database, which can be either an existing image or a hand-drawn sketch.

In our approach, both QBE and direct query are supported. In QBE, users give an example image to the system and describe their expectation as an image-based fuzzy query like "very similar to this image" or as a feature-based query like "very similar in Color Layout and similar in Dominant Color or not similar in Edge Histogram". So the query model of our CBIR system includes fuzzy querying. In this paradigm, nearest neighbor queries are also supported like 'retrieve top 10 nearest images to the query image'.

With direct query, user must supply amount of main colors (Red, Green and Blue) on image as a similarity degree like 'retrieve images which have mainly *Red* color and very few *Green* or mostly *Blue*'. This is another type of fuzzy query in our system.

## 1.3 Organization of the Thesis

The chapters of the thesis have been organized as follows: In Chapter 2, several main components of a CBIR system are briefly discussed and previous works on CBIR systems are listed. Chapter 3 introduces MPEG-7 briefly. In Chapter 4, the content-based retrieval system that is developed in the scope of the thesis is presented. The performance experiments of the content-based retrieval system are given in Chapter 5. Finally, Chapter 6 concludes the thesis.

# CHAPTER 2

# CONTENT-BASED IMAGE RETRIEVAL

In this chapter, first main components of a CBIR system are discussed and the techniques that are used for similarity measurement are given. A survey on some of the existing multi-dimensional index structures and content-based retrieval systems is provided.

## 2.1 Overview

Content-based retrieval from image databases is a wide field of research interests. In CBIR systems, images are indexed on the basis of low-level features, such as color, texture, and shape. An ideal CBIR system should extract the semantic content of images automatically. Automatic object recognition and classification are difficult problems in image understanding and computer vision. This is the main reason why low-level features such as colors, textures, and shapes of objects are widely used for content-based image retrieval [7]. Mapping the high-level semantic concepts used by humans to understand image content to the low-level visual features extracted from images is the basic problem in CBIR. Thus two important research topics in CBIR are [5];

- Selection of the used features and the measure of similarity between them.
- Techniques for indexing the images.

A typical content-based image retrieval system is depicted in Figure 2.1 [6].

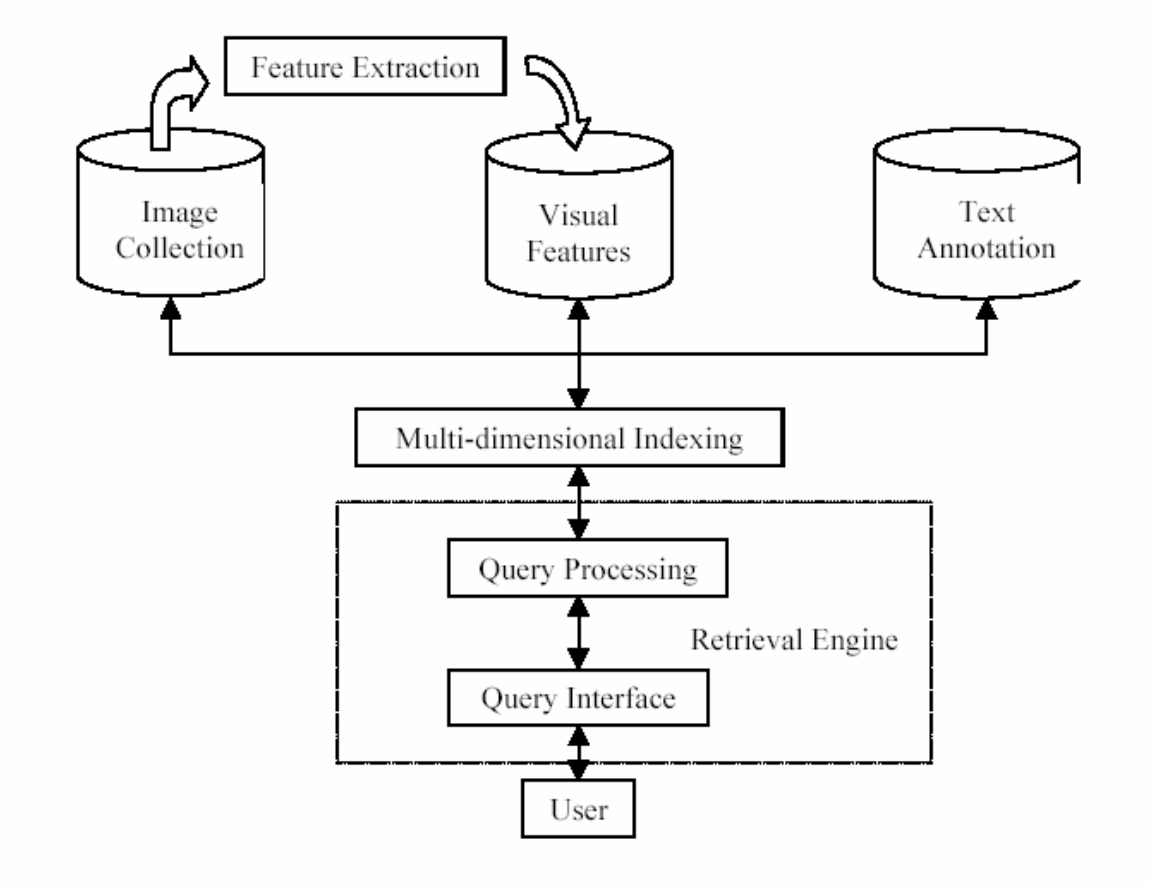


Figure 2.1: A typical content-based image retrieval system

### 2.1.1 Image Database

The image database contains images for the purpose of visual display. Unlike traditional database, image database faces many problems. Image data is often large in size and the content-based analysis is an expensive process. Thus, preprocessing is required for querying the database. Moreover image data is subjective, for a given image, it may have different interpretation for different users.

The visual feature database stores visual features extracted from images needed to support content-based image retrieval. The text annotation repository contains keywords and free-text descriptions of images [7].

### 2.1.2 Feature Extraction

Feature extraction is the basis of content-based image retrieval. Feature extraction is concerned with the detection and localization of particular feature in a multimedia object in images. The features, within the visual feature scope, can be classified as low-level features and high-level features. Low-level features include color, texture, and shape features while high-level features are application-dependent and may include, for example, human faces and fingerprints.

#### 2.1.2.1 Color

Color is one of the most recognizable elements of image content and is the most commonly used feature image retrieval because of its invariance with respect to image scaling, translation and rotation [7]. Color features are independent of image size and orientation and can be used for describing content in still images and video.

#### 2.1.2.2 Texture

Texture is widely used and refers to the visual patterns that have properties of homogeneity or not, that result from the presence of multiple colors or intensities in the image [12]. Texture features of the images can be seen as the structural information of surfaces and their relationship to the surrounding environment. There are many ways to describe texture: Statistical methods often use spatial frequency, co-occurrence matrices, edge frequency, primitive length etc. [4]. Using texture descriptors in a CBIR system provides powerful means for similarity matching and retrieval.

#### 2.1.2.3 Shape

The shape of image objects provides a powerful visual clue for similarity matching and defining the shape of an object is often very difficult. In image retrieval, it is usually required that the shape descriptor is invariant to scaling, rotation, and translation. In general, shape description can be divided into two categories [45], boundary-based and region-based. In the boundary-based shape description, only boundary information of objects is used and the boundary information is suitable to describe objects that have similar contour characteristics. In the region-based shape description, the entire shape region is used to extract a meaningful description, which is most useful when objects have similar spatial distributions of pixels in objects. Dependent on the application or objects characteristics, it is useful to employ either region- or contour-based descriptors.

### 2.1.3 Similarity Measures

Instead of exact matching, content-based image retrieval calculates visual similarities between a query image and images in a database. After extracting features of images in the database, the search results are obtained by measuring the similarity between the pre-extracted features of the image database and the query. Distances or similarities are mathematical representations of what is meant by close or similar. Accordingly, the retrieval result is not a single image but a list of images ranked by their similarities with the query image. Many similarity measures have been developed for image retrieval based on empirical estimates of the distribution of features in recent years. Different *similarity/distance measures* will affect retrieval performances of an image retrieval system significantly. The choice of distance is extremely important. In some cases, a Euclidean metric will be sensible while in others a Manhattan metric will be a better choice. Generally, some experience or subject matter knowledge is very helpful in selecting an appropriate distance for a given project.

The problem of whether the similarity distance should be a *metric* or not is not decided yet since human vision is very complex and the mechanisms of the human visual system are not fully understood. We prefer the similarity distance to be a metric and must satisfy the following properties [4]:

- *Similarity*: The distances between an image to itself should be equal to zero:

$$d(A,A) = 0; \tag{2.1}$$

- *Minimality*: An image should be more similar to itself than to other images:

$$d(A,A) < d(A,B); \tag{2.2}$$

- *Symmetry*: It is unreasonable if we say image A is similar to image B but image B is not similar to image A:

$$d(A,B) = d(B,A); \tag{2.3}$$

- *Transitivity*: It is also unreasonable if image A is very similar to image B, and B in turn very similar to C, but C is very dissimilar to A.

Many (dis) similarity measures have been proposed and we list here some of the most commonly used [4].

- *Minkowski-form distance*: If each dimension of image feature vector is independent of each other and is of equal importance, the *Minkowski-form distance* $L_p$ is appropriate for calculating the distance between two images. This distance is defined as:

$$D(x,y) = L_p = \left[ \sum_{i=1}^{d} \left| x(i) - y(i) \right|^p \right]^{\frac{1}{p}} \tag{2.4}$$

where x and y feature vectors and d is feature dimension.

- *Weighted Minkowsky-form distance***:** In this form of Minkowsky distances, the individual dimensions can be weighted differently using non-negative weights and it is defined as:

$$D(x,y) = \left[ \sum_{i=1}^{d} w_i \left| x(i) - y(i) \right|^p \right]^{\frac{1}{p}} \tag{2.5}$$

- *Euclidean distance:* The Euclidean distance is defined as:

$$D(x, y) = \sqrt{\sum_{i=1}^{d} [x(i) - y(i)]^2} \tag{2.6}$$

- *Weighted Euclidean distance:* The weighted Euclidean distance is defined as:

$$D(x, y) = \sqrt{\sum_{i=1}^{d} w_i \cdot [x(i) - y(i)]^2} \tag{2.7}$$

- *Mahalanobis distance:* The Mahalanobis distance metric is appropriate when each dimension of image feature vector is dependent of each other and is of different importance. It is defined as:

$$D(x, y) = \left| \det C \right|^{\frac{1}{d}} (x - y)^T C^{-1} (x - y) \tag{2.8}$$

where *C* is the covariance matrix of the feature vectors.

- *Generalized Euclidean distance:* This distance is a generalization of the Mahalanobis distance where the matrix **K** is positive definite but not necessarily a covariance matrix, and the multiplicative factor is omitted:

$$D(x, y) = (x - y)^T K(x - y) \tag{2.9}$$

- *Manhattan distance***:** Manhattan distance or city block defined as:

$$D(x, y) = \sum_{i=1}^{d} |x(i) - y(i)| \tag{2.10}$$

- *Chebychev distance:* it is defined as

$$D(x, y) = \left( \max_{i=1}^{d} |x(i) - y(i)| \right) \tag{2.11}$$

### 2.1.4 Multi-dimensional Indexing

An ideal CBIR system should be scalable to large image collections and should support fast retrieval. For this purpose multi-dimensional indexing is used. For an efficient similarity search in a typical CBIR system it is necessary to store the feature vectors in a multi-dimensional index structure and use the index structure to efficiently evaluate the distance metric. The multi-dimensional index structure is used must efficiently support both range and nearest neighbor queries.

There are two main classes of multi-dimensional indexes [16], vector-space methods and metric-space methods.

#### 2.1.4.1 Vector-Space Methods

Since vector spaces contain more information these methods allow a better structuring of data than general *metric spaces*. A lot of work has been done on *vector spaces* by exploiting their geometric properties, but normally these cannot be extended to general metric spaces where the only available information is the distance among objects. In contrast to metric spaces, the operations in vector spaces tend to be simple and hence the goal is mainly to reduce I/O.

#### 2.1.4.2 Metric-Space Methods

Instead of using a feature transformation into a *vector space*, data can also be directly processed using a *metric space* index structure. In this case, the user has to provide a metric distance, which corresponds to the properties of the similarity measure.

A *metric space* is a pair, M = (D, *d*) where D is a domain of feature values and *d* is a distance function with the following properties [15]:

- *Symmetry*: $d(A, B) = d(B, A)$ (2.12)
- *Positivity*: $d(A, B) > 0\ (A \neq B)$ *and* $d(A, B) = 0$ (2.13)
- *Triangle inequality*: $d(A, B) \leq d(A, C) + d(C, B)$ (2.14)

where *A*, *B* and *C* are objects in a *metric space* U, the universe.

In these methods, the distance is normally quite expensive to compute, so the general goal is to reduce the number of distance evaluations. To reduce the number of distance evaluations at query time, an index structure is built which is used to prune branches in processing the queries.

#### 2.1.4.3 Overview of Some Popular Multidimensional Index Structures

Recently, many new Bounding Rectangle (BR)-based data structures have been proposed. All of them are derived from the R-tree [17]. The R-tree suffers from a high degree of overlap among indexed subspaces and low fan-out at high dimensionalities that leads to poor query performance. The proposed data structures extend the R-tree to scale to higher dimensionality and/or support arbitrary distance metric.

The TV-tree [18] is an R-tree like data structure that exploits the fact that not all dimensions of the feature vector are necessary to discriminate among the objects. It uses a transform to achieve an ordering of the dimensions based on their discriminating power. Only the first few dimensions in that ordering, called the ``active'' dimensions, are used to define the BRs. Each BR is specified by a center, which is an n-dimensional vector where n is the number of active dimensions, and a scalar radius. The non-discriminatory dimensions are ignored. At the data node level, since it is possible for a leaf to consist of points that all agree on some of the inactive dimensions, these common dimensions are introduced into the center representation. The scalability of the TV-tree to high dimensionality relies upon the fact that there exists an ordering among the dimensions based on their discriminating power and this order is known in advance and does not change. This may not be possible in dynamic database environments.

The X-tree [19] is another R-tree like data structure with a modification of the R-tree node splitting algorithm to reduce overlap among the index nodes. If splitting a node causes a large amount of overlap, the node is not split at all, thus creating a supernode i.e., a node that spans over multiple pages on disk. The intuition is that since there is large overlap between the nodes after the split, the probability that both nodes will be accessed by a search operation is high, and hence a sequential scan over the nodes is better than random accesses to each of the nodes. As the dimensionality increases, the X-tree degenerates to a few random I/O at the higher levels and a linear

scan over the entire database at the lower levels. But, contrary to linear scan, X-tree has the overhead of performing disk management operations to create and maintain variable sized nodes on disk.

Distance based variants of the R-tree include the SS-tree [21] and M-tree [15]. The SS-tree uses k-dimensional spheres as BRs instead of k-dimensional rectangles. There are two advantages of the SS-tree over the R-tree. First, on average, the minimum distance of a query point from a BR is lower when the BRs are bounding spheres rather than bounding rectangles. Since the processing of nearest neighbor queries depends on the minimum distance, SS-tree provides better performance for nearest neighbor queries compared to R-tree. Second, since the SS-tree stores only the centroid and a scalar radius for each entry in the index node instead of the bounding rectangle, it requires only half the space compared to an R-tree entry and hence has almost twice the fan-out. But since the volume of bounding spheres is much higher compared to the volume of bounding rectangles especially at high dimensionality, the overlap between the spheres is much higher compared to the R-tree leading to poor range search performance as the dimensionality increases. To avoid the overlap problem, the SR-tree [22] maintains both the bounding rectangle like R-trees as well as the bounding sphere like the SS-tree. Therefore, it has small minimum distances like the SS-tree as well as lower overlap of the R-tree.

In M-tree [15], instead of fixing the BRs to be boxes or spheres, the data structure is parametric on the distance function. The user can provide the distance function which the M-tree will invoke as a black box to construct the BRs. In addition, M-tree exploits the triangle inequality to save several distance computations during tree traversal.

Several Spatial Point (SP)-based data structures have been proposed in the literature as well. While all BR-based data structures are paginated that is each node of the index structure implicitly corresponds to a disk page and balanced, that is not the case with SP data structures. Paginated SP-based data structures are derived from

the KDB-tree [24]. The partitioning of the indexed space is usually represented by a kd-tree [23].

Each internal node of the kd-tree represents a partition of the space. A kd-tree partition, unlike BRs, represents a clean split that the two subspaces after the split are mutually disjoint. Several SP-based data structures are described below.

The KDB-tree [24] works analogously to the B-tree but instead of nodes containing search values in disjoint intervals of a one dimensional space, each node ``covers" a brick-like region of k-dimensional space. The space partitioning within a KDB-tree node is represented using a kd-tree. Whenever a data or index node becomes full, KDB-tree chooses a single (k-1)-dimensional hyperplane to split the node into two non-overlapping subspaces. In case of data nodes, this can violate storage utilization guarantees. In case of index nodes, in addition to adversely affecting storage utilization, it also makes the splitting process itself very costly due to the cascading splits.

The hB-tree [25] is a variant to the KDB-tree. To circumvent the problems of storage utilization and cascading splits when split is performed using just one single dimension, hB-tree may use multiple dimensions to split a node. The space partitioning within a node is represented using a kd-tree. hB-tree provides guaranteed storage utilization and also avoids cascading splits. However, if a node is split using multiple dimensions, portions of kd-tree (called the ``full path") needs to be replicated at the parent and child nodes. The utilization guarantee of hB-tree does not factor in the information that is replicated at various nodes. The performance evaluation of the hB-tree shows that it performs well at medium dimensional features spaces.

There are also nonpaginated SP-based data structures. They can be either feature based which the splits are based on a feature value (VAMSplit tree, LSDh-tree) or distance based which the splits are based on distances from one or more suitably chosen pivot points (vp-tree, mvp-tree). However, their utility is

limited in the context of large dynamic databases when the entire data structure cannot reside in main memory. To circumvent the problem, some memory based data structures like the LSD-tree provide an explicit paging algorithm when the size of the directory exceeds the size of main memory. Still, these trees are not balanced and their performance is usually sensitive to presorted data.

### 2.1.5 Retrieval Engine

The search methods used for image databases differ from those of traditional databases, since query method for multimedia databases is usually retrieval-by-similarity [16]. A good query method is natural to the user as well as capturing enough information from the user to extract meaningful results. The following query methods are commonly used in content-based image retrieval research [4]:

- *Query by Example (QBE):* QBE [8] is a common retrieval paradigm in content-based image retrieval applications. With QBE, the image queries are based on example images shown either from the database itself or some external location. And the image database is to be searched and compared with this example image. The target query image can be a normal image or a user drawn sketch using graphical interface paint tools.

- *Query by Feature (QBF):* In the QBF type system, users specify queries by explicitly specifying the features they are interested in searching for. For example, a user may query an image database like ”retrieve all images, which contains 20% red pixels”. Specialized users of an image retrieval system may find this query type natural, but general users may not.

- *Query by painting (or query by color):* Users paint a simple color image as the query specification, and those images with similar colors in the same spatial arrangement are retrieved.

- *Direct query:* Users specified their desired image features directly.

Most research and commercial efforts are focused on building systems that perform well with QBE method. With QBE method, depending on application, different types of similarity queries are required. The most frequently used types of similarity queries are [7][16]:

*Range query*: find all objects that are within a specific distance from a query object*;*

*k-nearest neighbors query:* find the first k closest objects to a given query object.

## 2.2 Related Works

In the literature, a wide variety of content-based retrieval methods and systems may be found [44]. In this section, we discuss some of them.

### 2.2.1 IBM's QBIC

QBIC [26], 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 mainly queries based on example images, user-constructed sketches and drawings, and selected color and texture patterns. The color feature 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 [27]. QBIC's texture feature is an improved version of the Tamura texture representation [28]; combinations of coarseness, contrast, and directionality [29]

Shape features in QBIC consist of area, circularity, eccentricity, and major axis orientation, plus a set of algebraic moment invariants [27] [30]. QBIC is one of the few systems, which takes into account the high dimensional feature indexing. In its indexing subsystem, KLT is the first used to perform dimension reduction and then *R**-tree is used as the multidimensional indexing structure [27][31]. In its new system, text-based keyword search can be combined with content-based similarity search.

### 2.2.2 Netra

Netra is a prototype image retrieval system developed in the UCSB Alexandria Digital Library (ADL) project [32]. Netra uses color, texture, shape, and spatial location information in the segmented image regions to search and retrieve similar regions from the database. Main research features of the Netra system are its Gabor filter-based texture analysis, neural net-based image thesaurus construction and edge flow-based region segmentation.

