Republic of Iraq
Ministry of Higher Education & Scientific Research
Al-Nahrain University
College of Science

Development of a Content-Based Image Retrieval System

A THESIS
SUBMITTED TO THE COLLEGE OF SCIENCE OF
AL-NAHRAIN UNIVERSITY IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

By

MEHDI GZAR DUAIMI
B.Sc., Computer Science, Nahrain Univ., 1992
M.Sc., Computer Science, Nahrain Univ., 1995

Supervisors

DR. LOAY E. GEORGE **DR. LAITH A. AL-ANI**

December, 2006 AD Dhul-Hijja, 1427 AH

Dedication

For My Children:

Qudama &

Qudus.

Supervisors Certification

We certify that this thesis was prepared under our supervision at the Department of Computer Science/College of Science/Al-Nahrain University, by **Mehdi Gzar Duaimi** as a partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science.

Signature: [illegible]	Signature: [illegible]
Name: **Dr. Loay E. George**	Name: **Dr. Laith A. Al-Ani**
Title: **Senior Researcher**	Title: **Assistant Professor**
Date: 24/12/ **2006**	Date: 24/12/ **2006**

In view of the available recommendations, I forward this thesis for debate by the examination committee.

Signature: T Bashaga

Name: **Dr. Taha S. Bashaga**

Title: **Head of the Department of Computer Science**

Date: 24/12/ **2006**

Certification of the Examination Committee

We the examination committee certify that we have read this thesis titled "**Development of a Content-Based Image Retrieval System**" and we have examined the student **Mehdi Gzar Duaimi** in its contents and what is related to it, and in our opinion it meets the standard of a thesis for the degree of **Doctor of philosophy in Computer Science**.

Signature: [illegible]
Name: ***Prof. Dr.*** **Imad Hussain Al-Hussaini**
Title: Chairman
Date: 15/03/2007

Signature: [illegible]
Name: ***Assist. Prof. Dr.*** **Jane Jaleel Stephan**
Title: Member
Date: 19/3/2007

Signature: [illegible]
Name: ***Assist. Prof. Dr.*** **Bushra K. AL-Abudi**
Title: Member
Date: 12/3/2007

Signature: [illegible]
Name: ***Assist. Prof. Dr.*** **Nasser N. Khamiss**
Title: Member
Date: 12/3/2007

Signature: [illegible]
Name: ***Assist. Prof. Dr.*** **Ali Abid D. AL-Zuky**
Title: Member
Date: 13/3/2007

Signature: [illegible]
Name: ***Assist. Prof. Dr.*** **Laith A. Al-Ani**
Title: Member (*Supervisor*)
Date: 12/3/2007

Signature: [illegible]
Name: ***Senior Researcher Dr.*** **Loay E. George**
Title: Member (*Supervisor*)
Date: 12/3/2007

Approved by the Dean of the College of Science, Al-Nahrain University.

Signature:
Name: ***Assist. Prof. Dr.*** **Laith A. Al-Ani**
Title: Dean of the College
Date: / /2007

Acknowledgment

First and foremost, I would like to express my sincere gratitude to my supervisors; *Dr. **Loay E. George*** and *Dr. **Laith A. Al-Ani*** for their excellent guidance and enthusiastic support throughout the duration of this work. They were always there to answer my questions about different aspects of the project.

Many thanks go to the staff of the College of Science at Al-Nahrain University including Computer Science Dept. staff and Head of the Dept. *Dr. **Taha S. Bashaga*** for his valuable assistance.

I am grateful to my referees; chairman and members of the examination committee, for carefully reading and commenting this thesis.

I appreciate the efforts of my friends and especially my family for preparing nurturing environment to work in. Finally, I would like to extend my gratitude to ***everyone*** who contributed to this thesis by pleasant collaboration, motivation, or inspiration.

Mehdi

Abstract

In this thesis, a Content-Based Image Retrieval (CBIR) system is presented that supports querying with respect to color and texture low-level features. The fundamental idea is to generate automatically image descriptors by analyzing the image content. The focus will be on computing global similarity between images. Query is made upon images of homogeneous color/texture that do not require segmentation. The selected images domain is fashion and interior design.

The underlying techniques are based on the adoption of Gray Level Co-occurrence Matrix (GLCM) and correlogram (correlation histogram) as statistical approaches to texture analysis. In addition, cumulative histogram and moments are utilized in color analysis. These techniques are applied in separated and combined manners.

Each image is represented by features vector(s) in the features space. These vectors are indexed using an iterative clustering algorithm called Hierarchical Agglomerative Clustering (HAC) which provides easy-to-index data structures as well as faster query execution facilities. The degree of similarity between images is defined by the distance in the features space. Given a query image, the system first extracts its features vector, and then compares this vector with those of the images pointed along the index structure using wide or narrow search algorithms. In this way, the matched images could be ranked and put into group according to the distance of their features vectors to the query one. This ranked group is considered as the query result.

During the evaluation process a comparison study is made between different applied retrieval schemes. Cumulative histogram proved to be the best according to the selected domain, both as a separated retrieval scheme or when it is combined with GLCM or correlogram, respectively. The conducted experimental evaluation showed that the clustering based indexing algorithm offers high retrieval accuracy with a considerable reduction in the number of required similarity comparisons. Search efficiency is improved due to the fact that the query image is not compared exhaustively with all the images listed in the database.

Acronyms

Bmp	Bit-Map Image
B-Tree	Balanced Tree
CBIR	Content-Based Image Retrieval
CIE	Commission Internationale de l'Eclairage
CMY	Cyan Magenta Yellow
DBMS	Database Management System
DNA	Deoxyribo Nucleic Acid
FE	Feature Extraction
GLA	Generalized Lloyd–Max Algorithm
GLCM	Gray Level Co-occurrence Matrices
GUI	Graphical User Interface
HAC	Hierarchical Agglomerative Clustering
HCA	Hierarchical Cluster Analysis
HIS	Hue Intensity Saturation
HLS	Hue Lightness Saturation
HSB	Hue Saturation Brightness
HSV	Hue Saturation Value
HSx	refers to HSI, HSV, HSB, and HSL color spaces
IDM	Inverse Difference Moment
IR	Information Retrieval
k-NN	k-Nearest Neighbor
MAP	Mean Average Precision
MIS	Management Information System
MSE	Mean-Squared Error
PCA	Principle Component Analysis
PDF	Probability Density Function
PVQ	Product Vector Quantization
QBIC	Query by Image Content system by IBM
RDBMS	Relational Database Management System
RGB	Red Green Blue
SM	Similarity Measurement
SVQ	Standard Vector Quantization
VIR	Visual Information Retrieval
VQ	Vector Quantization
YIQ	Y-axis In-phase Quadrature

Abbreviations

CH	Cumulative Histogram
CO	Co-occurrence Matrix
Comp.	Component
COR	Correlation
Correlogram	**Correl**ation Histo**gram**
CR	Correlogram
Cum. Hist.	Cumulative Histogram
ENT	Entropy
HOM	Homogeneity
Mom	Moments
N	Narrow Search
P	Precision
PRM	Cluster Prominence
Quan.	Quantization
R	Recall
Sec	Seconds
SHD	Cluster Shade
W	Wide Search

Contents

Chapter One

General Introduction

1.1 Overview

The steady growth of the Internet, the falling price of storage devices, and the increasing pool of available computing power make it necessary and possible to manipulate very large repository of digital information efficiently. Generally speaking, Content-Based Image Retrieval (CBIR) technique aims to develop techniques that support the effective searching and browsing tasks performed on large image digital libraries by using automatically derived image features [1].

The rapid increase in the size of digital image collections has motivated the research on image retrieval. The early researches on image retrieval discipline proposed annotating the images manually to facilitate their retrieval. Obviously, annotating images manually is a cumbersome and expensive task for large image databases, and is often subjective, context-sensitive and incomplete [2].

Sometimes, it is almost impossible to describe the content of an image by words, especially for the textures found in an image. As a result, it is difficult for the traditional text-based methods to support a variety of task-dependent queries [3]. To avoid manual annotation, an alternative technique was developed. It is called CBIR, by which images would be indexed according to their visual contents (such as color, texture, and shape). The basic task of a CBIR system is to find similar images (according to their visual features) within a large image database [4]. Typically, a CBIR system pre-processes an image database in order to extract information from all images in the database. This information is referred to as visual features of images. The visual features may be represented by different representations, which facilitate image's searching and indexing [5].

Mostly, the two-step approach to search the image database is still adopted. In the first step, a feature vector is extracted from each image in the database. These features vectors which characterize some image properties are stored in a dedicated features database. In the second step, for a given query image, its features vector is computed, and then compared with the features vectors contained in the features database. The most similar images to the query image are returned to the user. The similarity measure used to compare the features vectors should be effective enough to ensure correct matching between similar images, as well as being able to discriminate dissimilar ones [6]. In the next sections, the fundamental principles of CBIR are presented.

1.2 Historical Perspective

CBIR is a technique that uses the visual contents to search the images listed in large scale image databases according to users' interests. Since 1990s, this research area has been one of most active and rapid growing disciplines [3]. Early work on image retrieval can be traced back to the late 1970s. Since 1979, the application potential of image database management techniques has attracted the attention of researchers from multiple disciplines [7].

The early techniques were not generally based on visual features but on the textual annotation of images. In other words, images were first annotated with text and then searched using a text-based approach from traditional database management systems. In the early 1990s, it was widely recognized that a more efficient and intuitive way to represent and index visual information would be based on properties that are inherent in the images themselves [8].

Researchers from the communities of computer vision, database management, human-computer interface, and information retrieval were attracted to this field. Since 1997, the number of research publications on the techniques of visual information extraction, organization, indexing, query interaction, and database management has increased enormously. Similarly, large numbers of academic and commercial retrieval systems have been developed by universities, government organizations, companies, and hospitals [3].

1.3 Visual Information Retrieval

The term “Information Retrieval” (IR) was coined in 1952, and since 1961 it has gained popularity in the research community. One may simply describe such a system as one that stores and retrieves information. In the past, information retrieval has meant textual information retrieval, but the above definition still holds when applied to Visual Information Retrieval (VIR). However, there is a distinction between the type of information and the nature of the retrieval of text and visual objects [9].

Textual information is linear while images are bi-dimensional. Generally, there are two approaches to solve the VIR problem. They are based on the form of the visual information: attribute-based and feature-based methods. Attribute-based methods rely on traditional textual information retrieval and Relational Database Management System (RDBMS) methods. While feature-based solutions concentrate on visual features such as color, texture, and shape of images that would be indexed according to these features [10].