### 2.2.3 Photobook

Photobook [33] 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 on the basic of the corresponding features in each of the three sub-books. More recent version of Photobook includes the human users in the image annotation and retrieval loop. The motivation for 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 effective in interactive image annotation.

### 2.2.4 RetrievalWare

RetrievalWare is a content-based image retrieval engine developed by Excalibur Technologies Corp. Its recent search engine uses color, shape, texture, brightness, color layout, and aspect ratio of the image, as query features. It also supports the combinations of these features and allows the users to adjust the weights associated with each feature.

### 2.2.5 Virage

Virage is a content-based image search engine developed at Virage Inc. Similar to QBIC, Virage [34] 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. The system is available as an add-on to existing database management systems such as Oracle or Informix.

### 2.2.6 VisualSeek and WebSeek

VisualSEEk [35] is a visual feature search engine and WebSEEk [36] is a World Wide Web oriented text/image search engine, both of which have been developed at Columbia University. Main research features are spatial relationship query of image regions and visual feature extraction from compressed domain. The visual features used in their systems are color sets and wavelet transform-based texture features. To speed up the retrieval process, they also developed binary tree-based indexing algorithms. VisualSEEk supports queries based on both visual features and their spatial relationships. This enables a user to submit a sunset query as red-orange color region on top and blue or green region at the bottom as its ”sketch”.

WebSEEk is a web-oriented search engine. It consists of three main modules, i.e., image/video collecting module, subject classification and indexing module, and search, browse, and retrieval module. It supports queries based on both keywords and visual content.

# CHAPTER 3

# MPEG-7

In this chapter, a brief discussion on MPEG-7 is given and Descriptors in MPEG- are described.

## 3.1 Introduction

MPEG-7, formally known as Multimedia Content Description Interface, is introduced as an ISO/IEC standard by MPEG (Moving Pictures Experts Group) to represent the audio-visual content [10]. While the prior standards (MPEG-1, MPEG-2, and MPEG-4) focus on coding and representation of audio-visual content, MPEG-7 focuses on description of multimedia content. The key issue here is that MPEG-7 does not standardize the way to obtain these descriptions or how to use them, but only standardizes the descriptions and the way of structuring them.

The content-based indexing and retrieval of audio-visual information is the main application for MPEG-7. MPEG-7 achieves these goals by defining a set of methods and tools for different aspects of multimedia description.

The MPEG-7 Visual *Descriptors (Ds)* describe basic audiovisual content of media based on visual information. These MPEG-7 visual descriptors can be used to search, filter, or browse visual material based on suitable similarity measures. Weighted combination of visual descriptors can be used in implementation of CBIR system, to make the system more effective and for this purpose, MPEG-7 also defines *Description Schemes (DSs).* These schemes specify the types of the descriptors that

can be used in a given description, and the relationships between these descriptors or between other DSs.

The *Description Definition Language (DDL)* forms a core part of the MPEG-7 standard. With DDL, users can create their own Description Schemes and Descriptors. The DDL defines the syntactic rules to express and combine Description Schemes and Descriptors. The DDL must satisfy the MPEG-7 DDL requirements [13]. It has to be able to express spatial, temporal, structural, and conceptual relationships between the elements of a DS, and between DSs.

## 3.2 Scope of the MPEG-7

Searching, indexing, filtering, and access of audio-visual (AV) content are goals of the MPEG-7 standard. So MPEG-7 standard is used in devices and applications that deal with AV content description. MPEG-7 specifies the description of features related to the AV content. As illustrated in Figure 3.1, the scope of the standard is to define the representation of the description. Feature extraction is outside the scope of the MPEG-7. Search and query also are outside the scope of the MPEG-7 since they could be application dependent. However, in order to guarantee interoperability for some low-level features, MPEG-7 also specifies part of the extraction process. Future improvements can be included in MPEG-7 compliant applications.

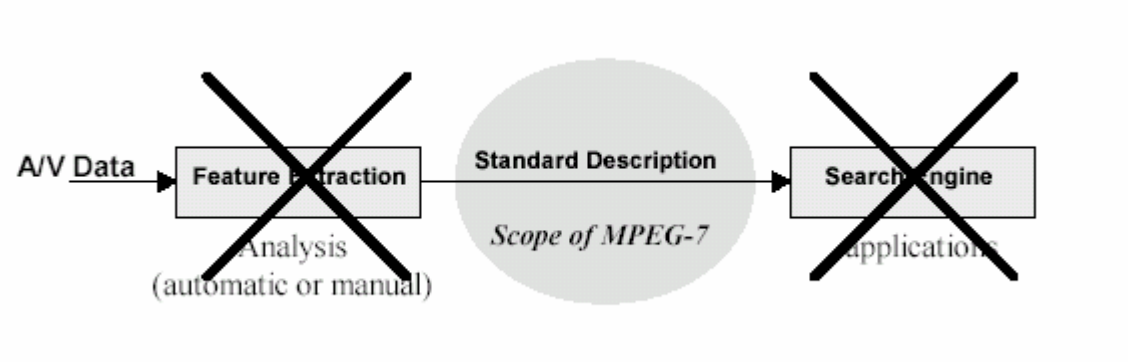


Figure 3.1: Scope of the MPEG-7

## 3.3 MPEG-7 Visual Descriptors

The main objective of MPEG-7 visual descriptors is to provide a standardized description of image or video to use in applications to identify, categorize or filter images or videos. The MPEG-7 visual descriptors are classified into general visual descriptors and domain-specific descriptors. The former include color, texture, shape and motion features, while the latter are application dependent and include a face-recognition descriptors. A brief description of each descriptor is given below.

### 3.3.1 Color Descriptors

Color is one of the most widely used and extensively studied features in content-based image retrieval. MPEG-7 provides 7 color descriptors [37]:

- *Color Space:* This descriptor allows a selection of a color space to be used in the description. In the current description, the following color spaces are supported:
    - R, G, B
    - Y, Cr, Cb
    - H, S, V
    - HMMD
    - Linear transformation matrix with reference to R, G, B
    - Monochrome
- *Color Quantization:* This descriptor specifies the partitioning of the given color space into discrete bins. Color Space Descriptor and Color Quantization Descriptor are used in conjunction with other color descriptors.

- *Dominant Color:* This descriptor allows specification of a small number of dominant color values that is the percentage of each quantized color and a spatial coherency. Its purpose is to provide an effective, compact and intuitive representation of colors present in a region or whole image.

- *Scalable Color:* The Scalable Color Descriptor is a Color Histogram in HSV Color Space with fixed color space quantization. It uses a Haar transform coefficient encoding. This descriptor is useful for image-to-image matching and retrieval based on color feature.

- *Color Layout:* This descriptor captures the spatial layout of the representative colors on a region or image. Representation is based on coefficients of the Discrete Cosine Transform. This is a very compact descriptor being highly efficient in fast browsing and search applications. It provides image-to-image matching as well as ultra high-speed sequence-to-sequence matching.

- *Color Structure:* Color Structure Descriptor captures both color content and information about the structure of this content. Its main functionality is image-to-image matching and aims at identifying localizing color distributions using a small structure window.

- *Group of Frames or Group of Pictures:* This descriptor is an extension of the scalable color descriptor to a group of frames in a video or a collection of pictures. This descriptor is based on aggregating the color properties of the individual images or video frames.

### 3.3.2 Texture Descriptors

There is three texture Descriptors [37]: Homogeneous Texture, Edge Histogram, and Texture Browsing.

- *Homogenous Texture:* The Homogeneous Texture descriptor provides a precise quantitative description of a texture that can be used for accurate search and retrieval. This descriptor is useful for similarity retrieval and it is quite effective in characterizing homogeneous texture regions.

- *Texture Browsing:* Texture Browsing is defined for coarse level texture browsing. It provides a perceptual characterization of texture, similar to a human characterization, in terms of regularity, coarseness and directionality of the texture pattern. Since the browsing descriptor relates closely to human characterization, it can also be manually instantiated. This representation is useful for browsing applications and coarse classification of textures.

- *Edge Histogram:* This descriptor captures spatial distribution of edges in an image. The edge histogram descriptor represents the spatial distribution of five types of edges, namely four directional edges and one non-directional edge. Since edges play an important role for image perception, it can retrieve images with similar semantic meaning. Thus, it primarily targets image-to-image matching. Its effectiveness is demonstrated on image data that are not necessarily homogeneously textured, for example, nature images, sketch images and clip art images.

### 3.3.3 Shape Descriptors

There are three shape Descriptors [37]: Region Shape, Contour Shape, and Shape 3D.

- *Region Shape:* This descriptor takes into account all pixels constituting the shape, which are both the boundary and interior pixels. It is applicable to objects consisting of a single connected region or multiple regions, possibly

with holes. This descriptor performs well where region-based similarity is important.

- *Contour Shape:* Contour Shape Descriptor captures characteristic shape features of an object or region based on its contour.

- *Shape 3D:* It is targeted to search and retrieve and browse 3D models. It aims at providing and intrinsic shape description of 3D models.

### 3.3.4 Motion Descriptors

The main aim of motion-based indexing and of MPEG-7 in particular is to capture essential motion characteristics into effective descriptors. There are four motion Descriptors [37]: Camera Motion, Motion Trajectory, Parametric Motion, and Motion Activity.

- *Camera Motion:* This descriptor characterizes 3-D camera motion parameters. It is based on 3-D camera motion parameter information, which can be automatically extracted or generated by capture devices.

- *Motion Trajectory:* This descriptor is an object-oriented descriptor. It describes the displacement of objects in time. It records the path of the moving object.

- *Parametric Motion:* This descriptor addresses the motion of objects in video sequences, as well as global motion. It represents the motion and/or deformation of a region or image by classic parametric model.

- *Motion Activity:* The activity descriptor captures intuitive notion of “intensity of action” or “pace of action” in a video segment and used to describe the level or intensity of activity, motion, or action in that video segment.

### 3.3.5 Face Descriptor

Face recognition can be used in image and video retrieval. The MPEG-7 Face Descriptor can be used to retrieve face images which match a query face image. The descriptor represents the projection of a face vector onto a set of basis vectors, which span the space of possible face vectors. These basis vectors are derived from eigenvector of a set of training faces and are reasonably robust to view-angle and illumination changes.

### 3.3.6 Combination of Visual Descriptors

In [48], visual content descriptors, which are extracted with MPEG-7-descriptors, are analyzed from the statistical point of view. For the analysis, three media collections were used and eight basic visual descriptors were applied on them. These media collections contain monochrome textures, color images, which form our test set (Corel dataset) and artificial color images with few color gradations. The main results show that the best descriptors for combination are Color Layout, Dominant Color, Edge Histogram and Texture Browsing. The others are highly dependent on these. In this thesis, combination of Color Layout, Dominant Color and Edge Histogram is used to describe visual content of the images. The detailed description of these descriptors is given in Chapter 4.

# CHAPTER 4

# CBIR SYSTEM

In this chapter, proposed XML-based CBIR system with MPEG-7 Content Descriptors is explained. And also this CBIR system's three modules are presented.

## 4.1 Overview

In this work, we developed a content-based image retrieval (CBIR) system by using MPEG-7 software and overall structure of the system is shown in Figure 4.1. The first process of the system is extracting visual features from images such as Dominant Color Descriptor, Color Layout Descriptor, and Edge Histogram Descriptor. MPEG-7 reference software (XM) [11] includes these low-level feature extraction methods and stores them in XML format. After extracting process, an XML database, Berkeley DB, is used to store these features.

The second part of the system consists of indexing the XML database for efficient retrieval of the query results. For this purpose we use a metric indexing technique called M-Tree. M-tree project is implemented by using The Generalized Search Tree (GiST) [41]. GiST provides a nice framework for a fast and reliable implementation of search trees. An advantage of GiST is that the basic data structures and algorithms as well as main portions of the concurrency and recovery code can be reused.

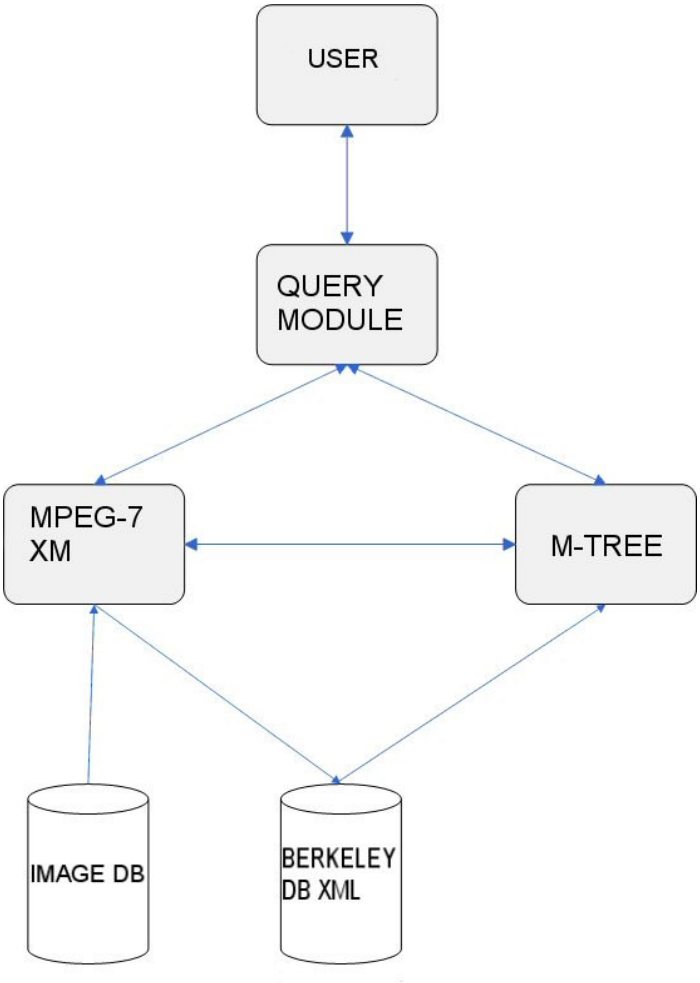


Figure 4.1: CBIR System Architecture

.

The M-tree Project [42] provides M-tree implementation classes. Only objects are needed be defined in the tree. Since M-Tree is a distance-based tree structure, the CBIR system must provide a metric distance function to find a dissimilarity (or similarity) value between each image for comparing them. In our CBIR system, Euclidean Distance is used as distance function to create M-Tree. And for objects, we use the image name as object id.

Since there are three low-level features that represent the image content, the system evaluates different distance value for each feature. But the system has to

compute an overall distances of these three distance values. For multi features, in general, weighted Euclidean Distance function is proposed to combine distances to one distance. For this purpose, we use OWA operator as mentioned in Section 4.4.

To create the tree, following initial parameters must be supplied to the system:

- DBSIZE: which holds image number in the database
- MIN_UTIL: which is the minimum utilization [15] value of the M-tree node, and must be in [0, 1].
- TYPE of WEIGHT: This specifies the weights of distances, OWA or equal weights.

The system creates the tree and is ready for online querying and retrieval. Query module is implemented by using both MPEG-7 and M-Tree software. Since content-based information retrieval requires non-exact match (fuzzy) queries, which go beyond the traditional approaches, we use fuzziness in query module. When user gives an example image to search the database with QBE paradigm, the system extracts the same three features from query image by using MPEG-7 XM Software again. The user also must supply the query type, which may be;

- feature-based fuzzy query
- image-based fuzzy query
- color-based fuzzy query
- k-nearest neighbor query.

For querying the M-tree, following parameters must be given to the system:

- IMAGE: query image (if color-based fuzzy query then this parameter is not important)

- QUERY_TYPE: type of the query

- QUERY VALUE(s): If the query is a nearest neighbor query, this value is *k* value (number of returned objects). If the query is a fuzzy query, then similarity value (for whole image or for each feature or for each main color) is supplied by this value.

The system starts to search the database by using M-tree with extracted features of a query image to retrieve the images according to a query type. And finally the result objects are taken from XML DB and these objects are shown to the user as ranked by their degrees of satisfying the query object.

## 4.2 Feature Extraction

The MPEG-7 framework consists of Descriptors (Ds), Description Schemes (DSs), a Description Definition Language (DDL), and coding schemes. Descriptors are the features or attributes of multimedia data such as color, texture, textual annotations, and media format. Description schemes represent more complex structures and other description schemes. The description definition language allows defining and extending descriptors and description schemes.

The eXperimentation Model (XM) software [11] is the simulation platform for the MPEG-7 Descriptors (Ds), Description Schemes (DSs), Coding Schemes (CSs), and Description Definition Language (DDL). The XM applications are formed by the data structures and the procedural code together and are divided in two types: the server (extraction) applications and the client (search, filtering and/or transcoding) applications.

The modules of the XM software are designed in a way that all modules are using specified interfaces [37] to reuse and to combine individual modules in bigger application. This also allows easy navigation through all the different modules for the various Ds and DSs. XM applications are related to one particular descriptor or description scheme. There are two type of applications; server applications and client applications. Server applications create the descriptor (D) or description scheme (DS) that they are testing. On the other hand, client applications use the D or DS under test. Server applications are needed if the D or DS is a low-level descriptor. Low-level descriptors can be extracted from the multimedia content applying an automatic process.

From MPEG-7 Color Descriptors, Dominant Color Descriptor, and Color Layout Descriptor are chosen for our system. And also to increase the efficiency of the system, a texture descriptor, Edge Histogram Descriptor, is added to these color descriptors.

### 4.2.1 Color Layout (CL)

*Color Layout* [37] specifies a spatial distribution of colors for high-speed retrieval and browsing at very small computational costs. It provides image-to-image matching as well as sequence-to-sequence matching. This descriptor captures the layout information of color feature. Descriptor is extracted from an 8x8 array of local dominant colors determined from the 64 (8x8) blocks the image is divided into [38]. Descriptors are matched with a tailored similarity metric.

The advantages of this descriptor are [1] [37]:

- That there are no dependency on image/video format, resolutions, and bit-depths. The descriptor can be applied to any still pictures or video frames even though their resolutions are different. It can be also applied both to a whole

image and to any connected or unconnected parts of an image with arbitrary shapes.

- That the required hardware/software resource for the descriptor is very small. It needs as law as 8 bytes per image in the default video frame search, and the calculation complexity of both extraction and matching is very low. It is feasible to apply this descriptor to mobile terminal applications where the available resources is strictly limited due to hardware constrain.

- That the captured feature is represented in frequency domain, so that users can easily introduce perceptual sensitivity of human vision system for similarity calculation.