1.4 Image Databases

Interest in applications involving the search and management of digital images has increased tremendously over the last few years. Image databases exist for storing art collections, satellite images, medical images and many other real-time applications. Image databases can be huge, containing hundreds of thousands of images. However, it is not possible to access or make use of this information unless it is organized to allow for efficient browsing and retrieval. CBIR systems are required to effectively and efficiently access images using information contained in image databases. A CBIR system uses information from the content of images for retrieval and helps the user retrieve database images relevant to the contents of a query image [11, 12].

1.5 Principal Components of a CBIR System

CBIR uses the visual contents of an image such as *color*, *shape*, *texture*, and *spatial layout* to represent and index the image. In a typical image retrieval system; its layout is shown in Figure (1.1), the visual contents of the images in the database are extracted and described by multi-dimensional features vectors. The features vectors of the images in the database form a features database. Then, the system changes these examples into its internal representation of features vectors. To retrieve images, Graphical User Interface (GUI) is used to provide the retrieval engine with example images or sketched figures to search for [13].

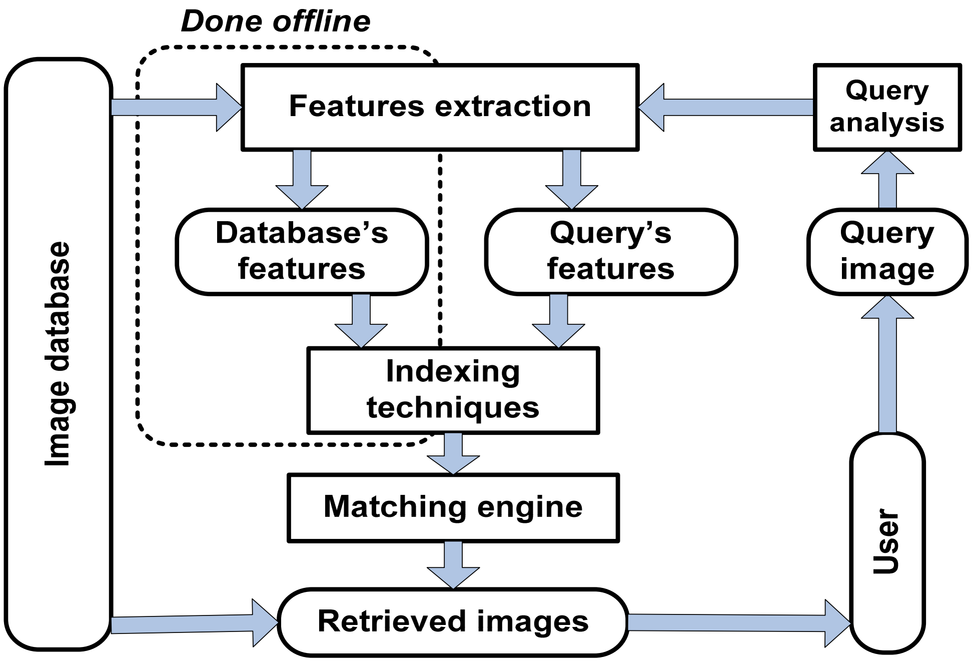


Figure (1.1) The principal components of a CBIR system. [10]

Afterwards, the similarities /distances between the features vectors of the query example and those of the images in the database are calculated. Retrieval is performed as a last step with the aid of the built indexing structure [3].

The indexing scheme provides an efficient way to search for the image database. The image with the smallest distance to the given image is retrieved as the result. In current CBIR systems, instead of only retrieving the best image, a number of similar images are retrieved and sorted according to their distance to the given image. [5]

1.6 Emergence of the Image Retrieval Problem

Researches in the analysis, classification, and retrieval of images from large visual repositories are of the most active topics in Visual Information Retrieval (VIR). There are a few reasons for this: First, the retrieval problem is of great practical interest. While digital cameras make picture taking inexpensive, and large amounts of new images become available on the web every day, there is still a shortage of effective tools for searching / manipulating visual content. Second, it touches a significant number of unsolved challenging questions in image understanding [14].

The difficulties faced by text-based retrieval became more and more severe. The efficient management of the rapidly expanding visual information became an urgent problem. This need formed the driving force behind the emergence of CBIR techniques [3].

1.7 Practical Applications of CBIR

The primary benefit of using content-based retrieval is the reduced time and effort required to obtain image-based information. With frequent adding and updating of images in massive databases, it is often not practical to require manual entry of all attributes that might be needed for queries, and content-based retrieval provides increased flexibility and practical value [15].

A wide range of possible applications for CBIR technology has been identified. The applications are sometimes divided into three categories depending on the goal of the query [11]:

1. Target search: The target image is known (but the user's memory of it might not be exactly correct, he will misplace some objects, or ask for a slightly different color distribution).
2. Category search: Search images from a category (e.g. a person wants to buy a dress, but is not sure of exactly how it should look, she has only a notion of the general texture and tone of color).
3. General browsing: The goal can be very vague or even unknown– "I know it when I see it".

The first category is an objective search, while the other two are subjective. There are many applications that fall into one of the three categories mentioned above. Potentially the most fruitful areas include [1, 9, 16]:

1. Trademark registration
2. Architectural and engineering design
3. ***Fashion and interior design***
4. Crime investigation (i.e, Face matching).
5. Military uses like target identification.
6. Journalism and advertising,
7. Medical diagnosis,
8. Geographical information and remote sensing systems,

9. Weather forecasting,
10. Education and training,
11. Home entertainment,
12. Web searching and picture archiving, and
13. Digital libraries, galleries and museums.

Closer examination of many of these areas reveals that, few examples of fully-operational CBIR systems can yet be found [16, 17]. Image samples used in this research fall mainly in the above third category (i.e, *fashion and interior design*) in which, the integration of color and texture features is of great value.

1.8 Fashion and Interior Design

Imagery is very important for graphic, fashion and industrial designers. Visualisation seems to be part of the creative process. Whilst there are individual differences in the way designers approach their task, many of them use images of previous designs in the form of pictures, photographs and graphics and other visual information from the real world, to provide inspiration and to visualise the end product. There is also a need to represent the way garments hang and flow [17].

The need for CBIR can be observed in the design process of many fields, including fashion and interior design. Here, the designer has to work within externally-imposed constraints. The ability to search a collection of fabrics to find a particular combination of color or texture is increasingly being recognized as a useful aid to the design process [18]. Interior designers follow general principles of form, space, color, and style. Similarly, fashion designers could use a database with images of fabric swatches, designs, concept sketches, and finished garments to facilitate their creative processes [15].

Generally, designers and their customers regularly face the tedious chore of manually searching for, and matching materials according to general design principles and taste. There is an opportunity for a high degree of computer assistance with these tasks [18]. Designers could compose queries using primitive and logical features and specify constraints according to design principles. They may want to inspect patterns from a large collection of images which look similar to a reference color and/or texture pattern [19].

Results of a series of queries must be self-consistent according to designer specified rules. These requirements may lead an interest toward the following query classes [18]:

1. Color: Fabrics and wallpapers are often selected according to their color content. This is a perfect application of color features.
2. Texture: Floor coverings, wallpaper, and fabric all have important textural components ideal for queries based on textural features

From a customer point of view, a web-based interface to a retail clothing catalog might allow users to search by traditional categories (such as style or price range) and also by image properties (such as color or texture). Thus, a user might ask for formal shirts in a particular price range that are off-white with pin stripes [15].

So far, little systematic development activity has been reported in this area. Attempts have been made to use general-purpose CBIR packages for specific tasks such as color matching of items from electronic versions of mail-order catalogues, and identifying textile samples bearing a desired pattern, but no commercial use appears to be made of this at present. Hence the ability to search design archives for previous examples which are in some way similar, or meet specified suitability criteria, can be valuable [17].

1.9 Functions of a Typical CBIR System

A typical CBIR system─ shown in Figure (1.1) has major functions [6, 20]:

1. Analyze the contents of the source information, and represent the contents of the analyzed sources in a way that will be suitable for matching user queries (space of source information is transformed into features space for the sake of fast matching in a later step). This step is normally very time consuming since it has to process sequentially all the source information (images) in the database. However, it has to be done only once and can be done off-line.
2. Analyze user queries and represent them in a form that will be suitable for matching with the source database. Part of this step is similar to the previous step, but applied only to the query image.
3. Define a strategy to match the search queries with the information in the stored database. Retrieve the relevant information in an efficient way. This step is done

online and is required to be very fast. Modern indexing techniques can be used to reorganize the features space to speed up the matching process.

1.10 Problem Definition

The image retrieval problem can be defined as: Let there be an image database, populated with images ($I_0, I_1, I_2, \ldots, I_{n-1}$). Let Q denote a query image, and let *D* denote the real inter-image distance function. The real inter-image distance between two images I_x and I_y is denoted by $D(I_x, I_y)$. The goal is to efficiently and effectively retrieve a set *R* of the best ($k <= n$) images from the image database such that $D(Q, I_i) < t$, where I_i stands for any image in the set *R*, *t* is a certain threshold, and *n* is the number of images in the database.

1.11 Project Motivation

Image databases are becoming increasingly popular due to the large amount of images that are generated by various applications and the advances in computation power, storage devices, scanning, networking, image compression, and desktop publishing. Many application areas need better techniques and mechanism to store and retrieve such huge amount of images [21].

The early implementation of image databases were based on simply giving descriptive keywords to each image, and allowing users to make query on these keywords for accessing the images. However, since text-based methods have real deficiencies, a CBIR approach is proposed based on image features extracted automatically from images [2]. Since color and texture are fundamental aspects of human visual perception, a set of techniques for search and manipulation of color and texture patterns is developed in an integrated manner [28].

Although great progress has been made in both theoretical research and system development of CBIR, none of the existing search engines offers a complete solution to the general image retrieval problem. In addition, many challenging research problems continue to attract researchers [3]. This study is a step forward in solving these open research issues.

1.12 Project Methodology

This project will focus on some specific features, particularly color/texture based features for general image searching applications. However, there is no single best feature that gives accurate results in any general setting. Usually a costumed combination of color/texture features is needed to provide adequate retrieval results.

Features of single-textured images are extracted to construct a features vector by applying Gray Level Co-occurrence Matrices (GLCM) for each textured image. In addition, GLCM is extended to colored textures by introducing a Correlation Histogram (Correlogram). A set of color features is extracted using different color features extraction methods such as color histogram, cumulative histogram, and color moments. Then, the extracted features vectors are clustered into groups by using hierarchical clustering techniques, and a representative (i.e., template vector) for each group is selected according to specific rules.