- That it supports scalable representation of the feature by controlling the number of coefficients enclosed in the descriptor. The user can choose any representation granularity depending on their objectives without interoperability problems in measuring the similarity among the descriptors with different granularity. The default number of coefficients is 12 for video frames while 18 coefficients are also recommended for still pictures to achieve a higher accuracy

Example XML Document for Color Layout looks like;

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
        <Mpeg7 xmlns = "http://www.mpeg7.org/2001/MPEG-7_Schema" xmlns:xsi =
        "http://www.w3.org/2000/10/XMLSchema-instance">
            <DescriptionUnit xsi:type = "DescriptorCollectionType">
                <Image name = "0.jpg">
                    <Descriptor xsi:type = "ColorLayoutType">
                        <YDCCoeff>15</YDCCoeff>
                        <CbDCCoeff>28</CbDCCoeff>
                        <CrDCCoeff>32</CrDCCoeff>
```

```
                    <YACCoeff5>13  12  12  12  14 </YACCoeff5>
                    <CbACCoeff2>17  19 </CbACCoeff2>
                    <CrACCoeff2>15  18 </CrACCoeff2>
                </Descriptor>
            </Image>
        </DescriptionUnit>
    </Mpeg7>
```

The ColorLayout descriptor uses the YCbCr color space with quantization to 8 bits performed in the following way [2]:

$$\begin{aligned} Y &= 219*Ynorm + 16 \\ Cb &= 224*Cbnorm + 128 \\ Cr &= 224*Crnorm + 128 \end{aligned} \qquad (4.1)$$

Here, the Ynorm, Cbnorm and Crnorm are the normalized YCbCr color values.

The meanings of each tag are;

*Name* is image name. To add this tag into descriptors, MPEG-7 XM extracting utilities had been modified.

*YDCCoeff, YACCoeff, CbDCCoeff, CbACCoeff, CrDCCoeff* and *CrACCoeff* specify the integer arrays that hold a series of zigzag-scanned DCT coefficient values.

*YDCCoeff* is the first quantized DCT coefficient of the Y component.

*CbDCCoeff* is the first quantized DCT coefficient of the Cb component.

*CrDCCoeff* is the first quantized DCT coefficient of the Cr component.

*YACCoeff* is the second and the successive quantized DCT coefficients of the Y component. In the DDL representation, separate elements (YACCoeff2, YACCoeff5,

YACCoeff9, YACCoeff14, YACCoeff20, YACCoeff27 and YACCoeff63) are used to cover all valid array lengths.

*CbACCoeff* is the second and the successive quantized DCT coefficients of the Cb component. In the DDL representation, separate elements (CbACCoeff2, CbACCoeff5, CbACCoeff9, CbACCoeff14, CbACCoeff20, CbACCoeff27 and CbACCoeff63) are used to cover all valid array lengths.

*CrACCoeff* is the second and the successive quantized DCT coefficients of the Cr component. In the DDL representation, separate elements (CrACCoeff2, CrACCoeff5, CrACCoeff9, CrACCoeff14, CrACCoeff20, CrACCoeff27 andCrACCoeff63) are used to cover all valid array lengths.

### 4.2.2 Dominant Color (DC)

*Dominant Color* [37] specifies a set of dominant colors in any arbitrary shaped region. Color quantization is used to extract a small number of representative colors in each region or image. Descriptors are matched with a spatial coherency measure. DC is suitable for representing local features (objects or image regions), where a small number of colors are sufficient to characterize color content. Whole images are also applicable [38].

Example XML Document for Dominant Color looks like;

```
<?xml version='1.0' ?>
<Mpeg7 xmlns = http://www.mpeg7.org/2001/MPEG-7_Schema xmlns:xsi =
"http://www.w3.org/2000/10/XMLSchema-instance">
    <DescriptionUnit xsi:type = "DescriptorCollectionType">
        <Image name = "img0.jpg">
            <Descriptor size = "4" xsi:type = "DominantColorType">
                <SpatialCoherency>0</SpatialCoherency>
```

```
                    <Values>
                        <Percentage>3</Percentage>
                        <ColorValueIndex>1  0  0 </ColorValueIndex>
                    </Values>
                    <Values>
                        <Percentage>3</Percentage>
                        <ColorValueIndex>15  16  15 </ColorValueIndex>
                    </Values>
                    <Values>
                        <Percentage>1</Percentage>
                        <ColorValueIndex>10  14  20 </ColorValueIndex>
                    </Values>
                    <Values>
                        <Percentage>11</Percentage>
                        <ColorValueIndex>21  21  20 </ColorValueIndex>
                    </Values>
                </Descriptor>
            </Image>
        </DescriptionUnit>
    </Mpeg7>
```

The meanings of each tag are;

*name* is image name.

*Size* is the number of dominant colors in the region. The maximum allowed number of dominant colors is 8, the minimum number of dominant colors is 1.

*SpatialCoherency* [37] specifies the spatial coherency of the dominant colors described by the descriptor. It is computed as a single value by the weighted sum of per-dominant-color spatial coherencies. The weight is proportional to the number of pixels corresponding to each dominant color. Spatial coherency per dominant color captures how coherent the pixels corresponding to the dominant color are and whether they appear to be a solid color in the given image region. In Figure 4.2, red

pixels in the left image have low spatial coherency and in the right image high spatial coherency. 0 is used to signal that this element is not computed (note that if it is not computed it does not mean that the spatial coherency is low).

Figure 4.2: Examples of low (a) and high (b) spatial coherency of color.

*Percentage* specifies the percentage of pixels that have the associated color value. The percentage value is uniformly quantized to 5 bits with 0 corresponding to 0 percentages and 31 corresponding to 100%. Note that the sum of the Percentage values for a given visual item does not have to be equal to 100%.

*Index i*s the index of the dominant color. In this thesis, index is represented by 5-bits.

Since MPEG-7 XM software needs parameters for extracting DC Descriptor, we are expected to give some initial values for;

- *ColorSpacePresent*: This field indicates the presence of the ColorSpace element. The following color spaces are supported [37]:
  - RGB
  - YcbCr
  - HSV
  - HMMD
  - Linear transformation matrix with reference to RGB
  - Monochrome

we set this parameter to 0,so ColorSpace is not present and RGB color space is used.

- *ColorQuantizationPresent*: This element signals the presence of the ColorQuantization element. This element is only present in the binary representation,so we set this parameter to 0, ColorQuantization is not present.

- *VariancePresent***:** This field indicates the presence of the color variances in the descriptor and is only present in the binary representation,so is set to 0.

- *SpatialCoherency*: is set to 0, so this element is not computed.

### 4.2.3 Edge Histogram (EH)

*Edge Histogram* [37] captures the spatial distribution of edges, which are grouped into five categories: vertical, horizontal, 45o diagonal, 135o diagonal and isotropic, (four directional edges and one non-directional edge, Figure 4.3).

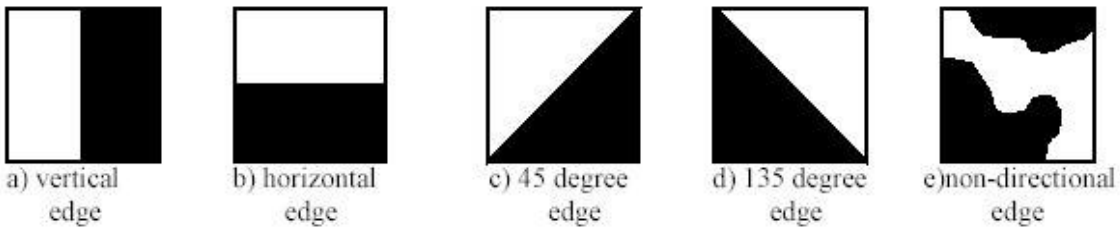


Figure 4.3: Five Categories of Edges

This descriptor primarily targets image-to-image matching (by example or by sketch), especially for natural images with non-uniform edge distribution, since it can retrieve images with similar semantic meaning. The input image is divided into 4x4

sub-images and the frequency of each type of edge is determined in each sub-image, resulting in 80 (16x5) bin local edge histogram.

The image retrieval performance can be significantly improved if the edge histogram descriptor is combined with other Descriptors such as the color histogram descriptor [1].

Example XML Document for Edge Histogram looks like;

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
    <Mpeg7 xmlns = http://www.mpeg7.org/2001/MPEG-7_Schema xmlns:xsi =
    "http://www.w3.org/2000/10/XMLSchema-instance">
        <DescriptionUnit xsi:type = "DescriptorCollectionType">
        <Image name = "img0.jpg">
            <Descriptor xsi:type = "EdgeHistogramType">
                <BinCounts>
                    0 2 5 2 4 4 0 7 2 3 6 0 4 3 6 6 0 3 7 5 1 1 5 2 6 4 2 5
                    5 6 5 1 4 5 6 5 1 4 7 5 3 2 5 4 5 4 2 5 2 5 6 2 5 3 6 3
                    1 7 5 4 6 0 2 5 3 4 1 2 3 6 4 0 7 0 6 5 2 6 3 3
                </BinCounts>
            </Descriptor>
        </Image>
        </DescriptionUnit>
    </Mpeg7>
```

Here *BinCounts* is 3-bit representation of 80 edge histogram values.

### 4.2.4 Feature Extraction Process

In this CBIR system, extracting these three low-level features, CL, DC and EH, is done offline. Firstly the image collection is supplied from Corel Database [39]. There are ten categories in image collection, Architecture, Beach, Bus, Elephant,

Flower, Food, Dinosaur, Horse, Human and Mountain and each one has 100 images. For each category, CL, DC and EH Descriptors are extracted by using MPEG-7 XM Software. After creating each feature XML documents separately, we insert them in our XML database manually.

MPEG-7 XM Software is also used in the process of querying the database. Since this CBIR system uses Query By Example paradigm, the same steps in creating XML documents of each feature for an image collection are applied to the query image. The query image is given to client application of MPEG-7 XM Software as a parameter and three features are extracted from that image and stored in a text document for further processing. In standard client application of MPEG-7 XM Software has a searching module for querying but we excluded this module from client application.

## 4.3 Image Database

We use Berkeley DB XML for storing XML Documents. Berkeley DB XML is an open source native XML DB [14] and we can make XPath queries over it.

Berkeley DB XML is specifically designed to store and manage XML data in its native format. Berkeley DB XML is implemented as C++ library on top of Berkeley DB, which provides fast, reliable, scalable, and mission-critical database support. In Figure 4.4, Berkeley DB XML system architecture is shown. Berkeley DB XML provides the following functionality [14]:

- "Embedded: Berkeley DB XML is a library and this library can be linked into the client application to increase performance by eliminating communication among processes or systems. The Berkeley DB XML library exposes API's that enable C++ and Java applications to interact with the XML data containers." Figure 4.4 shows the Berkeley DB XML system architecture

- "Document Storage: Within Berkeley DB XML, documents are stored in containers."

- "Native Storage: client application retrieves the documents exactly as they were stored. "

- "Indexing: Index is defined at the container level and a container may have multiple indices. Berkeley DB XML offers effective and flexible indexing functionality that gives application developers powerful control over query performance. "

- "Query Processing: Berkeley DB XML queries are expressed as XPath expressions."

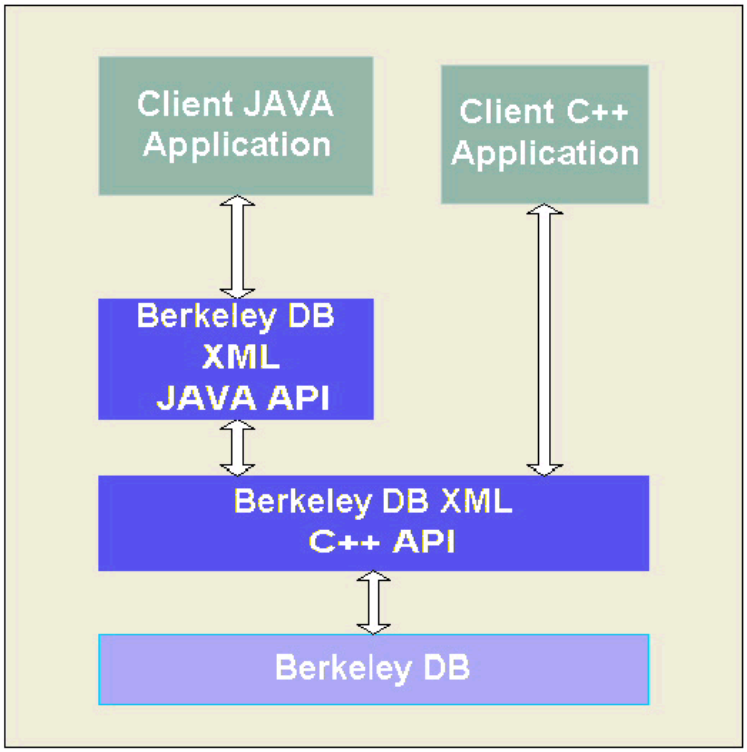


Figure 4.4: Berkeley XML DB system architecture

- "Threading: Berkeley DB XML is thread-safe, and supports multithreaded and multiprocess applications. "

- "Standards: Berkeley DB XML is implemented to conform to the W3C standards for XML, XML Namespaces, and XPath 1.0. "

In Berkeley DB XML, we store XML documents of DC, CL and EH features of an image collection, separately. Extracting these features from image collection and creating the image database is done offline. For each collection, there are 100 images' features in one XML document. Because this CBIR system includes three features, three XML documents are created and stored in Berkeley DB XML. So to query an image over this image database, system firstly queries relevant XML Document then continues to query process over that document.

Berkeley XML DB supports insertion/deletion of XML documents but our system does not include these functionalities yet.

## 4.4 Similarity Measurement

### 4.4.1 Ordered Weighted Averaging (OWA) Operator

An OWA operator [20] of dimension n is a mapping:

$$F : R^n \rightarrow R, \tag{4.2}$$

that has an associated weighting vector W

$$\mathrm{W} = [w_1 w_2 ... w_n]^{\mathrm{T}} \tag{4.3}$$

such that

$$\sum_{i=1}^{n} w_i \quad \text{where } w_i \in [0,1] \tag{4.4}$$

and where

$$F(a_1,...,a_n) = \sum_{i=1}^{n} w_i \cdot b_i \tag{4.5}$$

where $b_i$ is the $i$ th largest element of the collection of the aggregated objects $a_1,\ldots,a_n$. The function value $F(a_1,\ldots,a_n)$ determines the aggregated value of arguments, $a_1,\ldots,a_n$. For example, assume

$W = [0.4\ 0.3\ 0.2\ 0.1]$

Then,

$F(0.7, 1, 0.3, 0.6) = (0.4)(1) + (0.3)(0.7) + (0.2)(0.6) + (0.1)(0.3) = 0.76$ .

A fundamental aspect of the OWA operator is the re-ordering step, in particular an argument $a_i$ is not associated with a particular weight $w_i$ but rather a weight $w_i$ is associated with a particular ordered position $i$ of the arguments. A known property of the OWA operators is that they include the Max, Min and arithmetic mean operators.

### 4.4.2 Distance Function

In general, similarity evaluation of query object with respect to the object in database is done by applying some distance function to these two objects. In this case, what is actually measured is the distance between feature values, so distance function returns a dissimilarity value between two objects. It means that high distances correspond to low scores and low distances correspond to high scores.

Commonly used distance function is Minkowski-form distance (*Lp*):

$$D(x,y) = \left[ \sum_{i=1}^{d} w_i \left| x(i) - y(i) \right|^p \right]^{\frac{1}{p}} \quad (4.6)$$

where x and y feature vectors and d is feature dimension. If

- p = 1, $L_1$ is Manhattan or city-block distance
- p = 2, $L_2$ is Euclidean distance
- p = ∞, $L_\infty$ is maximum distance

In this study, we have implemented two versions of M-Tree. In both versions distance evaluation is carried out by Euclidean distance function. Euclidean distance is a metric distance, which is needed for M-Tree.

Since there are three low-level features that represent the image content, the system evaluates different distance value for each feature. But the system has to compute an overall distances of these three distance values. For this purpose we use OWA operator.

To compute an overall distances of three distance values, first system computes distances of each feature and then finds the maximum and minimum distances. Finally, the system applies OWA operation to these distances.

#### 4.4.2.1 M-Tree with Non-Fuzzy Dominant Color Distance

In this version of M-Tree, CL, DC and EH distances are computed by applying Euclidean distance function. For CL feature, the distance function is as follows:

$$D_{CL} = \sqrt{\sum_{i=0}^{5}\left(YCoeff[i] - YCoeff'[i]\right)^2} + \sqrt{\sum_{i=0}^{2}\left(CbCoeff[i] - CbCoeff'[i]\right)^2} + \sqrt{\sum_{i=0}^{2}\left(CrCoeff[i] - CrCoeff'[i]\right)^2} \quad (4.7)$$

And for DC feature, distance function is:

$$D_{DC} = \sqrt{\sum_{i=0}^{n}\sum_{j=0}^{n}\sum_{k=0}^{n}(Percentage[i][j][k] - Percentage'[i][j][k])^2} \quad (4.8)$$

where n = 31 and for EH feature, distance function is:

$$D_{EH} = \sqrt{\sum_{i=1}^{n}(Bincounts[i] - Bincounts'[i])^2} \quad (4.9)$$

where n = 80.

After computing distance values, each distance value is normalized to make the distance more straightforward, after that the range should be from 0(similar) to 1(dissimilar). To apply normalization, the system needs maximum and minimum values of the distances, after that maximum distance is set as upper bound, which is 1, and minimum distance as lower bound, which is 0. So for each feature there are two special distance values evaluated; maximum distance value and minimum distance value.

Normalization the distances really cause some trouble**.** Because normally we can calculate the distance of two images just from their only features, but in order to do normalization, we need to calculate all the distances in the whole database to find

maximum and minimum values, which surely reduce the performance heavily. To avoid computing all the distances in the whole database, system finds maximum and minimum distance values while creating M-Tree, because there is a lot of distance calculation in construction of the tree. If, in query phase, a distance exceeds maximum or minimum, then this distance value is set to 1 or 0.

To compute an overall distance between two images, the system firstly computes CL, DC and EH distances and applies normalization each of them separately. After normalization of each feature's distances, the system computes overall distance value from these three distances by using OWA operator.

#### 4.4.2.2 M-Tree with Fuzzy Dominant Color Distance

This version differs from previous one in computing DC distance and also normalization process of CL, DC, EH distances. For evaluating DC distance of two images, we took color similarity into account by applying Single Mode DC Search in [50]. In this search, system firstly evaluates color similarity by calculating Euclidean Distance between two color indexes of first image and second image. If color distance is less than a threshold value, which is '5' in this work, then color similarity is calculated by extracting color distance from 1. After calculating color similarity, the system evaluates DC similarity by selecting minimum dominant color's percentage of two dominant colors' percentages and normalizes this minimum value. Then color similarity is multiplied with this DC similarity to find final DC similarity of two images. And finally DC distance of two images is calculated by extracting final DC similarity from 1.

For all colors of first and second images, this process is applied and minimum DC distance is selected as overall DC distance of two images.

For CL and EH, distance calculation is similar to other version of M-Tree. Only difference comes from normalization of these distances. For normalization, the

system calculates possible maximum CL and EH distances and set these values to 1. Then CL and EH distances of two images are divided into these maximum CL and EH distances to get normalized values.

#### 4.4.2.3 Using OWA

To use OWA operator, the system finds a maximum value and a minimum value of CL, DC and EH distances. These distances are normalized to be in [0,1]. From the definition of OWA aggregation method [20], overall distance is in [0,1], too.

Suppose that $(d_1,d_2,..,d_n)$ are n distance values and order these numbers increasingly: $d_1 \leq d_2 \leq \ldots \leq d_n$ . The OWA operator associated to the n nonnegative weights $(w_1,\ldots,w_n)$ with

$$\sum_{i=1}^{n} w_i \quad \text{where } w_i \in [0,1] \text{ and } w_n \leq \ldots \leq w_2 \leq w_1 \qquad (4.10)$$

corresponds to

$$F(d_1,...,d_n) = \sum_{i=1}^{n} w_i \cdot d_i \qquad (4.11)$$

It should be noted that the weight $w_n$ is linked to the greatest value, $d_n$ and $w_1$ is linked to the lowest value $d_1$ to emphasize similarity between two objects.

If $(d_1,d_2,..,d_n)$ are metric distances, then

$$D_1 = (d_1 + d_2 + ..+ d_n) \qquad (4.12)$$

$$D_2 = \max(d_1, d_2, ..,d_n) \qquad (4.13)$$

$$D_3 = \min(d_1, d_2, ..,d_n) \qquad (4.14)$$

are also metrics [40]. So $F(d_1, d_2, .., d_n)$ is also metric if $(d_1, d_2, .., d_n)$ are metrics. From the definition of OWA aggregation method [20], since

$$F_{max}(d_1, d_2, .., d_n) = \max(d_1, d_2, .., d_n), \text{ where } w_n = 1 \text{ and } w_i = 0 \text{ for } i < n$$
$$F_{min}(d_1, d_2, .., d_n) = \min(d_1, d_2, .., d_n), \text{ where } w_1 = 1 \text{ and } w_i = 0 \text{ for } i > 1$$

and

$$F_{min}(d_1, d_2, .., d_n) \leq F(d_1, d_2, .., d_n) \leq F_{max}(d_1, d_2, .., d_n) ,$$

then $F(d_1, d_2, .., d_n)$ is also metric.

For example, for to objects $O_1$ and $O_2$, we want to calculate distance between these objects, let's say $d(O_1, O_2)$, and assume that, for each feature, CL, DC and EH, normalized Euclidean distance values are;

$$d_{CL}(O_1, O_2) = 0.325$$

$$d_{DC}(O_1, O_2) = 0.570$$

$$d_{EH}(O_1, O_2) = 0.450$$

and OWA weights are;

$w_1 = 0.7$

$w_2 = 0.2$

$w_3 = 0.1$

that is

$$w_1 + w_2 + w_3 = 0.7 + 0.2 + 0.1 = 1 ,$$

then overall distance is:

$$d(O_1, O_2) = F(d_{CL}(O_1, O_2), d_{DC}(O_1, O_2), d_{EH}(O_1, O_2))$$

$$= w_1 * d_{DC}(O_1, O_2) + w_2 * d_{EH}(O_1, O_2) + w_3 * d_{CL}(O_1, O_2)$$

$$= 0.7 * 0.570 + 0.2 * 0.450 + 0.1 * 0.325$$

$$= 0.522$$

## 4.5 Indexing and Querying

In tradition database, indices are based on text, character string or number. Indices in multimedia database, however, are not restricted in text based, but also possible be icon records. So an ideal CBIR system should be scalable to large image collections and should support fast retrieval. For this purpose multi-dimensional indexing is used. For an efficient similarity search in a typical CBIR system it is necessary to store the feature vectors in a multi-dimensional index structure and use the index structure to efficiently evaluate the distance metric. Moreover multi-dimensional index structure must efficiently support both range and nearest neighbor queries.

For indexing multimedia data we have used M-Tree known as a dynamic and balanced access structure suitable to index generic metric spaces. With this structure, indexed objects must belong to a metric space.