These representatives (or template vectors) are then used in query matching process to narrow down the entire search space. In this way, a much smaller subset of the whole database is explored in order to reply for user query. Cluster based retrieval has some advantages over retrieval performance. In this thesis we are focusing on the integration of color and texture features, for pattern retrieval and matching.

1.13 Literature Review

Although CBIR is still immature, a lot of prior work is performed in this discipline. Several image retrieval systems are now available as commercial or experimental packages, with demonstration versions available on the Web.

Among these CBIR systems are:

1. ***CANDID***, (1994): Comparison Algorithm for Navigating Digital Image Database operated by University of California for the US Department of Energy [7]. Some of local features are extracted (e.g. color and/or texture) at every pixel location. Instead of making a histogram with a discrete number of bins of the features vectors, the continuous probability density function (pdf) is calculated over the multidimensional features space. The k-means clustering algorithm followed by an optional cluster merging process is used. Several similarity measures between the probability density functions are proposed (e.g. the L_2 distance measure).
2. ***QBIC,*** (1995)*:* IBM's Query By Image Content [22].

   It is probably the best-known of all image content retrieval systems. It is available either in standalone form, or as part of other IBM products (such as the DB2 Digital Library). It offers retrieval by any combination of color, texture, or shape— as well as by text keyword. The involved color features are: the 3D average color vector of an object or the whole image in RGB, YIQ, Lab, and Munsell color space. The utilized texture features are the modified versions of the coarseness, contrast, and directionality features which are proposed by Tamura. The multidimensional features vector is firstly reduced by using the KL transform, and then indexed by using R* trees to improve the search efficiency.

3. ***Excalibur Visual RetrievalWare,*** (1996): by Excalibur Technologies [23].

   It is one among the first systems, used by Yahoo Image Surfer, and Infoseek WWW search engines. The system creates a features vector for the image based on HSV color histograms, relative orientation, curvature and contrast of lines in the image, and texture attributes (that measure the flow and roughness in the image). There is also a text retrieval system that uses a semantic network to link words. The indexing technique is based on neural network methods.

4. ***Netra,*** *(1996): 'netra'* means eye in Sanskrit— an ancient Indian language. This system was developed by Manjunath et al, at the Department of Electrical and Computer Engineering, University of California [36]. In this system images are automatically segmented into about 6—12 non-overlapping homogeneous regions. The RGB color space is quantized, and represented by a color codebook of 256 colors. Texture is represented by a features vector containing the normalized mean and standard deviation of a series of Gabor wavelet transforms of the image. Indexing is based on the SS-tree. Color, texture, and shape are indexed separately.

5. ***VisualSEEK,*** (1997): by Columbia University, New York [26].

   It is the first of a whole family of experimental systems. Both global color histograms and regional color descriptions are analyzed (after transformation from RGB-space into HSV-space). Texture histograms are derived from spatial-frequency channels (i.e., Wavelet transform sub-bands). Web images are identified and indexed by an autonomous agent which assigns them to an appropriate subject category according to their associated text.

6. ***MARS,*** *(1997):* The Multimedia Analysis and Retrieval System project at the University of Illinois [33].

   Color is represented in the HSV color space, and then color histogram and color moments are calculated. It uses Tamura textures and co-occurrence matrices in different directions to extract coarseness, contrast, directionality and inverse difference moments. The features can be extracted globally or from 5 x

5 sub-images. The k-means clustering method, in the color-texture space, is applied.

7. ***Surfimage, (1998):*** project at Institute National de Research en Informatique et en Automatique (INRIA), France [4]. This has a similar philosophy to the MARS system. The low level features are RGB color histogram, edge-orientation histogram computed after applying a Canny edge detector. Texture signature is derived from the gray-level co-occurrence matrix, Fourier transform, and Wavelet transform.
8. ***ASSERT,*** (1999): Automatic Search and Selection Engine with Retrieval Tools, developed at Indiana University / Medical Center, and Purdue University / School of Electrical and Computer Engineering [37].

A human (physician) marks interesting areas in the lung. The region is extracted using binary-image analysis routines. Using a statistical approach based on gray-level co-occurrence matrix, a set of texture features is extracted. Mean and standard deviations of the gray-levels with respect to the pixels in the rest of the lung and the histogram of the local gray-levels are measured. The dimension of the set of features vectors is reduced into 12 attributes. A multi-dimensional hash table is used to index the features efficiently.

9. ***PicSOM,*** (1999), Pictures with Self Organizing Maps, performed at Laboratory of Computer and Information Science, Helsinki University of Technology [31]. The Average values of color components (R, G, and B) are calculated in five separate regions of the image resulting in a 15-dimensional color features vector. The Y-values of the YIQ color representation of every pixel's 8-neighborhood are examined, and the estimated probabilities for each neighbor being brighter than the center pixel are used as features. The features vectors are then separately quantized with a Tree-Structured vector quantization algorithm that uses Self-Organizing Maps (TS-SOMs) at each of its hierarchical levels.
10. ***MultiResolution Image Database Search,*** (2000), Jau-Yuen Chen et al, Purdue University, West Lafayette [16].

Histograms of color, texture, and edge features are calculated for a number of regions in the image (maximal 4 x 4 regions). The L*a*b* space is used. The texture feature is formed by histogramming the magnitude of the local image gradient for each color component. To speed up the search a Tree-Structured Vector Quantization (TSVQ) is calculated.