The similarity between the objects in M-Tree index structure is calculated by a distance function satisfying the properties of *symmetry*, *positivity* and *triangle inequality* for any triple of objects.

### 4.5.1 M-Tree

The M-tree is a dynamic paged structure that can be efficiently used to index multimedia databases, where the object is represented by means of complex features and the object proximity is defined by a distance function satisfying the positivity, symmetry, and triangle inequality postulates. Similarity queries of the objects require the computation of time-consuming distance functions. Previously, the M-tree indexing structure and the algorithms of inserting, querying and bulk loading have been reported [15] [43]. And it turns out that M-tree is an excellent indexing technique for the query of multimedia database.

M-tree organizes objects in an arbitrary metric space, which is defined in Section 2.1.4. Examples of distance functions that can be used in M-tree are Minkowski-form distances (Euclidean distance, Manhattan distance etc). Since metric spaces strictly include vector spaces, M-tree has a far more general applicability than spatial access methods, such as the R-tree [17]. An example view of the tree structure is shown in Figure 4.5 and Figure 4.6.

The concept of M-tree relies on metric tree that partitions a given search space by considering relative distances between objects, and such partitioning algorithm is critical to the effectiveness of the tree. A major differentiation of M-tree from other metric trees is that the design has to give efficient secondary storage organization [15]:

- *Paged*: tree is paged (consisting of fixed-size or variable-size nodes)
- *Balanced*: paths from the root to leaves all have the same length
- *Dynamic*: able to deal with insertions and/or deletions without degrading search performance and storage utilization, and avoiding global tree reorganization, like Spatial Access Methods.

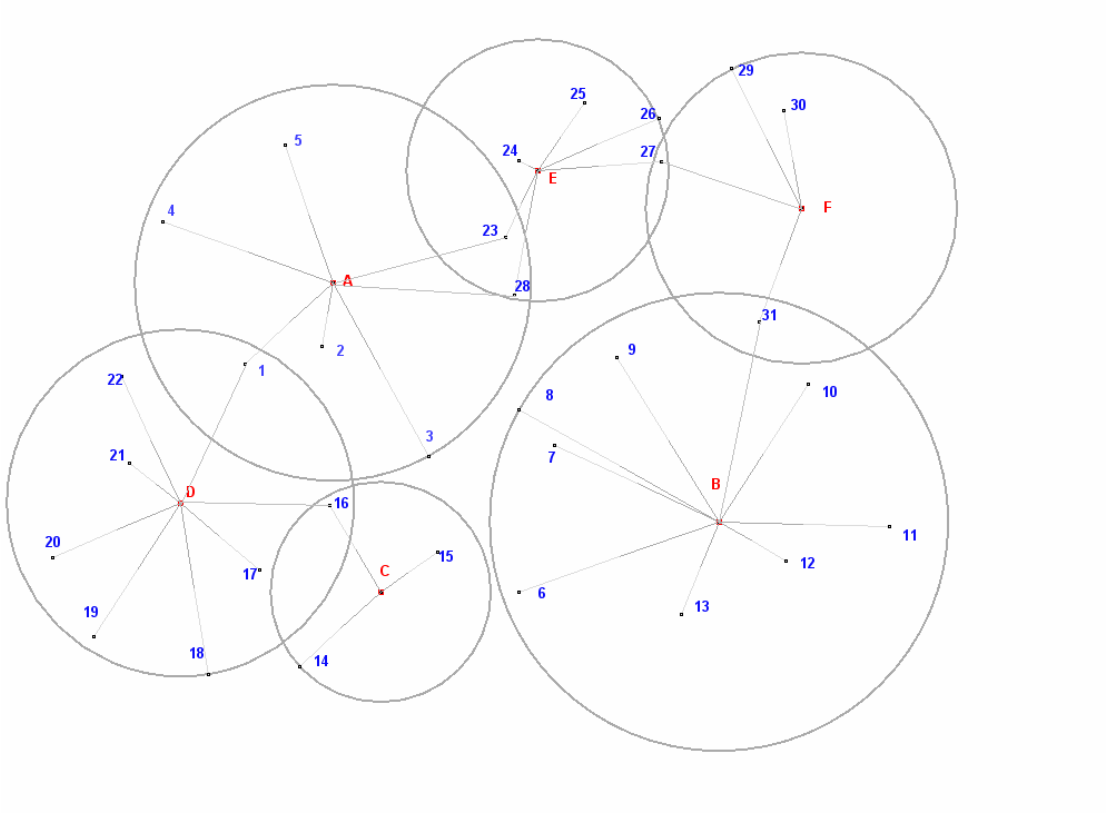


Figure 4.5: Example distribution of data and covering regions.

M-tree has two types of node structures: leaf node and internal node. Leaf node stores a *ground object* and an internal node that stores a *routing object*. Database objects are recursively organized by considering their distances from reference or *routing objects*. And these *routing objects* are also database objects, which acquire their routing roles according a specific promotion algorithm.

The general information for a routing object entry is shown in Figure 4.7 and includes [15]:

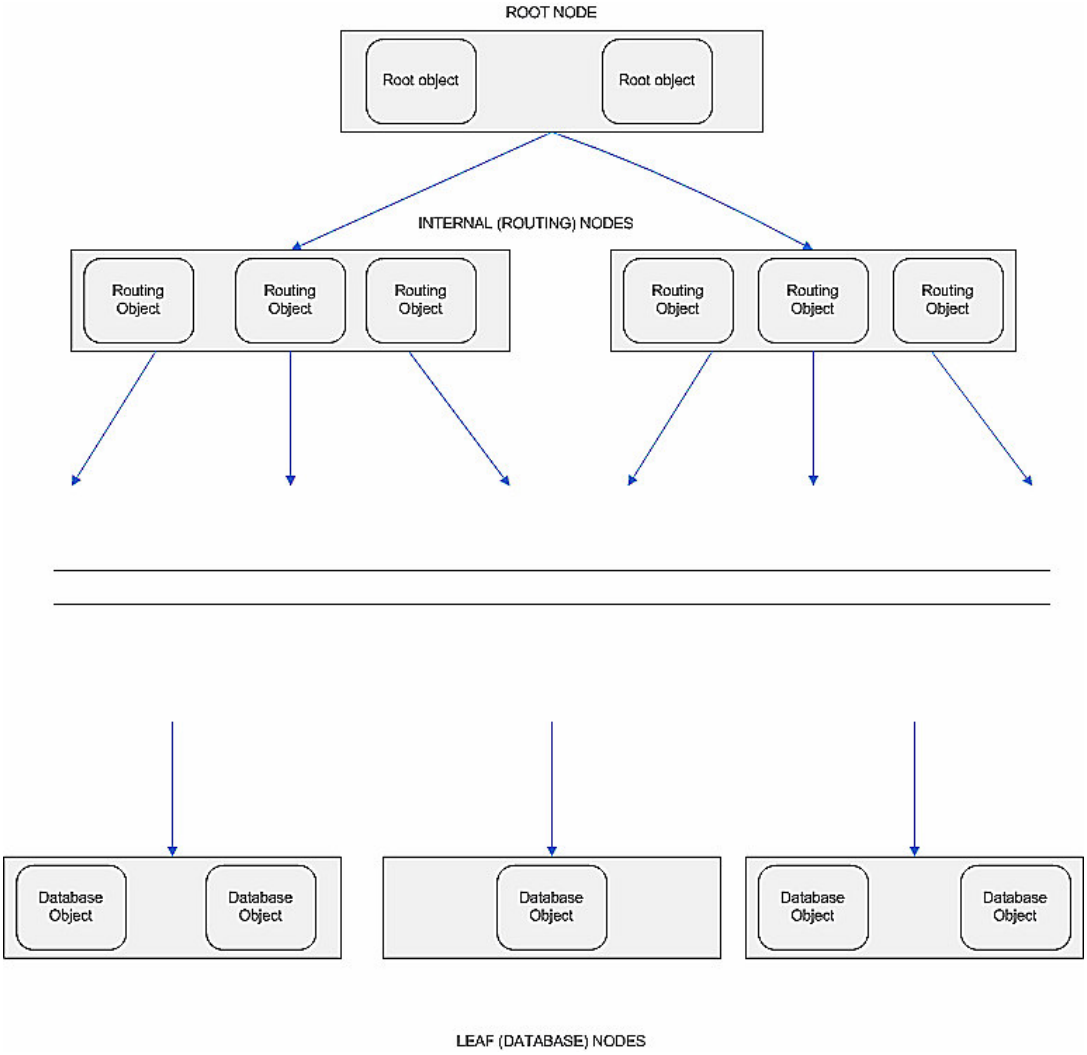


Figure 4.6: M-Tree overview

- *Or*: (feature value of the) routing object
- *ptr(T(Or))*: pointer to the root of T(*Or*), where T(*Or*) is a sub-tree
- *r(Or)*: covering radius of *Or*
- *d(Or, P(Or))*: distance of *Or* from its parent, where *P(Or)* is parent of routing object

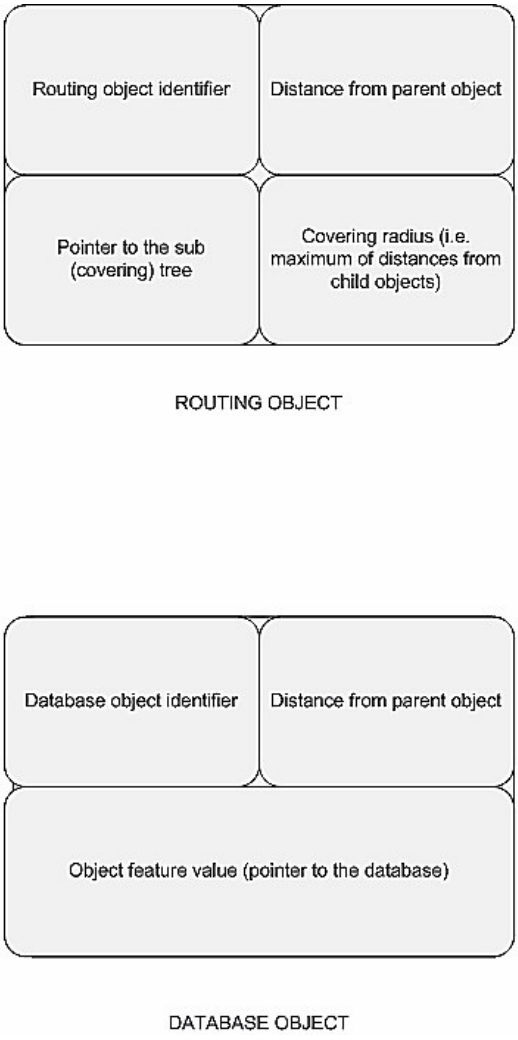


Figure 4.7: Routing and Database objects of M-Tree.

And the leaf nodes (database object entry) are shown in Figure 4.7 and contain:

- $O_j$: (feature value of the) database object
- $oid(O_j)$: object identifier
- $d(O_j, P(O_j))$: distance of $O_j$ from its parent

Routing object $O_j$ is used to access to a sub-tree, $T(O_j)$, through a root pointer, $ptr(T(O_j))$,where $T(O_j)$ is the covering tree of $O_j$. $T(O_j)$ consists of the union of $\{O_j\}$ and the set of objects in $T(O_j)$. A covering tree has the property that all objects in the covering tree of $O_j$ are within the distance $r(O_j)$ from $O_j$, $r(O_j) > 0$, which is called the *covering radius* of $O_j$. Hence the covering radius of $O_j$, $r(O_j)$ is defined as:

$$r(O_j) \geq \max\{d(O_j, O_i) \mid O_i \in T(O_j)\} \quad (4.15)$$

And the covering region $R(O_j)$:

$$R(O_j) = \{ O_i \in T(O_j) \mid d(O_j, O_i) \leq r(O_j)\} \quad (4.16)$$

The basic M-tree operations include querying, insertion, deletion and tree construction (bulk loading), and details can be found in M-tree specifications [15][43]. In next section, querying the M-Tree is briefly explained.

#### 4.5.1.1 Querying the M-Tree

For a given specific metric defined by its distance, M-tree is able to support processing of two main types of queries: *range queries;* finding all objects that are within a specific distance from a given object and *nearest Neighbor Query* (*k*-NN); finding a specific number, *k*, of closest objects to a given query object. These queries are defined as follows:

Range Query: Given a query object $Q \in D$, where $D$ is domain of feature values, and for a distance (range) *r(Q),* the range query range*(Q, r(Q))* selects all indexed objects $O_j$ such that

$$d(O_j, Q) \leq r(Q) \quad (4.17)$$

For example, a range query becomes:

"*Find all images which have a distance value less than 0.2 from query image*"

k-nearest Neighbors Query (k-NN): Given a query object $Q \in D$ and an integer $k \geq 1$, the k-NN query *NN(Q, k)* selects the k indexed objects which have the shortest distance from *Q*. For example, a k-NN query becomes:

"*Find 10 nearest images to query image*"

For querying the tree, triangle inequality is used to prune some nodes (i.e., sub-tree) from the search, thus reduce the distance computations. Triangle inequality is used as follows:

Suppose that we are looking for the closest point to *Q*, as in Figure 4.8 in a database of 3 objects. Further suppose that the triangular inequality holds, and that we have pre-compiled distances between all the items in the database. Such that

d*(a, b) = 6.70*
*d(a, c) = 7.07*
*d(b, c) = 2.30*

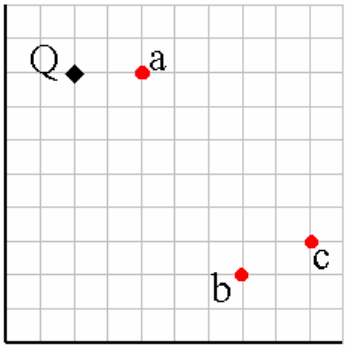


Figure 4.8: A Sample data for querying

And, we find *a* and calculate that it is 2 units from *Q*, it becomes our *best-so-far*. we find *b* and calculate that it is 7.81 units away from *Q*. Now we don't have to calculate the distance from *Q* to *c***,** because of triangle inequality, so that:

$$d(Q,b) \le d(Q,c) + d(b,c)$$
$$= d(Q,b) - d(b,c) \le d(Q,c)$$
$$= 7.81 - 2.30 \le d(Q,\ c)$$
$$= 5.51 \le d(Q,\ c)$$

The distance between *Q* and *c* is at least 5.51 units, but our *best-so-far* is only 2 units away.

An example range query is explained below. For this example, an M-Tree is shown in Figure 4.10.

Suppose that an image is given to the system for selecting the images which have a distance from query image less or equal than 0.2 (r = 0.2). Query image's CL, DC and EH feature values are shown below;

CL Descriptor:

```
<Image name = "97.jpg">
    <Descriptor xsi:type = "ColorLayoutType">
        <YDCCoeff>10</YDCCoeff>
        <CbDCCoeff>19</CbDCCoeff>
        <CrDCCoeff>30</CrDCCoeff>
        <YACCoeff5>14  10  16  16  10 </YACCoeff5>
        <CbACCoeff2>15  21 </CbACCoeff2>
        <CrACCoeff2>14  17 </CrACCoeff2>
    </Descriptor>
```

*</Image>*

DC Descriptor:

```
<Image name = "97.jpg">
        <Descriptor size = "7" xsi:type = "DominantColorType">
                <SpatialCoherency>0</SpatialCoherency>
                <Values>
                        <Percentage>9</Percentage>
                        <ColorValueIndex>2  2  1 </ColorValueIndex>
                </Values>
                <Values>
                        <Percentage>5</Percentage>
                        <ColorValueIndex>8  11  5 </ColorValueIndex>
                </Values>
                <Values>
                        <Percentage>2</Percentage>
                        <ColorValueIndex>12  9  8 </ColorValueIndex>
                </Values>
                <Values>
                        <Percentage>1</Percentage>
                        <ColorValueIndex>27  25  24 </ColorValueIndex>
                </Values>
                <Values>
                        <Percentage>5</Percentage>
                        <ColorValueIndex>5  6  3 </ColorValueIndex>
                </Values>
                <Values>
                        <Percentage>5</Percentage>
                        <ColorValueIndex>11  15  6 </ColorValueIndex>
                </Values>
```

<Values>
<Percentage>1</Percentage>
<ColorValueIndex>20 12 10 </ColorValueIndex>
</Values>
</Descriptor>
</Image>

EH Descriptor:

<Image name = "97.jpg">
<Descriptor xsi:type = "EdgeHistogramType">
<BinCounts>
2 4 5 4 6 1 3 5 3 7 3 2 5 4 7 3 2 5 3 7 1 5 5
5 3 2 4 4 6 5 3 4 4 5 4 2 3 6 5 5 2 6 5 3 4 2
3 4 7 6 2 3 4 5 7 2 2 5 3 4 1 4 6 5 6 2 3 5 5
7 1 2 5 4 7 2 4 4 6 6
</BinCounts>
</Descriptor>
</Image>

And query image is shown in Figure 4.9.

Figure 4.9: Query Image

So the system firstly calculates the distance between query image, Q, and root node entries of the M-Tree. Suppose that the distances are:

d(Q, A) = 0.360
d(Q, B) = 0.455
d(Q, C) = 0.045

For the sub-tree of A, the system decides whether this sub-tree will be pruned or not. This is done as:

For first child of A, which is A1 and equal to the A (A1 and A are the same objects), triangle inequality is used to prune (or not to prune) the sub-tree of A1.

If

$$\left|d(Q,A) - d(A,A_1)\right| > r(Q) + r(A_1)| \tag{4.18}$$

then we can prune the sub-tree of A1 (from M-Tree paper). From the M-Tree, we know the distance between an entry and its child entry,

d(A, A1) = 0  (Since A and A1 are the same objects)

Then

$$\left|d(Q,A) - d(A,A_1)\right| > r(Q) + r(A_1)$$

|0.360- 0| > 0.2 + 0.152

0.360 > 0.352

so we can prune the sub-tree A1.

For second entry of sub-tree of A, that is A2, the system prunes or doesn't prune the sub-tree by using triangle inequality. But this time r(A2) is equal to zero since A2 is also leaf node (a3 and A2 are the same objects). So, if

$$\left|d(Q,A) - d(A,A_2)\right| > r(Q) \qquad (4.19)$$

then we can prune the sub-tree of A2 (we don't need to calculate the distance between Q and A2, d(Q, A2)).

Then

$$\left|d(Q,A) - d(A,A_2)\right| > r(Q)$$

$$|0.360 - 0.229| > 0.2$$

$$0.131 < 0.2$$

so we have to calculate the value of d(Q,A2), which is equal to 0.333. But, since

$$d(Q,A2) > r(Q)$$

$$0.333 > 0.2$$

A2 is not included in query results.

For the root entry B and its child entries B1, B2 and B3, the same steps are applied and the sub-tree B1 and B2 is pruned while the leaf nodes (b4 and b5) of the sub-tree B3 should be evaluated. So d(Q,b4) and d(Q,b5) are calculated.

$$d(Q,b4) = 0.231$$

$$d(Q,b5) = 0.312$$

Since these values are greater than desired range (r(Q)), b4 and b5 (image 7 and image 8 respectively) is ignored.