11. ***An Approach to Content-Based Image Retrieval using Clustering and Space Transformation***, (2002), Shah, B.N. et al, Louisiana University [11].

    It proposes a new approach to CBIR that uses space transformation methods to transform the original low-level image space into a reduced dimension vector space that enables efficient query processing. This system uses the inexpensive "estimated" distance, as opposed to the computationally inefficient "real" distance, to retrieve the relevant results for a given query image.

12. ***Efficient Image Retrieval with Statistical Color Descriptors,*** (2003), Tran, L.V., Linkoping University, Sweden [10].

    Color is characterized by the probability distribution of the colors in the image. Distance measures between color distributions are then described in a differential geometry-based framework. In addition to normal distributions, linear representations of distributions are used to derive new compact descriptors for color-based image retrieval.

13. ***Retrieval by Content of Medical Images Using Texture for Tissue Identification***, (2003), Felipe, J.C., Institute of Superior Education, Sao Paulo, Brazil [48].

    This system focuses on medical images representation and comparison, based on image texture features. Regions of images that represent different human body tissues are used. An image is characterized by numeric values (signature) acquired through calculations on the brightness (gray) levels of the image. The inter-relationship between these brightness levels defines the image texture. Besides this, rules to establish the similarity between images based on the respective signatures are proposed. In addition, definition of a new texture descriptor─ the Gradient descriptor is introduced.

14. ***Texture Based Medical Image Indexing and Retrieval: Application to Cardiac Imaging,*** (2004), Glatard, T., et al, Villeurbanne-Cedex, France [50].

    This system is dedicated to medical images. It analyzes medical image properties and evaluates Gabor-filter based features for indexing and classification. The goal is to perform clinically relevant queries on large collection of cardiac images that do not require user supervision. This technique is used for indexing, retrieval, and extracting clinically relevant information out of the images.

15. ***Integrated Querying of Images by Color, Shape, and Texture Content of Salient Objects***, (2004), Saykol, E., et al, Bilkent University, Turkey [52].

    It provides an integrated mechanism to query images by color, shape, and texture content of the salient objects. Users focus on some specific parts of the images for querying purposes. Hence, multiple object regions can be queried on a single image. Video frames can also be processed after the salient objects in frames are extracted by an object extraction tool. The color vector is based on color histograms but the pixels are probabilistically distance-weighted during computations.

16. ***Hierarchical Indexing for Region Based Image Retrieval***, (2005), Aulia, E., Louisiana University [20].

    An improved region-based image retrieval system is developed. The system applies image segmentation to divide an image into discrete regions, which correspond to objects. During image segmentation, a modified k-means clustering algorithm is used to generate the initial number of clusters and the cluster centers. In addition, during distance computation, objects weights are introduced based on object's uniqueness.

1.14 Project Objectives

The basic task of this project is to find similar images (using their visual features) according to a query image within a large image database. Content-Based Image Retrieval (CBIR) system aims to develop techniques that support effective searching and browsing tasks performed on a large image collection by using automatically derived image features.

1.15 Contributions of the Thesis

This research has been made to extract the low-level image features that will be mentioned in chapter two, evaluate distance metrics, and looking for efficient searching/indexing schemes that will be presented in chapter three. The techniques developed to reach the above objectives are:

1. Developing an improved algorithm for statistical texture features extraction based on GLCM for gray textures. Correlation Histogram (Correlogram), i.e. the extended version of GLCM for colored textures, is adopted. Also, cumulative histogram and color moments are computed, and their performance is evaluated and compared.

2. Making similarity distance computation between the query image and the stored images in the database. This distance measure is based on color and textural features.
3. Developing an improved object clustering algorithms with some selected similarity measures driven by an improved index tree. To get faster retrieval speeds, a hierarchical clustering method is implemented in the features database.
4. Analyzing query performance for different color spaces in different quantization schemes. This is done for all the retrieval schemes adopted through out this project.

1.16 Thesis Outline

This thesis consists of six chapters. The contributions of this thesis are presented through chapters four, five, and six. The rest of the thesis is organized as follows:

1. Chapter two introduces the relevant subjects surrounding early and contemporary VIR, and introduces how features are extracted from images to be used in CBIR. In addition, some basic facts about color and texture features are summarized in it.
2. Chapter three reviews some background material on CBIR hierarchical indexing. It describes some useful properties that are convenient for indexed CBIR, and investigates the most commonly used similarity measures between images based on their pre-computed indexed features.
3. Chapter four introduces the proposed indexed color/texture based retrieval system. Also, this chapter provides algorithms on how color/texture features are extracted, how these pre-computed features are indexed using hierarchical agglomerative clustering methods, and finally describes the similarity measures used in comparing the extracted features.
4. Chapter five shows the experimental results with an evaluation of the proposed schemes.
5. Chapter six gives a list of derived conclusions from the results of the presented work. Also in this chapter some proposals are expressed for the direction that the future work may take.
6. Finally, the appendices contain extra information concerning the details on the implementations of this work.

Chapter Two

Visual Features of CBIR

2.1 Introduction

This chapter is organized as follows: Features extraction issues are presented in section (2.2). Also, Image retrieval basic tasks are discussed in this section. Section (2.3) introduces color-based retrieval opportunities with the explanation of different color spaces, in addition to the extracted color features. Different quantization schemes in different color spaces are compared in section (2.4). Color-based features extractors are explained in section (2.5). Section (2.6) presents Color Similarity Measures. Finally; texture-based features, Gray Level Co-occurrence Matrix (GLCM), and texture features descriptors are explained in section (2.7).