For the C entry of root node, the same steps are applied. And the distances of all leaf nodes in the sub-tree of C are calculated. These leaf nodes are also in the expected results of the query Q. The distance values are;

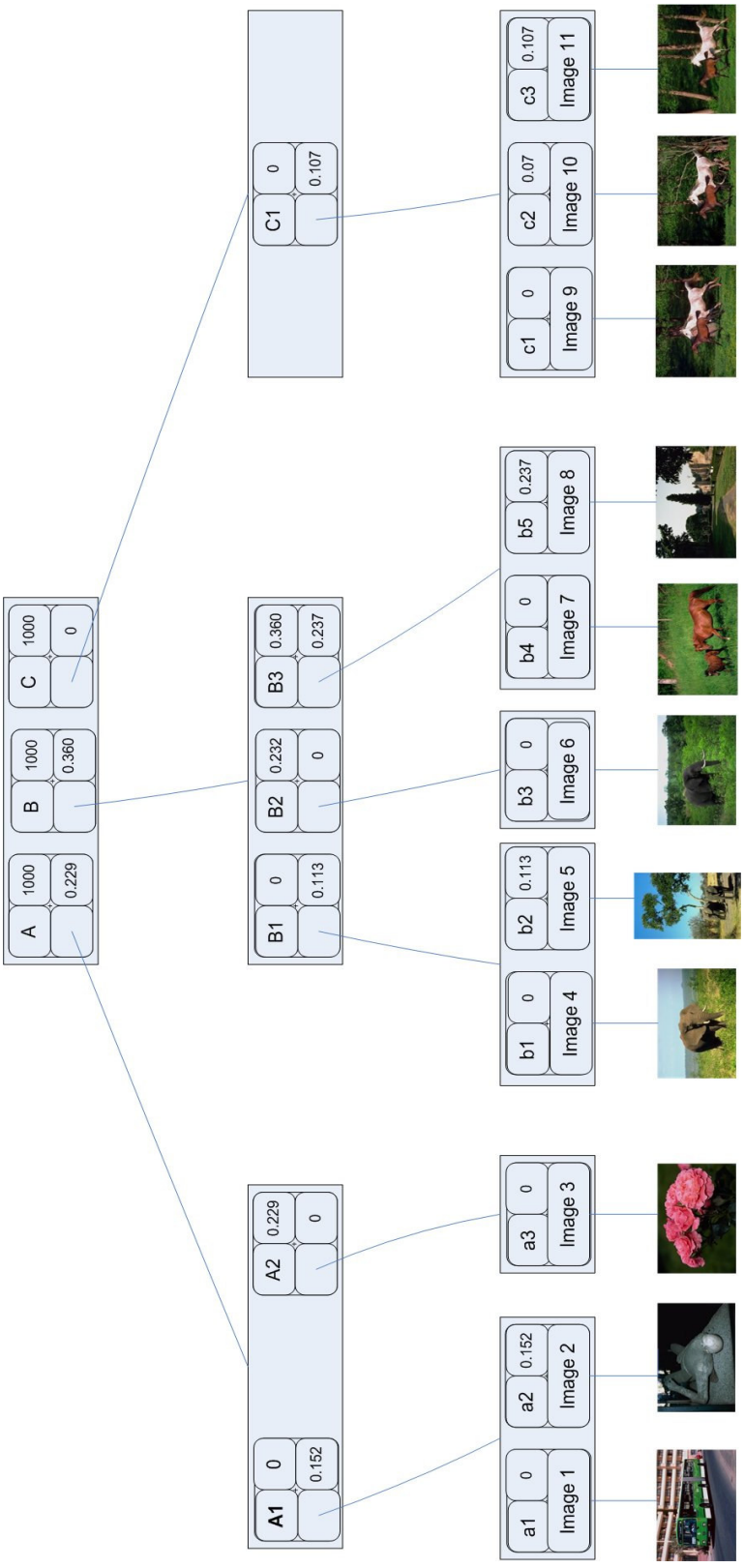


Figure 4.10: Example M-Tree for 11 images

d(Q,c1) = 0.045

d(Q,c2) = 0.072

d(Q,c3) = 0.110

and c1, c2 and c3 are added to the query result set.

### 4.5.2 Fuzzy Query

To support fuzzy queries, we developed Web-based user interface with JSP/Servlet technologies. There are three types of fuzzy queries;

- Image-based
- Feature-based.
- Color-based

*Image–Based Fuzzy Query:* If whole image query is selected, the user has to select similarity degree for query image which is consists of *'Almost Same', 'Very Similar', 'Similar'* and *'Not Similar'*. Then the system maps this similarity degree into a distance range and searches the tree to retrieve result images, which have a distance to query image in that range. And finally results are shown to the user with their distance value to the query image. The general syntax of this type of query is as follows:

QUERY={{<Similarity>} }

where

Similarity = {<Almost Same>|<Very Similar>|<Similar >|<Not Similar >|}

For an example, suppose that user gives the following similarity values for the features; '*Very Similar to Query Image'*. Then our query is defined as:

QUERY= {Very Similar to Query Image}

And suppose that these similarity values are mapped into distance ranges as follows:

'*Almost Same':* [1, 0.95)
'*Very Similar'*: [0.95, 0.85)
'*Similar'*: [0.85, 0.5)
'*Not Similar'*: [0.5, 0.0].

So final distance range is negotiation of this similarity range, which is

=(0.05, 0.15]

Finally the system retrieves the images, which have a distance value from query image in that range.

*Feature–Based Fuzzy Query:* Another type of query is feature-based fuzzy query. In this type, the user must supply similarity values for all three features DC, CL and EH. These similarity values are the same of the ones in image-based fuzzy query. For combining these similarities AND/OR operators must be given. Then the system applies some conjunction/ disjunction procedures to get final similarity values and maps these values into distance range. These conjunction/disjunction procedures are explained in [49].

Conjunction rule: $\mu_{A \wedge B} = \min\{\mu_A(x), \mu_B(x)\}$ (4.20)

Disjunction rule: $\mu_{A \vee B} = \max\{\mu_A(x), \mu_B(x)\}$ (4.21)

If AND operator is supplied to combine feature similarities, the system uses disjunction rule, and if OR operator is supplied to combine feature similarities, the system uses conjunction rule. The general syntax of this type of query is as follows:

QUERY={{<Similarity><Feature >}<&>{<QUERY>}}

where

Similarity = {<Almost Same>|<Very Similar>|<Similar >|<Not Similar >|}

Feature = {<CL>|<DC >|<EH >}

For an example, suppose that user gives the following similarity values for the features;

'*Very Similar'* for CL feature, '*Similar'* for DC feature, '*Almost Same'* for EH feature. Then our query is defined as:

QUERY= {Very Similar in CL OR Similar in DC AND Almost Same as EH}

And suppose that these similarity values are mapped into distance ranges as follows:

'*Almost Same':* [1, 0.95)
'*Very Similar'*: [0.95, 0.85)
'*Similar'*: [0.85, 0.5)
'*Not Similar'*: [0.5, 0.0].

Then our query is like:

*'Very Similar in CL OR Similar in DC AND Almost Same as EH'*

To get final similarity, the system combines these feature similarities as follows:

Firstly AND operator between DC and EH feature is taken into account, so query becomes:

*'Very Similar in CL OR (Similar in DC AND Almost Same as EH)'.*

*(Similar DC AND Almost Same as EH)* part of the query is mapped into similarity ranges and conjunction rule is applied to this part. So range value is equal to;

min([0.85, 0.5) , [1, 0.95))
= [0.85, 0.5)

Then system comes to combine CL feature similarity with this value by applying disjunction rule that is;

max ([0.95, 0.85) , [0.85, 0.5))
=[0.95, 0.85).

And final distance range is negotiation of this similarity range, which is

=(0.05, 0.15]

Finally the system retrieves the images, which have a distance value from query image in that range.

*Color–Based Fuzzy Query:* Color-Based Fuzzy Query differs from other fuzzy queries in query paradigm. This type of query is not an example image based query so user has to supply degree of three colors' percentages in expected images. By this type of query, the system has the facility of asking for a query in terms of the color content of the image [51]. To support this query type, system gives opportunity of defining amount of main colors in the image. To do this, the user must supply each color's percentage in terms of natural language like *'mostly', 'many', 'normally',*

*'few', 'very few'* so the user can able to pose a composite query in terms of colors. The general syntax of this type of query is as follows:

QUERY={{<Content><Color>}<&>{<QUERY>|<>}}

where

Content = {<mostly>|<many>|<normally>|<few>|<very few>}

Color = {<Red>|<Green>|<Blue>}

An example query becomes as follows:

QUERY={many red AND mostly green OR very few blue}.

Mapping function of these linguistic terms into similarity values is defined according to data set. For example, for testing Corel Dataset, we have used the following values:

'*Mostly:* [1, 0.88)

*'Many'*: [0.88, 0.85)

'*Normally'*: [0.85, 0.82)

'*Few'*: [0.82, 0.80).

'*Very Few'*: [0.80, 0.0].

After defining the query, system searches the tree for each color seperately by using predefined query features in DC and CL for pure red,green and blue colors. EH feature is not important since query is a color query, so the distance value for EH feature is set to zero. Then result sets of each color's query are combined into final result set. If AND operator is used in composite query then all objects which are in both result sets are shown to the user with similarity degree. If OR operator is used then all objects of both result sets are shown to the user with similarity degree.

## 4.6 User Interface

Graphical User Interface (GUI) is developed with JSP/Servlet. To run the system, user has to define some parameters. First one of them is dataset that will be used by the system. After selecting the dataset, database is loaded by pressing 'Load DB' button. Then version of the index structure must be selected. Query image is dependent to query type and also dependent to the selected dataset that is user can select query image only from selected dataset.

There are four types of queries, which are mentioned before; k-NN query, image-based fuzzy query, feature-based fuzzy query and color-based fuzzy query. And only one of them at a time can be selected.

If k-NN query is selected then the user has to supply a *'k'* value to see *k* nearest images to the query image. If image-based fuzzy query is selected then image similarity must be given to the system, which can be *'Almost Same', 'Very Similar', 'Similar'* and *'Not Similar'*. Then the system maps this similarity degree into a distance range and searches the tree to retrieve result images, which have a distance to query image in that range. And finally results are shown to the user with their distance value to the query image.

If the user selects feature-based fuzzy query, then each feature similarity must be supplied, which is the same of image-based fuzzy query. For combining feature similarities, user can select AND or OR operators. Then the system applies some conjunction/disjunction procedures to get final similarity values and maps these values into distance range.

If color-based fuzzy query is selected, then each color's percentage must be supplied in linguistic terms like *'mostly', 'many', 'normally', 'few'* and *'very few'*. Then system searches the tree for each color to retrieve results by applying some conjunction/ disjunction rules.

After defining system parameters, system runs by pressing ‘Run’ button. And results are shown to the user with rank and distance value from query image. The user can search the system by returning to the index page of the GUI.

Running examples of the system are shown in Appendix-A.

# CHAPTER 5

# PERFORMANCE EXPERIMENTS

To test the performance of our content-based image retrieval system, we have used images from Corel Database [39] and random images from web. For creating the index structure, we have made tests over image database that contains 100, 200, 300 and 400 Corel images to evaluate number of distance computation and construction time of the index structure. Our system also had been tested by k-NN query paradigm. With these tests, number of distance computation and query cost time had been examined. Also retrieval efficiency of the system had been evaluated by using Average Normalized Modified Retrieval Rank (ANMRR) metric [47].

## 5.1 Building the M-Tree

As any other dynamic balanced tree, M-tree grows in a bottom-up fashion [15]. Building a M-tree can repeatedly insert objects into null tree using insertion method or for better performance, bulk-loading techniques are also proposed [43]. The algorithm involves partitioning the set of objects by sampling and repeats the same partitioning from the leaf level up, which will eventually gives a non-balanced tree. Then refinement steps are invoked which reassigns objects in under-filled sets to other sets, and split “taller” (in terms of path length from root) sub-trees to obtain a number of “shorter” sub-trees.

### 5.1.1 Split Policies

#### 5.1.1.1 Choosing Routing Objects

The partition strategy is crucial to the tree performance. The "ideal" split policy should find the "most suitable" routing object, $O_{p1}$ and $O_{p2}$, and partition the objects such that the "volume" and "overlap" are minimized [15]. The possible strategies of selecting routing object are shown in Table 5.1.

Table 5.1: Possible strategies for selecting routing objects

| | | |
|---|---|---|
| `m_RAD` | "minimum" (sum of) RADii: consider all possible pairs of objects and promote the pair of objects which minimize the sum of covering radii | most complex (distance computation), but gives good tree structure |
| `mM_RAD` | similar to `m_RAD` but the maximum of the two radii is minimized | |
| `M_LB_DIST` | "Maximum Lower Bound on DISTance": uses pre-computed stored distance; fix *Op1* = *Op* and determines *Op2* as the farthest object from *Op* | a relatively "cheap" policy (in terms of distance computation) |
| `RANDOM` | randomly pick the reference object | not a "smart" strategy, but fast tree construction |
| `SAMPLING` | variant of `RANDOM` but iterated over a sample of objects for which the resulting maximum of the two covering radii is minimum | |

#### 5.1.1.2 Distribution of Entries

When the promoting objects are found, the entries are distributed into two sets, $N_1$ and $N_2$. Two suggested strategies are shown in Table 5.2.

Table 5.2: Possible strategies of distribution of entries.

| | |
|---|---|
| Generalized Hyperplane | assign each object to the nearest routing object |
| Balanced | compute $d(O_j, O_{p1})$ and $d(O_j, O_{p2})$ for all $O_j$ and repeat until N is empty:<br>▪ assign to $N_1$ the nearest neighbor of $O_{p1}$ and remove from N;<br>▪ assign to $N_2$ the nearest neighbor of $O_{p2}$ and remove from N; |

### 5.1.2 Evaluating Effectiveness of Building the M-Tree

We have used two different approaches for evaluating similarity between images to build the M-Tree. For the first approach, we have used weighted Euclidean distance function with equal weights which sum is equal to one. For the second approach, we have used OWA operators to define the weights in Euclidean distance function.

To evaluate the performance of building the tree two tests with different parameter sets, which are defined in M-Tree Project [42] [46], performed. These parameters are;

- MIN UTIL Minimum node utilization. It is used to guarantee a minimum fill factor for tree nodes during the split. It can assume values in the range [0 , 0.5]

- PROMOTE_PART_FUNCTION: This parameter is used for defining split policy. Specifies the algorithm used to promote objects in the parent role. Assuming the set of following values:

    - RANDOM: Random promotion.

    - CONFIRMED: Confirmed promotion, variable PROMOTE VOTE FUNCTION is then used to choose between confirmed policies.

    - MAX_UB_DIST: Maximum upper bound on distances policy; the two objects having the maximum distance from parent object are chosen.

    - MIN_RAD: Minimum maximum radius policy.

    - MIN_OVERLAPS: Minimum overlap policy

    - SAMPLING: Sampling promotion; variable NUM_CANDIDATES specifies the number of samples.

- PROMOTE_VOTE_FUNCTION: This parameter is meaningful when confirmed PROMOTE_PART_FUNCTION is used for defining split policy and specifies the algorithm used to promote one object as one of the two parents, the other being the parent object of the split node. Assuming the set of following values:

- RANDOMV: Random confirmed promotion.
- SAMPLINGV: Sampling confirmed promotion; NUM_CANDIDATES specifies the number of samples.
- MAX_LB_DIST: Maximum lower bound on distances promotion (i.e., the object farthest from the parent object is chosen).
- mM_RAD: minimum radius confirmed policy, variable RADIUS FUNCTION is then used to choose between available policies.

- RADIUS_FUNCTION: Minimum radius method. Assuming the set of following values:
  - LB: Minimum maximum lower bound on radius policy;
  - AVG: Minimum maximum average bound on radius policy;
  - UB: Minimum maximum upper bound on radius policy.

- SECONDARY PART FUNCTION: Root promotion method. It is only used when splitting the root node and can assume the same values of the PROMOTE PART FUNCTION variable. However, since the root node does not have a parent object, this cannot be a confirmed policy.

- NUM_CANDIDATES: Number of candidate objects for sampling methods.

- SPLIT_FUNCTION: This specifies the way objects in the overfull node are to be divided between the two new nodes. Assuming the set of following values:

- G_HYPERPL the generalized hyperplane partition strategy;
- BAL_G_HYPERPL the balanced hyperplane partition strategy;
- BALANCED the balanced strategy.

- PAGE_SIZE: The size of disk pages.

#### 5.1.2.1 Confirmed Promotion

For the first test, we used following values for these parameters:

- PROMOTE_PART_FUNCTION: CONFIRMED
- SECONDARY PART FUNCTION: mM_RAD
- RADIUS_FUNCTION: LB
- SPLIT_FUNCTION: G_HYPERPL

We have performed this test for five different minimum utilization values and for five different page sizes. Also four different databases are used in tests and results are shown in figures below. (EWS: Equal Weighted Sum, OWA: Ordered Weighted Aggregation)

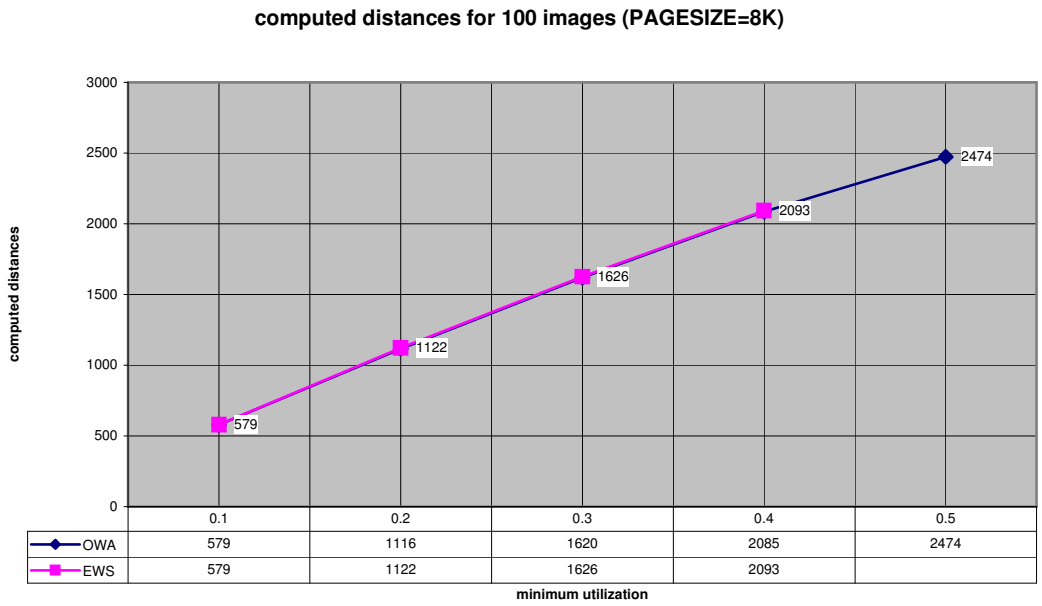


Figure 5.1: Computed Distances for 100 images as a function of minimum utilization with page size = 8K

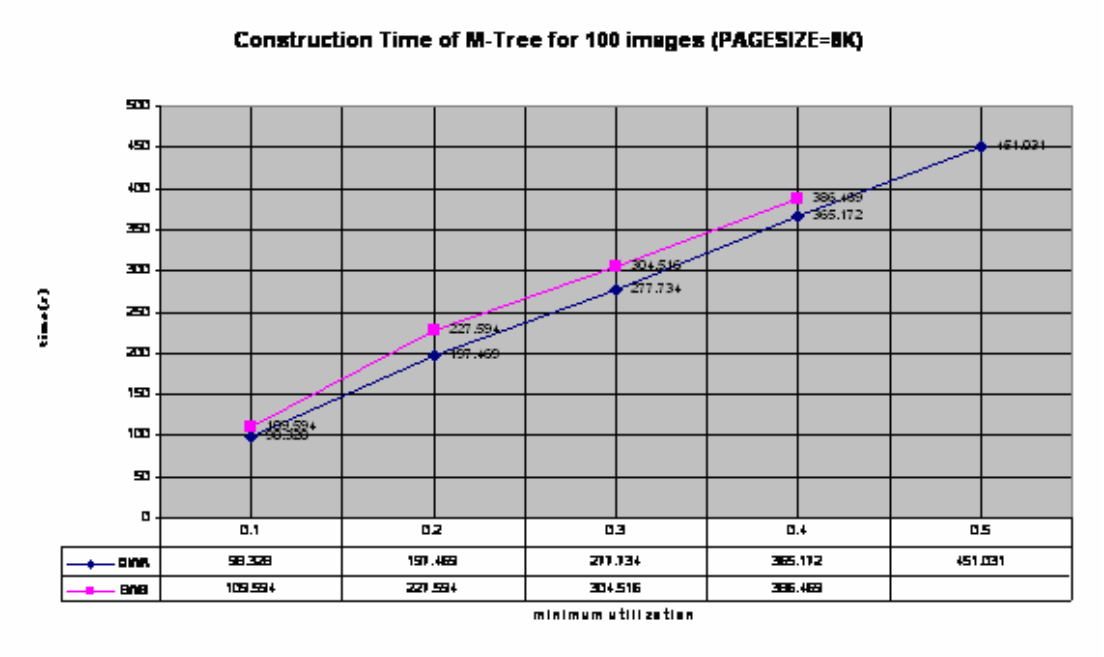


Figure 5.2: Construction Time for 100 images as a function of minimum utilization with page size = 8K

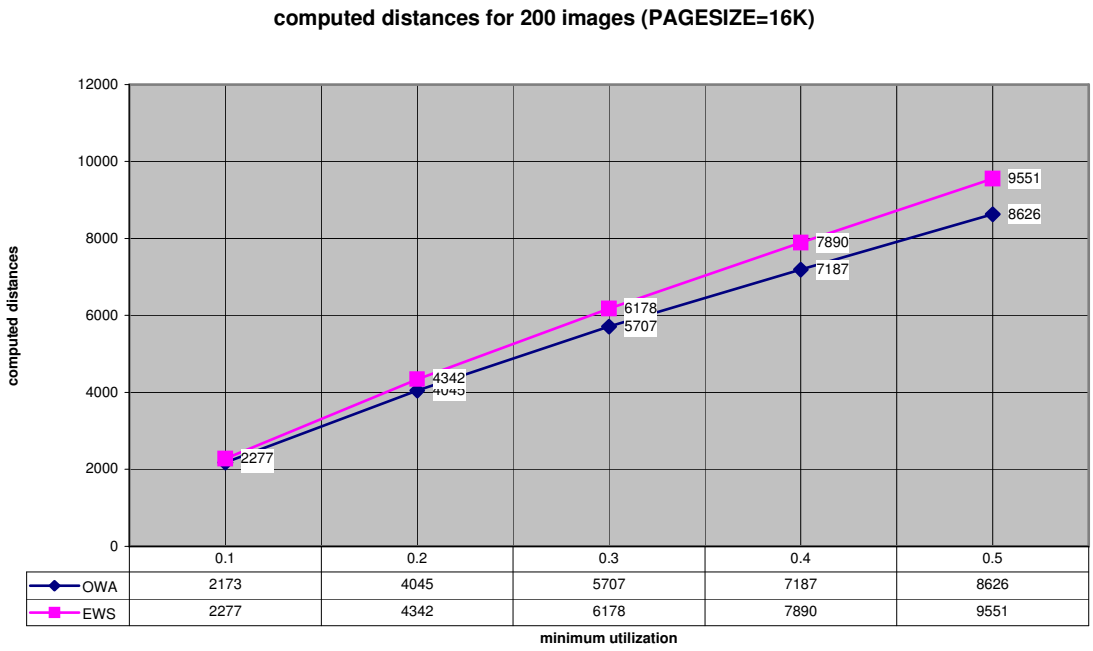


Figure 5.3: Computed Distances for 200 images as a function of minimum utilization with page size = 16K

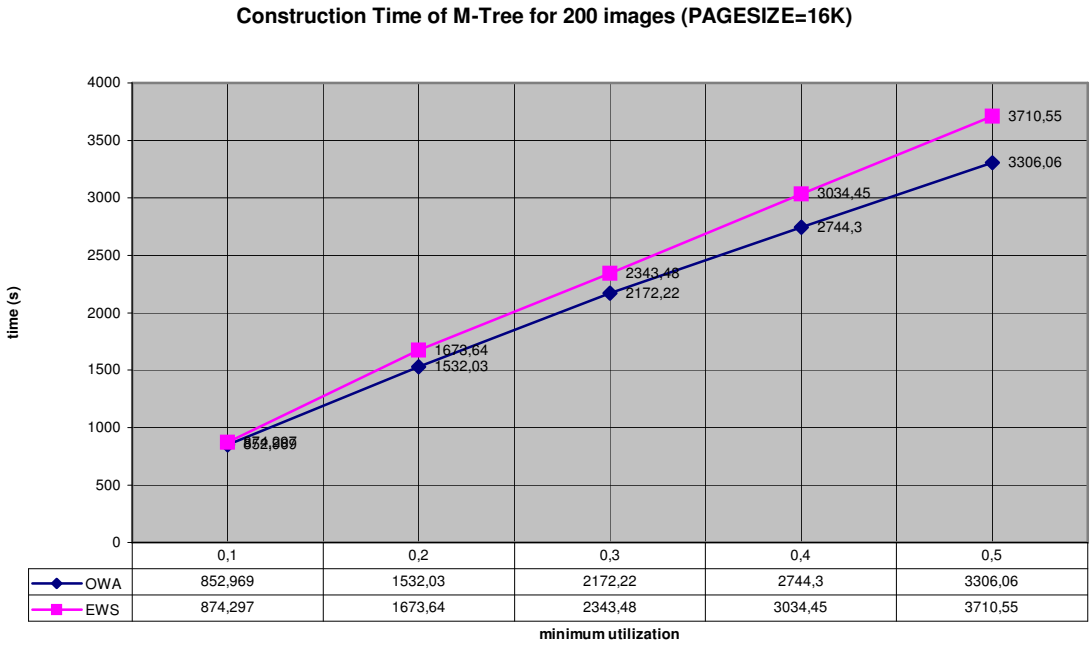


Figure 5.4: Construction Time for 200 images as a function of minimum utilization with page size = 16K

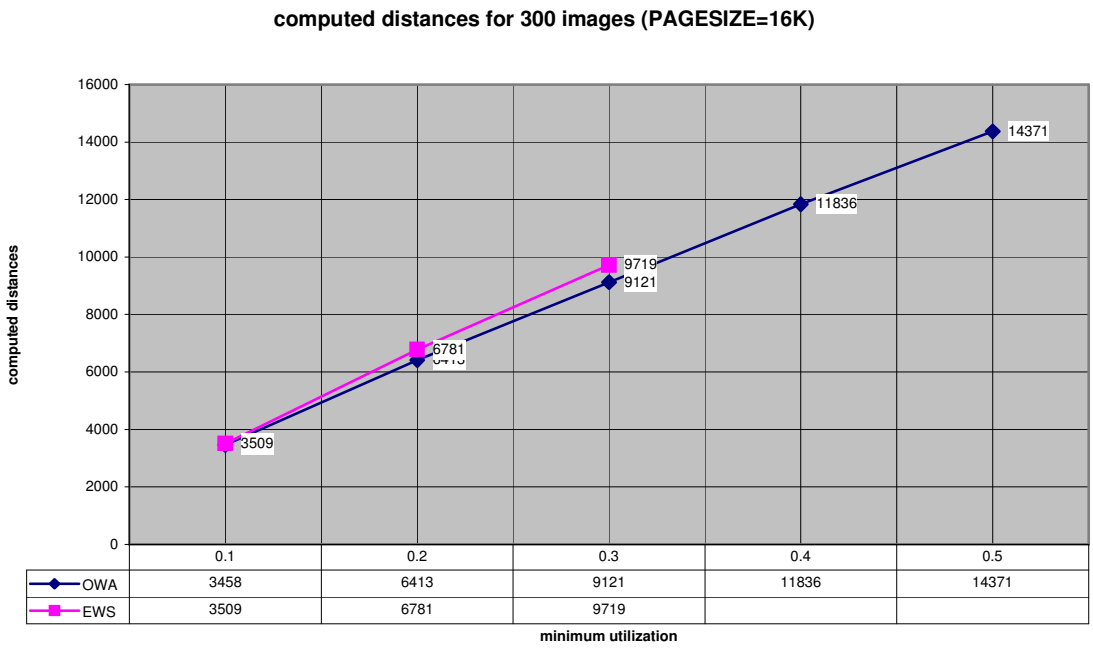


Figure 5.5: Computed Distances for 300 images as a function of minimum utilization with page size = 16K

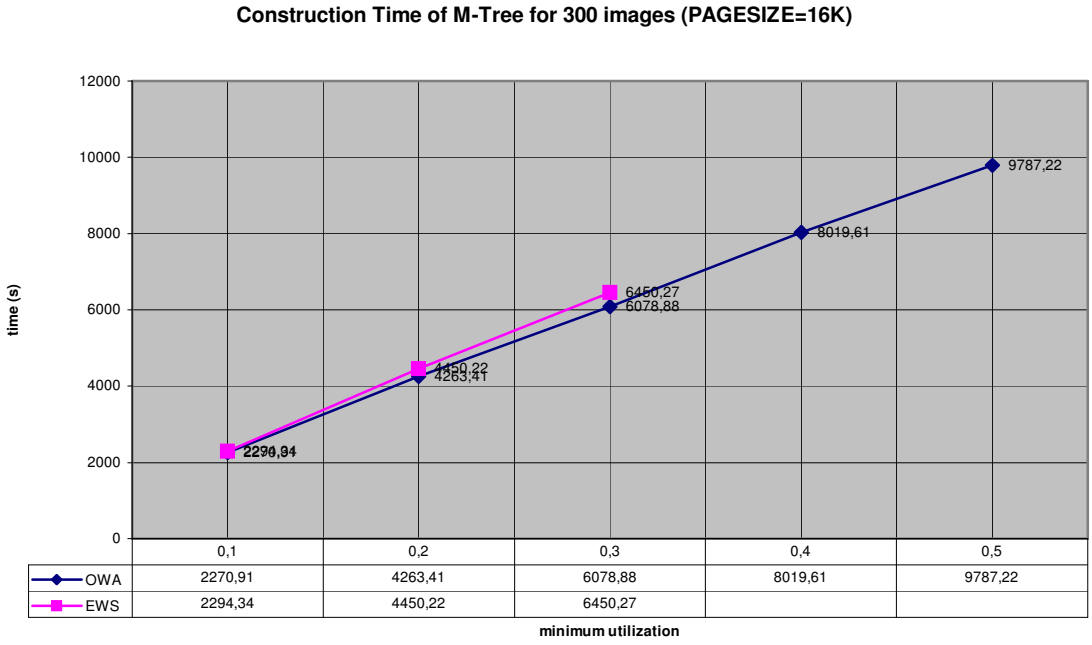


Figure 5.6: Construction Time for 300 images as a function of minimum utilization with page size = 16K

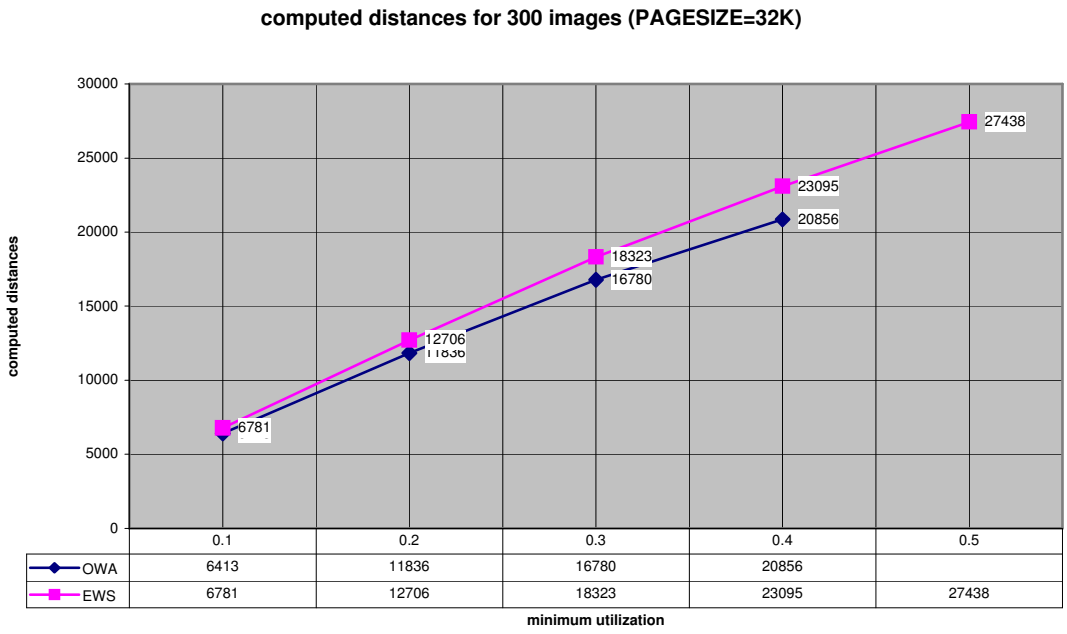


Figure 5.7: Computed Distances for 300 images as a function of minimum utilization with page size = 32K

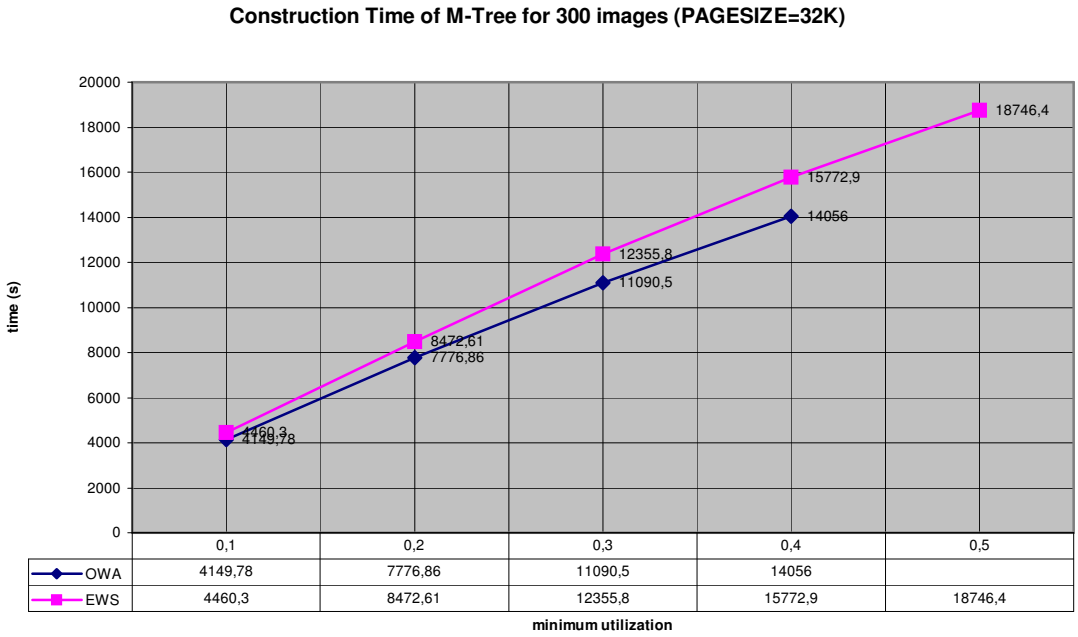


Figure 5.8: Construction Time for 300 images as a function of minimum utilization with page size = 32K

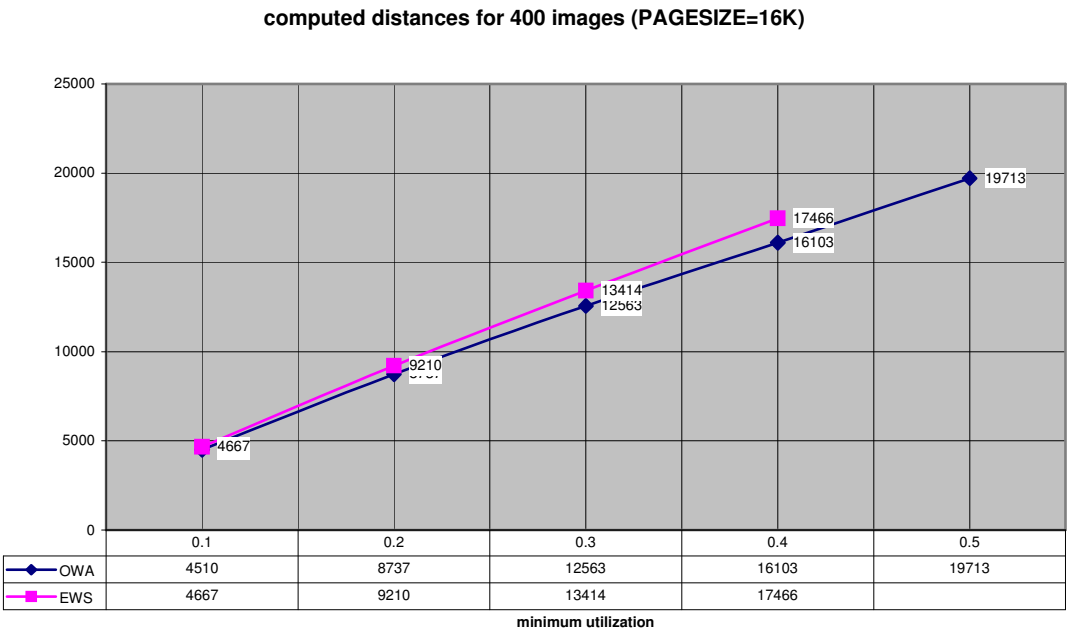


Figure 5.9: Computed Distances for 400 images as a function of minimum utilization with page size = 16K

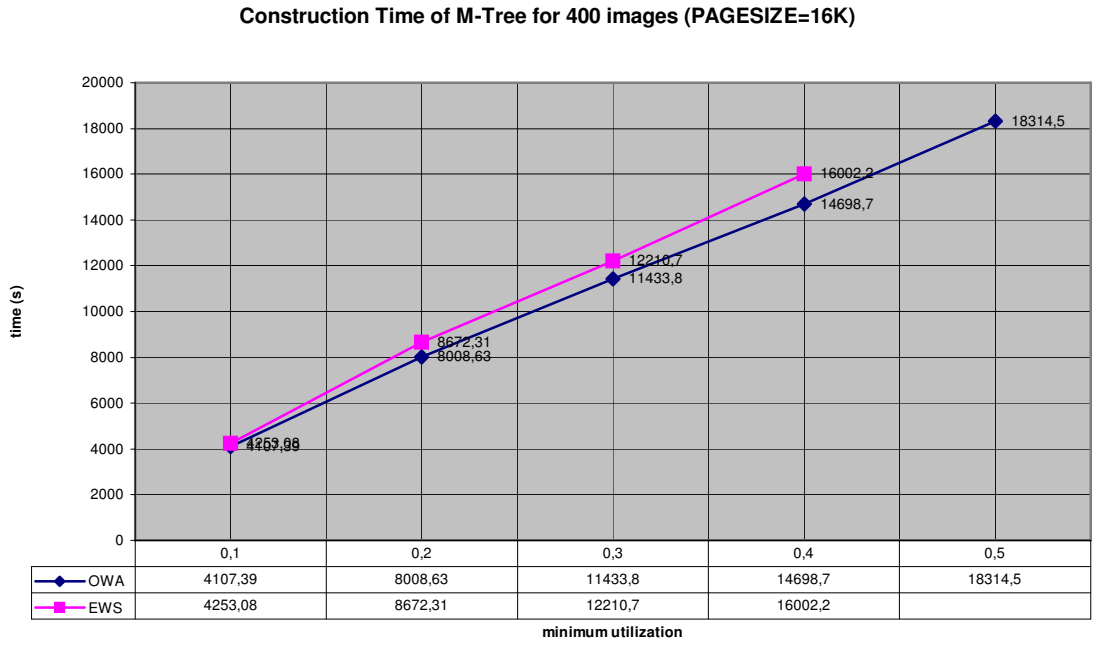


Figure 5.10: Construction Time for 300 images as a function of minimum utilization with page size = 16K

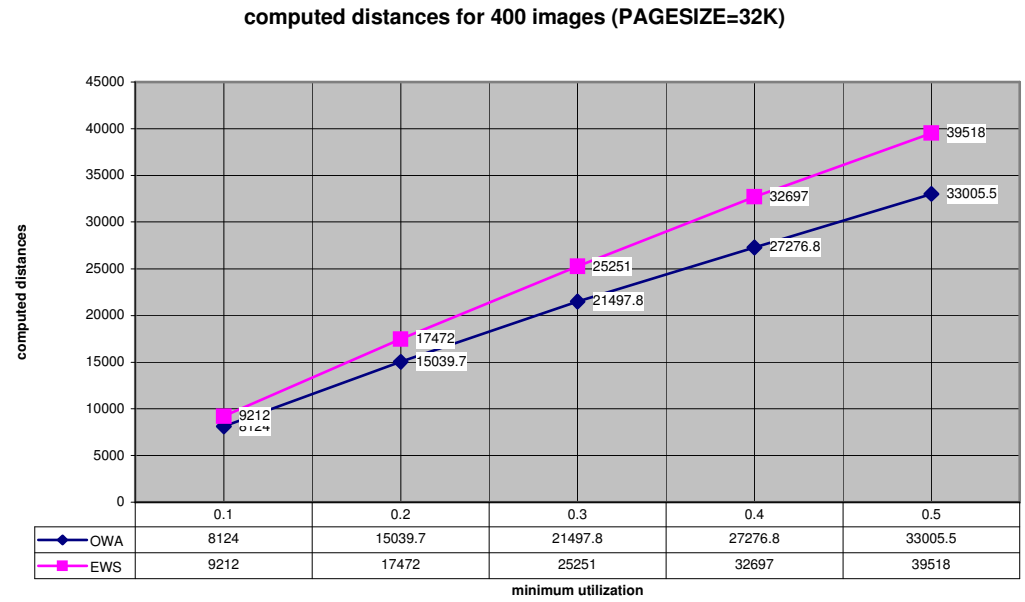


Figure 5.11: Computed Distances for 400 images as a function of minimum utilization with page size = 32K