2.2 Features Extraction

Image features extraction is the basis of CBIR. Within the visual features scope, the features include color, texture, and shape features. Since the visual features of an image are only based on the image itself, there is no problem of subjectivity [5]. In a CBIR system, different visual features of images are automatically extracted and stored for any future retrieval process. Searching the whole image database is based on searching these visual features [40].

Color is the first and most straightforward visual feature for indexing and retrieval of images. A typical color image taken from a digital camera, or downloaded from the Internet normally has three color channels, while gray images have only one [10]. Among various low-level features, the color information has been extensively studied because of its invariance with respect to image scaling and orientation. However, they do not support a detailed comparison of the color appearance among images [23].

Texture is another important feature of images. Texture perception plays an important role in the human visual system of recognition and interpretation due to the identifying and describing characteristics of texture feature [50]. Since the power of texture increases when combined with color, the proposed CBIR system provides techniques for querying with respect to texture and color in an integrated manner. The extraction of the texture and color content of the images takes place both during the database population phase and querying phase [5].

2.2.1 Features Selection

It could be useful to characterize image features into three levels of abstraction with increasing complexity [17, 37]:

Level 1: comprises retrieval by primitive features such as color, texture, shape or the spatial location of image elements. Its use is largely limited to specialist applications such as trademark registration, fashion design.

Level 2: comprises retrieval by derived (sometimes known as logical) features, involving some degree of logical inference about the identity of the objects depicted in the image.

Level 3: comprises retrieval by abstract attributes, involving a significant amount of high-level reasoning about the meaning and purpose of the objects or scenes depicted.

2.2.2 Significance of Features Space Selection

In developing a complete CBIR system, the following four important issues must be addressed [12]:

1. Features Space Selection: determines what image feature, or combination of image features, are to be used for image matching and retrieval purposes.
2. Features Capturing: selects algorithms to capture the image features or the image features set identified by the features space selection.
3. Indexing and Search Scheme: creates effective indices and data structures based on the selected features space to speed up image retrieval on the database.
4. Database Query Scheme: provides methods that enable users to effectively form database queries, and to refine the queries based on the retrieved images. Among the above four issues, the features space selection constitutes the most critical strategic decision because it largely affects, or even determines, the remaining three issues of system design.

2.2.3 Features Categories

Although many image features have been explored for content-based image retrieval purposes, they could be classified into the following five categories [35]:

1. Pixel-level features: they include color, location, and other derived features (e.g., the first and second order derivatives of gray scale value) at each pixel.
2. Global features: they include histograms, means, variances, moments and other statistical features computed over the entire image or sub-areas of the image.

3. Textural features: they usually include a combination of parameters that collectively describe characteristics of texture patterns.
4. Object features: they include object regions and edges that are generated as a result of image segmentation and edge ejection operations.
5. Conceptual features: they include identifiers of objects, time, location, and type of event contained in the image.

2.2.4 Features Representation

Keeping track of all the features vectors extracted from each image would pose a major difficulty to any retrieval system. Hence, there is a need for a feature representation to summarize the distribution of features vectors. There are two attributes: feature expressiveness and computational tractability, which are in fact the two main requirements for an effective feature representation. Notice that there are two aspects to tractability: the complexity of density estimation and the complexity of evaluating similarity. While the former is an off-line process that typically does not have great impact on the performance of retrieval systems, the latter must be performed thousands, or millions, of times for every retrieval operation and should be fast [14, 61].

Image content can be represented globally or locally. The former includes feature average value, standard deviation, and histograms (probability density functions) calculated globally for the entire image. While the later can be used for template matching or each local feature accompanied with position. The features are typically calculated off-line and stored for each image, so efficient computation is not a critical criterion (as opposed to efficient image/feature matching in the query stage) [16].

2.2.5 Image Signature

The term feature, as classified in subsection (2.2.2), denotes some elements of images such that it is useful for discriminating between them. For each application, where content based retrieval of images is required, features that provide a high discriminating power must be selected. A signature is a feature, or set of features, as generated for a particular media fragment. This signature is then stored in a features database, or index, with other signatures of the same type [44].

Image signature is a numeric value or a set of them, which can be used to represent an image regarding one or more characteristics of it. For example, one signature that can be assigned to an image is a vector of values obtained from the application of the energy descriptor over its set of co-occurrence matrices. The image signature is generally used as parameter of comparison between two images [48].

A feature is an attribute that characterizes a specific property of an object. An n-dimensional features vector represents an object, where *n* is the selected numbers of attributes. An object may be image, video, sound and etc. More formally, an image I, can be represented as a features vector in the following form:

$$I: < e_1, e_2, e_3, e_4, e_5, e_6, e_7, \ldots, e_n >,$$

where each entry e_i of this vector represents a feature of image I [13].

2.2.6 CBIR Basic Tasks

A CBIR system have two major tasks; see Figure (2.1). The first one is *Features Extraction* (*FE*), where a set of features, called image signatures, are generated to accurately represent the content of each image in the database [61]. A signature is much smaller in size than the original image, typically in the order of hundreds of elements (rather than millions). The second task is *Similarity Measurement* (*SM*), where a distance between the query image and each image in the database using their signatures is computed so that the top *n* "closest" images can be retrieved. Typically, the features used in CBIR systems are low-level image features such as color, texture, shape, and layout [51].