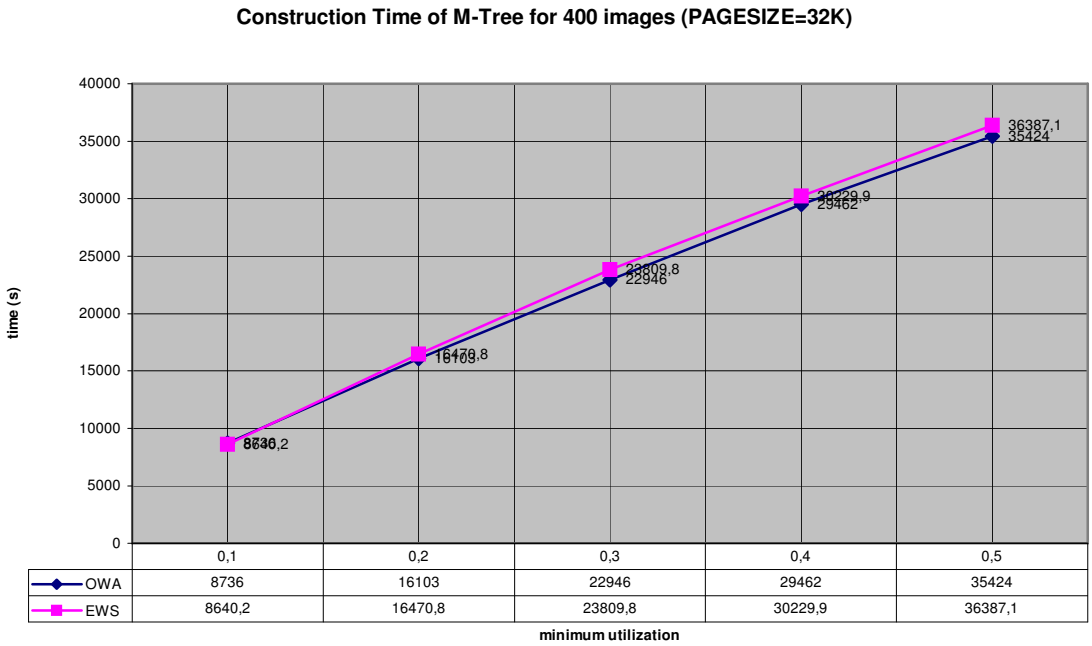


Figure 5.12: Construction Time for 400 images as a function of minimum utilization with page size = 32K

### 5.1.2.2 Random Promotion

For the first test, we used following values for these parameters:

- PROMOTE_PART_FUNCTION: RANDOM
- SECONDARY PART FUNCTION: RANDOM
- SPLIT_FUNCTION: G_HYPERPL

We have performed this test for five different minimum utilization values and for five different page sizes. Also four different databases are used and results are shown in figures below. (EWS: Equal Weighted Sum, OWA: Ordered Weighted Aggregation)

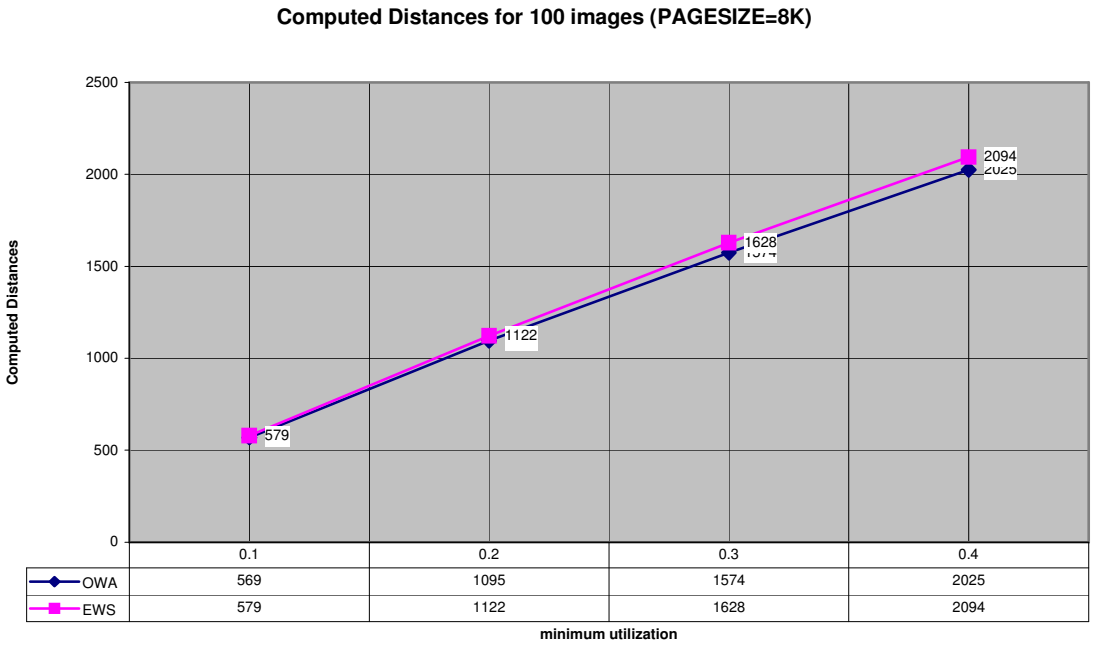


Figure 5.13: Computed Distances for 100 images as a function of minimum utilization with page size = 8K

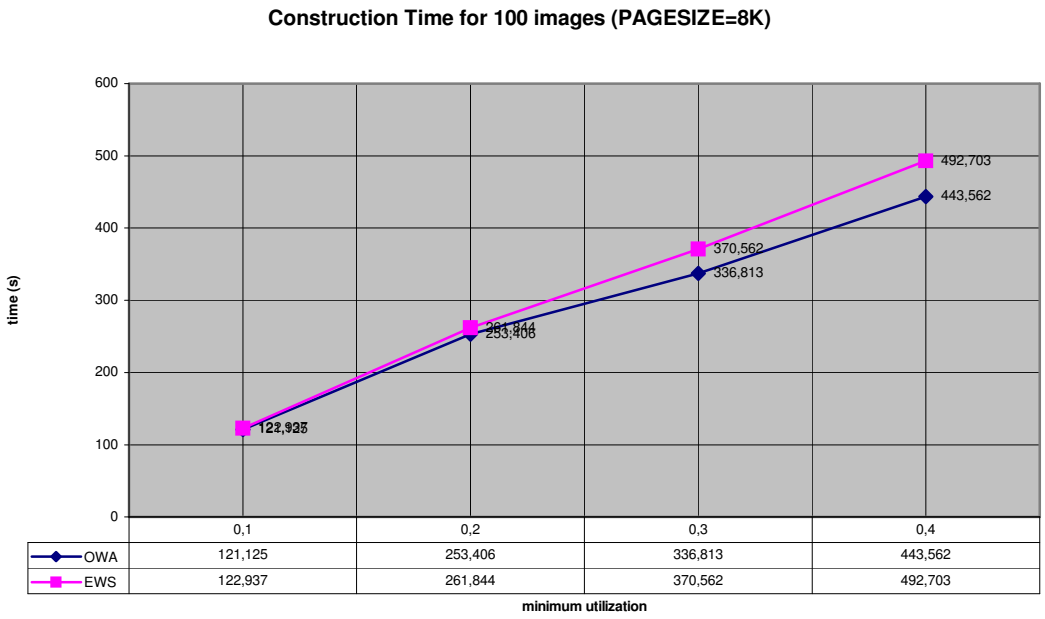


Figure 5.14: Construction Time for 100 images as a function of minimum utilization with page size = 8K

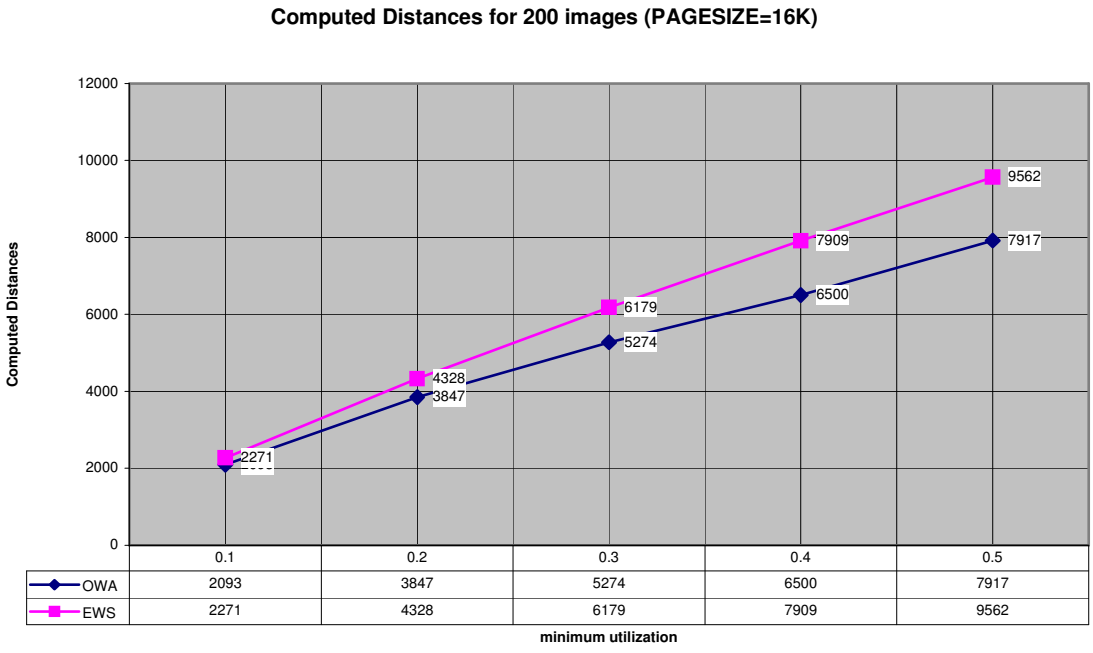


Figure 5.15: Computed Distances for 200 images as a function of minimum utilization with page size = 16K

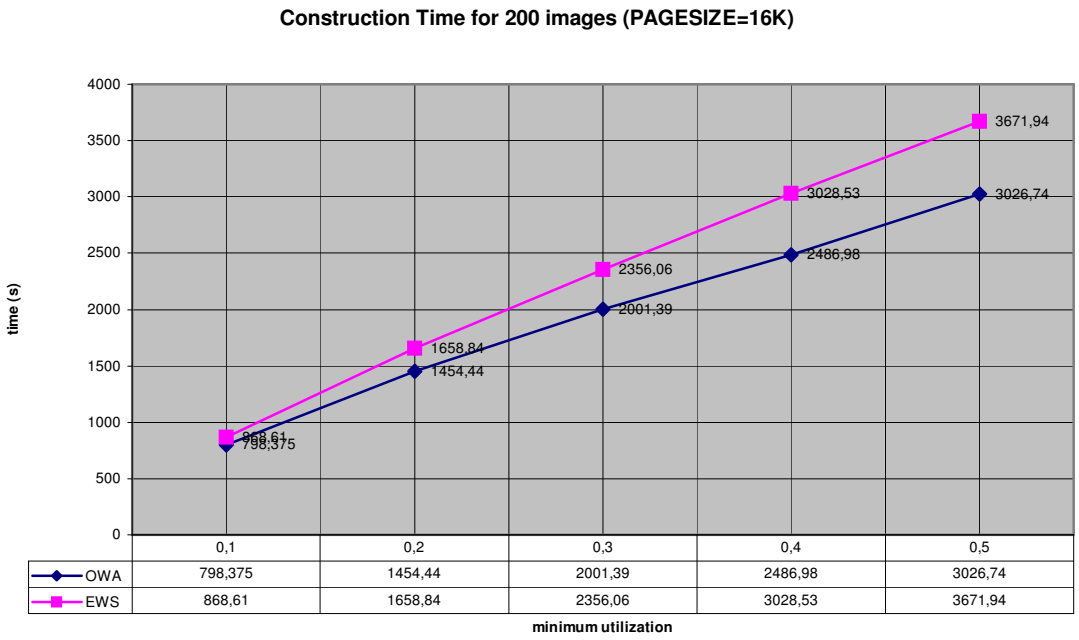


Figure 5.16: Construction Time for 200 images as a function of minimum utilization with page size = 16K

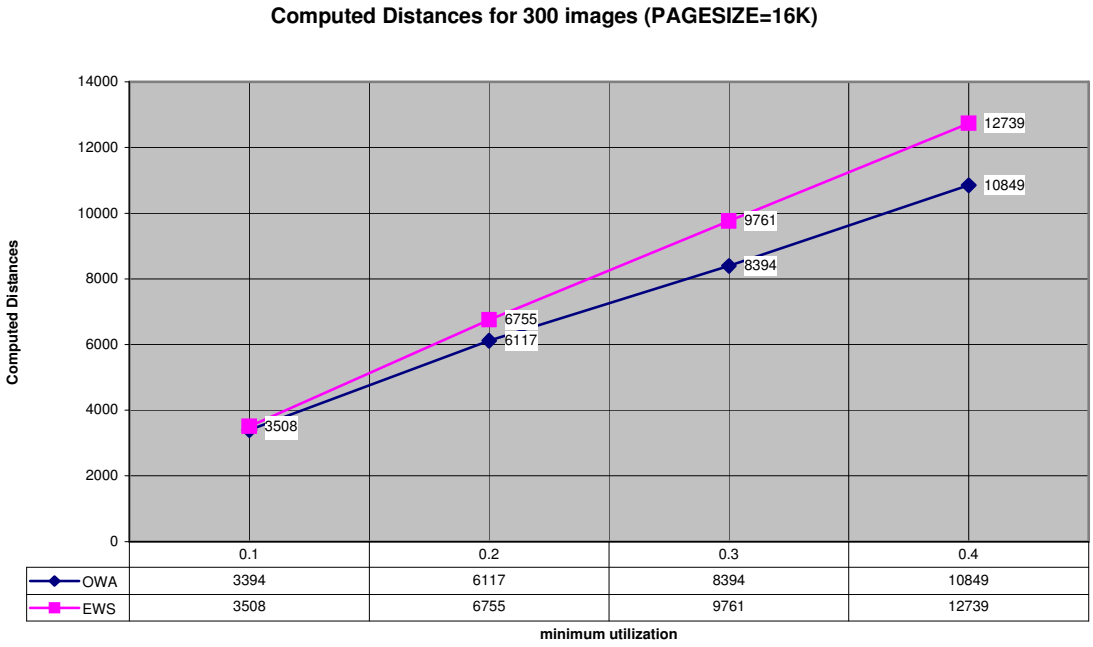


Figure 5.17: Computed Distances for 300 images as a function of minimum utilization with page size = 16K

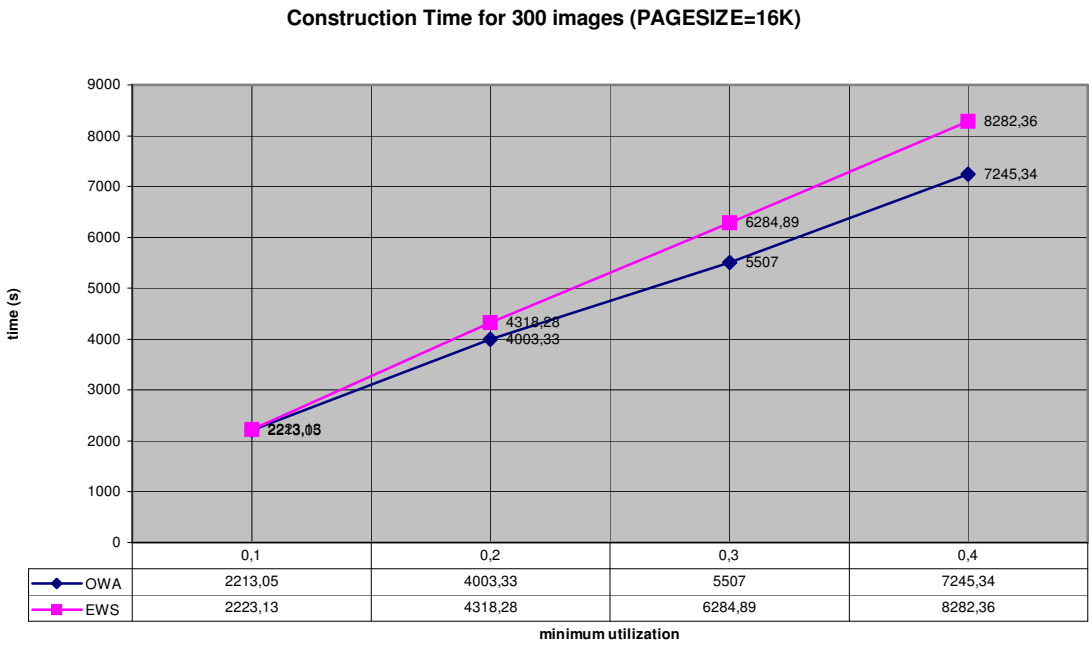


Figure 5.18: Construction Time for 300 images as a function of minimum utilization with page size = 16K

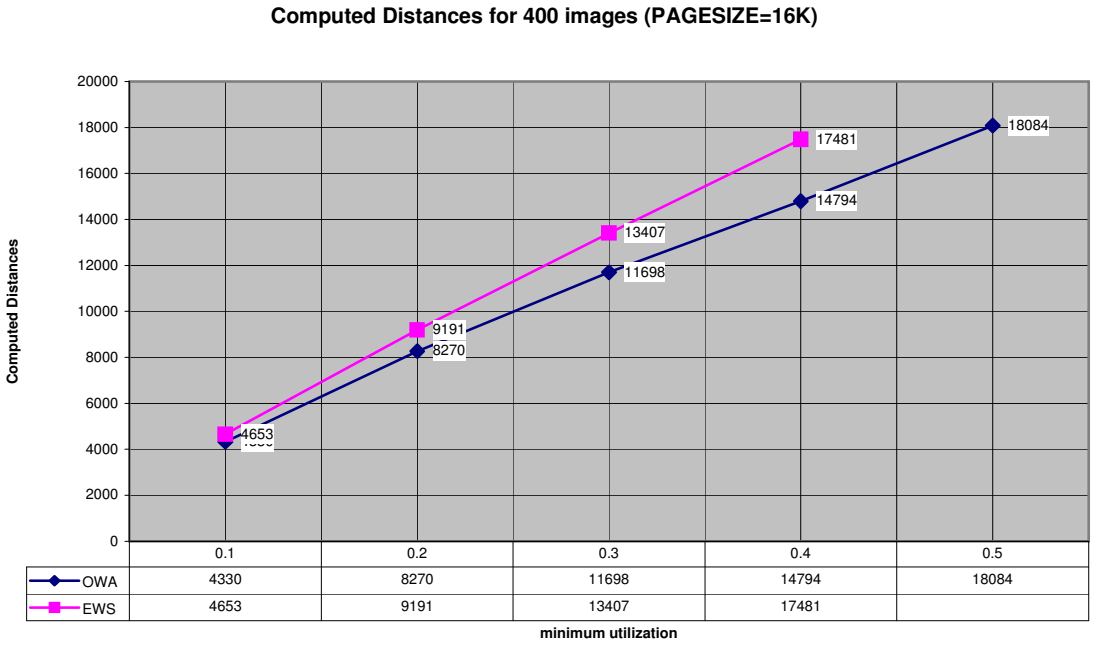


Figure 5.19: Computed Distances for 400 images as a function of minimum utilization with page size = 16K

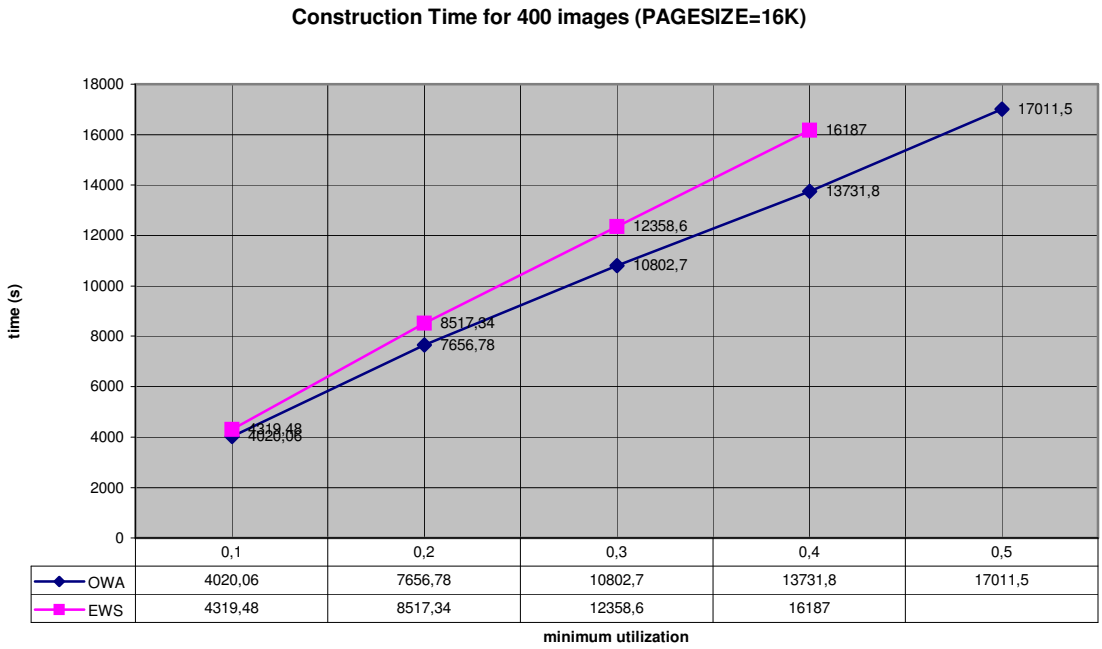


Figure 5.20: Construction Time for 400 images as a function of minimum utilization with page size = 16K

# 5.2 Querying the M-Tree

## 5.2.1 Retrieval Effectiveness

To evaluate the retrieval effectiveness of querying the M-Tree, we have used ANMRR performance metric [47]. This value is defined as follows:

First, we denote NG(q), K(q), R(k) as follows,

$NG(q)$ : the number of the ground truth images (expected result images) for a query q.

$K(q) = min(4 * NG(q), 2 * GTM)$, Where $GTM$ is $\max\{NG(q)\}$ for all $q$'s.

$R(k)$ = rank of an image k in retrieval results.

Rank(k) is defined as follows,

$$Rank(k) = \begin{cases} R(k), & if \quad R(k) \le K(q) \\ (K+1), & if \quad R(k) > K(q) \end{cases} \tag{5.1}$$

Using equation (5.1), AVR(Average Rank) for query $q$ is defined as follows:

$$AVR(q) = \sum_{k=1}^{NG(q)} \frac{Rank(k)}{NG(q)} \tag{5.2}$$

However, with ground truth sets of different size, the AVR value depends on $NG(q)$. To minimize the influence of variations in $NG(q)$, MRR (Modified Retrieval Rank) is defined as follows,

$$MRR(q) = AVR(q) - \left( \frac{1 + NG(q)}{2} \right) \tag{5.3}$$

The upper bound of MRR depends on $NG(q)$. To normalize this value, NMRR (Normalized Modified Retrieval Rank) is defined as follows,

$$NMRR(q) = \frac{MRR(q)}{(K+1) - 0.5 \cdot (1 + NG(q))} \tag{5.4}$$

NMRR($q$) has values between 0(perfect retrieval) and 1(nothing found). And evaluation measure value for whole set over query sets, ANMRR (Average Normalized Modified Retrieval Rank) is defined as follows,

$$ANMRR(q) = \frac{1}{Q} \cdot \sum_{q=1}^{Q} NMRR(q) \tag{5.5}$$

where Q is the total number of the queries.

#### 5.2.1.1 Results

For two types of both M-Tree versions that are created with distance functions, weighted Euclidean with equal weights and weighted Euclidean using OWA, we have tested 335 queries. And also we compared the ANMRR results of our system with the ANMRR results of each three feature (CL, DC and EH) of MPEG-7 XM Software. And results are shown in Table 5.3.

Table 5.3: ANMRR results of Our CBIR System and XM Software for 335 queries.

| | | ANMRR value (335 queries) |
|---|---|---|
| M-Tree - Non-Fuzzy DC Distance | Weighted Euclidean with OWA | 0. 342271 |
| M-Tree - Non-Fuzzy DC Distance | Weighted Euclidean with equal weights | 0. 394931 |
| M-Tree - Fuzzy DC Distance | Weighted Euclidean with OWA | 0.355033 |
| M-Tree - Fuzzy DC Distance | Weighted Euclidean with equal weights | 0.398003 |
| MPEG-7 XM Software | (Only CL Feature) | 0. 338113 |
| MPEG-7 XM Software | (Only DC Feature) | 0. 407258 |
| MPEG-7 XM Software | (Only EH Feature) | 0. 423513 |

### 5.2.2 K-NN Query

To evaluate the effectiveness of k-NN query, we have tested 400 queries to retrieve top 10 images (k=10) from the XML database, which has 400 images. For two types of M-Tree that are created with distance functions we have tested the tree with following parameters:

- PROMOTE_PART_FUNCTION: RANDOM
- SECONDARY PART FUNCTION: RANDOM
- SPLIT_FUNCTION: G_HYPERPL
- PAGE_SIZE: 16K
- MIN_UTIL: 0.1

#### 5.2.2.1 Distance Computations

The number of the distance computations is important for evaluating query performance. Because the distance function in M-Tree is expected to be complex, this number is directly related to the performance of the CBIR system.

In Table 5.4, minimum and maximum number of distance computations is shown for 400 queries.

#### 5.2.2.2 Query Cost Time

In Table 5.4, minimum and maximum number of query cost time is shown for 400 queries.

Table 5.4: Minimum and Maximum Query Cost Time and Computed Distances for 400 Queries in 10-NN Query.

| | Query Cost Time (s) | | Computed Distances | |
|---|---|---|---|---|
| | Min | Max | Min | Max |
| Weighted Euclidean with OWA | 194.516 | 398.297 | 215 | 403 |
| Weighted Euclidean with Equal Weights | 342.516 | 400.859 | 383 | 406 |

## 5.3 Discussion

We have designed and implemented a content-based image retrieval system that evaluates the similarity of each image features in its database to a query. For efficient search and retrieve process, we have built M-Tree index structure. Our system has been tested for constructing and for querying this tree.

While creating M-Tree, number of distance computations and cost time are the key values for evaluating efficiency of the system. For this purpose, tests for building the tree include the number of distance computations and construction time for M-Tree using weighted-Euclidean distance function with OWA and for M-Tree using weighted-Euclidean distance function with equal weights. Tests for building the tree has been made for two different promotions, Confirmed and Random. In both promotions, our database contains 100, 200, 300 and 400 images. Page size parameter

of the index structure varies from 8K to 32K and minimum utilization parameter is between 0.1 and 0.5. For construction time, it can be seen from the figures of Section 5.1.2.1 and Section 5.2.1.2 that a significant improvement can be achieved by using OWA operators in distance function to calculate distance. For example, for confirmed promotion with 16K page size, 0.5 as minimum utilization value and 200 images' features in database, M-Tree using weighted-Euclidean distance function with OWA has less construction time (3306.06 s) than M-Tree using weighted-Euclidean distance function with equal weights (3710.55 s).

Number of computed distances is another important value for evaluating the efficiency of the system. Tests have been made for calculating the number of computed distances with same parameters and same databases. From the figures of Section 5.1.2.1 and Section 5.2.1.2 it can be seen that a significant improvement can be achieved by using OWA operators in distance function to calculate distance. For example, for confirmed promotion with 16K page size, 0.5 as minimum utilization value and with database of 200 images' features, M-Tree using weighted-Euclidean distance function with OWA has less distance computation (8626 distance computations) than M-Tree using weighted-Euclidean distance function with equal weights (9551 distance computations).

Tests for querying M-Tree contain same values, number of distance computations and cost time. For k-NN queries, number of distance computations is important for performance of the CBIR system. We have made 400 queries for testing the system that contains 400 images' features. As can be seen from Table 5.4, our approach has less distance computations then equal weighted Euclidean distance function. And this improvement effects to the query cost time, directly.

Also for evaluating retrieval efficiency, we have used ANMRR metric. As can be seen from Table 5.3, our approach has less ANMRR values than equal weighted Euclidean distance function. Also this system achieves a significant improvement according to MPEG-7 XM Software, except CL feature, which has nearly same

performance with M-Tree. MPEG-7 XM Software's search engine with only CL feature has nearly same ANMRR values because of type of the images used in this work. Also it can be easily seen from Table 5.3 that for querying the system, both version of M-Tree have a good retrieval performance. Since we are using fuzzy DC distance in second version of M-Tree, retrieval results of this version has more objects than M-Tree with non-fuzzy DC distance. This is the effect of using color similarity to evaluate fuzzy DC distance.

The results indicate performance improvement using OWA operators for evaluating the weights of weighted Euclidean distance function in CBIR systems. As analyzing experimental results, we show evidence validating our method is effective in CBIR systems.

# CHAPTER 6

# CONCLUSION AND FUTURE WORK

We have designed and implemented a content-based image retrieval system that evaluates the similarity of each image by using OWA operators in its distance function. Also this system is fully based on XML and MPEG-7 frameworks.

For the distance evaluation between images, we use weighted Euclidean distance and each weight is evaluated by using OWA operators. In this system, we used three descriptors of MPEG-7, Color Layout, Dominant Color and Edge Histogram.

Most of the CBIR systems combine these features by associating weights to individual features. Main problem with that is that same weights are associated with the same features for all images in database and sum of these weighted features are used to build an index structure. However, when comparing two specific images, one feature can be more distinctive than the others, as a result that feature must be associated with higher weight. When comparing other two images, tha feature may be less distinctive than the other features and for this reason that feature must be associated with a lower weight. To overcome this problem we used OWA operators to evaluate weights in distance function.

These features are extracted by using MPEG-7 XM Software. Our system stores these features, not image itself, in a XML database, Berkeley DB XML. Our system has been tested on images of Corel database and shown to be an efficient for image retrieval.

This system supports fuzzy querying for whole image and for features of the images. It can be easily seen that both version of M-Tree have a good retrieval performance. Since we are using fuzzy DC distance in second version of M-Tree, retrieval results of this version has more objects than M-Tree with non-fuzzy DC distance. This is the effect of using color similarity to evaluate fuzzy DC distance.

Another difference of both versions comes from normalization process. In the first version, the system normalizes the distance values practically (according to data set). But in second version, normalization process is done by evaluating maximum and minimum distance values in theory.

A crucial future work to be done on our system is to enhance the effectiveness of building the M-Tree. To do this, the distribution of the nodes of the tree may be organized properly by selecting an appropriate split policy and page size parameters to get higher tree level. Thus the pruning efficiency of the tree can increase, and the performance of building and querying the M-Tree may be improved.

In our system, only images are used for indexing and retrieval. So another task can be completed in the future may be using video/audio objects in such a retrieval system.

Our system also does not include insertion/deletion methods for individual objects. Since XML database and M-Tree supports insertion/deletion mechanisms, these methods can be implemented easily.

# REFERENCES


[1] E. S. Konak. "A Content-Based Image Retrieval System For Texture And Color Queries". *,MS Thesis, Bilkent University*, August 2002.

[2] Arnold W.M. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain. "Content-Based Image Retrieval at the End of the Early Years". *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 12*, December 2000.

[3] C. Breiteneder, H. Eidenberger. "Content-Based Image Retrieval in Digital Libraries". *Proceedings of Digital Libraries Conference, pp. 67-74, Tokyo, Japan,* 2000.

[4] L. V. Tran. "Efficient Image Retrieval with Statistical Color Descriptors". *Doctoral Thesis, Linkoping Studies in Science and Technology, Dissertation No. 810, Linkoping University,* Sweden, May 2003.

[5] M. Koskela, J. Laaksonen, E. Oja. "Comparison of Techniques for Content-Based Image Retrieval". *Proceedings of the 12th Scandinavian Conference on Image Analysis*, *pp. 579-586*, Bergen, Norway, 2001.

[6] Y. Rui, T. S. Hang, S. Chang. "Image retrieval: Current technique, promising directions, and open issues," *Journal of Visual Communication and Image Representation, Vol. 10, pp.39-62*, 1999.

[7] Ying Li, X. Wan, C.-C. Jay Kuo. "Introduction to Content-Based Image Retrieval- Overview of Key Techniques", *In V. Castelli and L. D. Bergman, editors, Image Databases, chapter 10, pp. 261-284.* John Wiley & Sons, 2002.

[8] N.-S. Chang, K.-S. Fu. “Query by pictorial example”. *IEEE Transactions on Software Engineering*,6(6):519–524, November 1980.

[9] R. Fagin. “Fuzzy Queries in Multimedia Database Systems”. *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, *pp. 1-10,* IBM Almaden Research Center,Proc. 1998.

[10] MPEG Home Page, http://www.cselt.it/mpeg/ . Last date accessed: April, 2004.

[11] MPEG-7 XM Homepage,http://www.lis.ei.tum.de/research/bv/topics/mmdb.html Last date accessed: September, 2004.

[12] T. Sikora. “The MPEG-7 Visual Standard for Content Description—An Overview”. *IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 6*, June 2001.

[13] MPEG-7 DDL Working Draft 4.0, Beijing, July 2000

[14] SleepyCat Software, www.sleepycat.com. Last date accessed: September, 2004.

[15] P. Ciaccia, M. Patella, and P. Zezula. “M-tree: An efficient access method for similarity search in metric spaces”. *In Proceedings of the 23rd VLDB International Conference, pp. 426--435*, Athens, Greece, August 1997.

[16] V. Castelli. “IBM Research Report Multidimensional Indexing Structures for Content-based Retrieval”. *IBM Research Division, RC 22208,* February 2001.

[17] A Guttman. “R-trees a dynamic index structure for spatial searching”. *Proc ACM SIGMOD Int Conf on Management of Data, pp. 47-57*, 1984.

[18] K. Lin, H.V. Jagadish, C Faloutsos. “The W-Tree: An Index Structure for High-Dimensional Data”, *VLDB Journal,3, pp.* 517-542, 1994.

[19] S. Berchtold, D.A. Keim., H. Kriegel. “The X-Tree: An Index Structure for High-Dimensional Data”. *Proc. VLDB '96, pp. 28-39,* San Fransisco, U.S.A, 1996.

[20] R.R. Yager. “On ordered weighted averaging aggregation operators in multi-criteria decision making”. *IEEE Trans. Systems Man Cybernet.* 18, *pp.* 183-190, 1988.

[21] D.A.White and R.Jain. “Similarity indexing with the SS-Tree”. In *Proc. 12th Int. Conf. on Data Engineering*, *pp.* 516–523, New Orleans, USA, February 1996.

[22] N.Kata yama and S.Satoh. “The SR-tree: an index structure for high-dimensional nearest neighbor query”. In *Proc. 1997 ACM SIGMOD Int. Conf. on Management of Data*, *pp. 369–380*, Tucson, AZ, 12-15 May 1997.

[23] J.L. Bentley. “Multidimensional binary search trees used for associative searching”. *Communications of the ACM*, 18(9):509–517, 1975.

[24] J.T. Robinson. “The k-d-b-tree: A search structure for large multidimensional dynamic indexes”. In *Proc. 1981 ACM SIGMOD Int. Conf. on Management of Data*, *pp. 10–18*, May 1981.

[25] D.Lome t and B.Salz berg. “The hB-tree: a multiattribute indexing method with good guaranteed performance”. *ACM Trans. Database Systems (TODS)*, *vol. 15, No. 4, pp.625–658*, December 1990.

[26] W.Niblack, R.Barber. “The QBIC Project”. *In Proc. SPIE Storage and Retrieval for Image and Video Databses*, *Vol. 1, pp. 173-187*, Bellingham, Wash, Feb 1993.

[27] C. Faloutsos, M. Flickner, W. Niblack, D. Petkovic, W. Equitz, and R.Barber. "Efficient and effective querying by image content". *Technical report, IBM Research,* 1993.

[28] H. Tamura, S. Mori, T. Yamawaki. "Texture features corresponding to visual perception". *IEEE Trans. on Sys., Man. and Cyb.*, *Vol. 6, No. 4, pp. 460--473,* 1978.

[29] W. Equitz, W. Niblack. "Retrieving images from a database using texture alogrithms from the QBIC system". *Technical Report RJ 9805, Computer Science, IBM Research,* 1994.

[30] B. Scassellati, S. Alexopoulos, M. Flickner. "Retrieving images by 2D shape:a comparison of computation methods with human perceptual judgments". In *Proc. of SPIE Storage and Retrieval for Image and Video Databases*, *pp. 2-14,* San Jose, California, 1994.

[31] D. Lee, R. Barber, W. Niblack, M. Flickner, J. Hafner, D. Petkovic. "Indexing for complex queries on a query-by-content image database". *In Proc. of IEEE Int'l Conf. on Image Processing, pp.142-146,* 1994.

[32] W. Y. Ma, B. S. Manjunath. "Netra: A toolbox for navigating large image databases". In *Proc. of IEEE Int. Conf. on Image Processing, vol. 1, pp. 568-571*, Santa Barbara, 1997.

[33] A. Pentland, R. W. Picard, S. Sclaro. "Photobook: Content-based manipulation of image databases". *International Journal of Computer Vision, vol. 18, no. 3, pp. 233 - 254*, 1996.

[34] J. R. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, C. F. Shu. "The Virage image search engine: An open framework for image

management". In *Proc. of SPIE Storage and Retrieval for Image and Video Databases, vol. 2670, pp. 7687,* 1996.

[35] J. R. Smith, S. F. Chang. "Intelligent Multimedia Information Retrieval". *Ed. M. T. Maybury*, *chap. 2, pp.23-43, MIT Press*,1996.

[36] J. R. Smith, S. F. Chang. "Visually searching the web for content". *IEEE Multimedia Magazine*, *vol. 4, pp. 12-20,* 1997.

[37] "MPEG-7 Overview (version 9)". *International Organisation for Standartadisation, ISO/IEC JTC1/SC29/WG11*, May 2003.

[38] T. Ojala, M. Aittola, E. Matinmikko. "Empirical Evaluation of MPEG-7 XM Color Descriptors in Content-Based Retrieval of Semantic Image Categories". *Proc. 16th International Conference on Pattern Recognition, vol. 2, pp. 1021 – 1024,* Quebec, Canada, 2002.

[39] corel database, http://www.corel.com . Last date accessed: April, 2004.

[40] G. Beliakov. "Definition of general aggregation operators through similarity relations". *Fuzzy Sets and Systems, vol. 114, no. 3, pp. 437-453,* 2000.

[41] J. M. Hellerstein, J. F. Naughton and A. Pfeffer. "Generalized Search Trees for Database Systems." *Proc. 21st Int'l Conf. on Very Large Data Bases*, Zürich, *pp. 562-573,* September 1995.

[42] The M Tree Project Homepage, http://www-db.deis.unibo.it/Mtree/index.html . Last date accessed: July, 2004.

[43] P. Ciaccia, M. Patella. "Bulk Loading the M-tree". *ADC'98*, *pp. 15-26,* Australia, 1998.

[44] R. C. Veltkamp, M. Tanase. “Content-Based Image Retrieval Systems: A Survey”. *Utrecht University*, The Netherlands, October 2000.

[45] Y. Rui, A. C. She and T. S. Huang. “Modified Fourier descriptors for shape representation - a practical approach.” *Proc. of First International Workshop on Image Databases and Multimedia Search,* Amsterdam, The Netherlands,1996.

[46] M. Patella, “M-Tree User Guide Version 0.911”, December 19,2000.

[47] B.S. Manjunath, P. Salembier, T. Sikora. “Introduction to MPEG-7: Multimedia Content Description Interface”, John Wiley & Sons, 2002.

[48] H. Eidenberger. “How good are the visual MPEG-7 features”, *Proc. of the 5th ACM SIGMM international workshop on Multimedia information retrieval, pp.130-137,* Berkeley, California, 2003.

[49] R. Fagin. “Combining Fuzzy Information from Multiple Systems”, *Proc. Fifteenth ACM Symp. On Principles of Database Systems, pp.216-226, Montreal,* 1996.

[50] K. K. Guner, “MPEG-7 Compliant ORDBMS Based Image Storage and Retrieval System”, ,*MS Thesis,Middle East Technical University*, January 2004.

[51] B. Verma, S. Kulkarni. “Fuzzy Logic Based Interpretation and Fusion of Color Queries”, *International Conference on Knowledge Based Computer Systems, KBCS’02, pp. 107-116,* India, 2002.

# APPENDIX A

# QUERYING THE SYSTEM

In this chapter, running examples of query module of the system are shown.

## A.1 Fuzzy Query

We divide fuzzy query into three parts;

- Image-Based Query
- Feature-Based Query
- Color-Based Query

Example queries and results are shown in figures for both version of M-Tree. All queries had been made over 'Flower' data set.

### A.1.1 Image-Based Query

Query: *'Very Similar Images to Example Query'*.

Figure A.1- Results for Query with Fuzzy DC Distance


Figure A.2- Results for Query with Non-Fuzzy DC Distance

### A.1.2 Feature-Based Query

Query: *'Very Similar in CL AND Almost Same as DC OR Almost Same as EH features of Example Image'*


Figure A.3- Results for Feature-Based Fuzzy Query with Fuzzy DC Distance


Figure A.4- Results for Feature-Based Fuzzy Query with Non-Fuzzy DC Distance

### A.1.3 Color-Based Query

Query: *'Red Color: mostly AND Green Color: normal OR Blue Color: not important'*


Figure A.5- Results for Color-Based Fuzzy Query with Fuzzy DC Distance


Figure A.6- Results for Color-Based Fuzzy Query with Non-Fuzzy DC Distance

## A.2 K-Nearest Neighbor Query

Example queries and results are shown in figures for both version of M-Tree. All queries had been made over ‘Flower’ data set.

Query: ’*Top 8 similar images to example image’*.


Figure A.7- Results for k-NN Query with Fuzzy DC Distance

Figure A.8- Results for k-NN Query with Fuzzy DC Distance