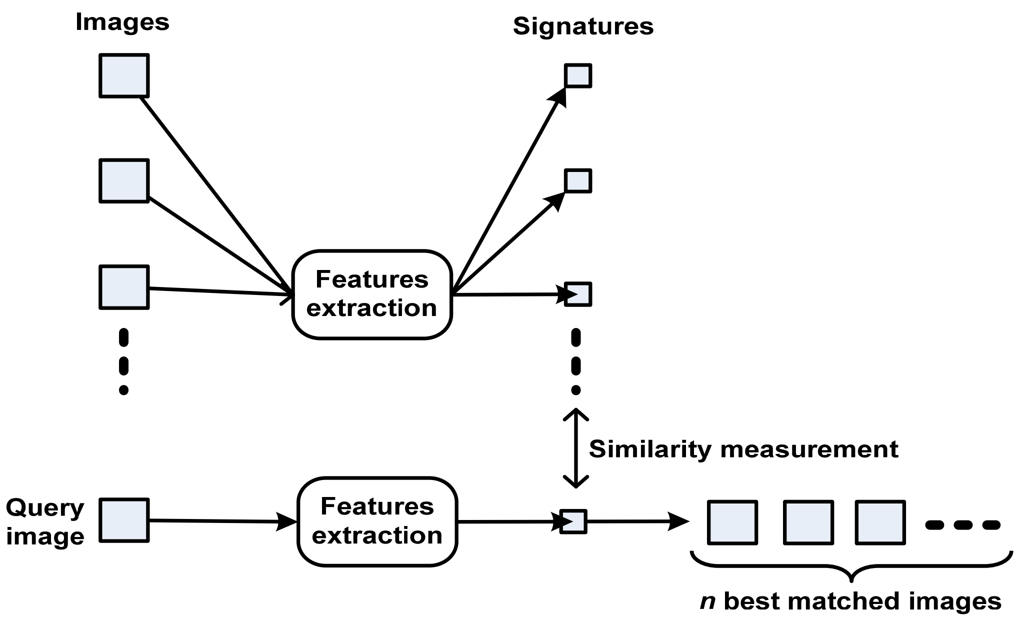


Figure (2.1) CBIR basic tasks [61].

2.3 Color-Based Retrieval

Color feature is one of the most widely used visual features in image retrieval, since color is immediately perceived by human beings when looking at an image. Therefore, Color-based Image Retrieval is the most popular CBIR technique. Color is the most extensively used visual content for image retrieval. Its three- dimensional values make its discrimination potentiality superior to the single dimensional gray values of images [3].

Using color features in CBIR requires taking many factors into consideration: color model selection, color feature representation, and the metric to compute the distance between color features [5]. Color features are relatively easy to extract and match, and have been found to be effective for indexing and searching color images in image databases. Before selecting an appropriate color description, color space must be determined first.

2.3.1 Perceptual Color Components

These components are: H (Hue), S (Saturation), and I (Intensity). "Hue" is what an artist refers to as "pigment"; it is what we think of as "color"; yellow, orange, cyan and magenta are examples of different hues [7]; see figure (2.2). The H parameter is measured as an angle of rotation around the vertical axes. Hue is invariant to the changes in illumination and camera direction and hence more suited to object retrieval [39].

The S parameter (or chroma) refers to the amount of white in the color. It controls the purity or vividness of the color. Low saturation means more white in the color, resulting in a pastel color. Very low saturation results in a washed-out color. For a pure, vivid color, the saturation should be maximum. The achromatic colors black, white, and gray have zero saturation and differ in their values [3].

Saturation represents the degree to which the color expresses its hue, and is calculated as the radial distance from the diagonal axis. All points on that axis have zero saturation, so they correspond to shades of gray. Points farther away from the axis have more saturation; they correspond to more vivid colors [27]. The vertical axis corresponds to the I parameter (or value). It starts at zero (black) at the bottom and ends at one (white) at the top. Intensity refers to the amount of light in the color. It controls the brightness of the color. Maximum lightness always creates white, regardless of the hue. Minimum lightness results in black [39].

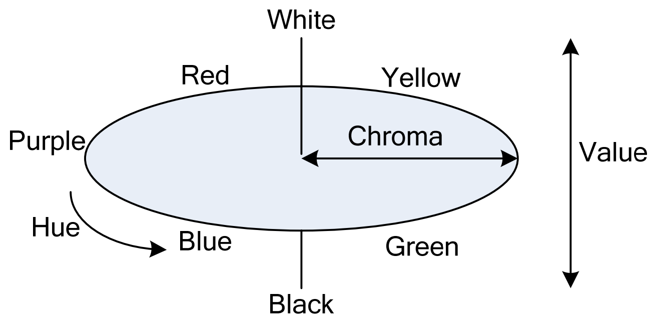


Figure (2.2) Illustration of the three color components [64].

An artist usually starts with a highly saturated (i.e., pure), and intense (i.e., bright) pigment, and then adds some white to reduce its saturation and some black to reduce its intensity. Red and Pink are two different "saturations" of the same hue, Red [7].

2.3.2 Color Spaces (Models)

One of the main aspects of color features extraction is the choice of a color space. A color space is a multidimensional space in which the different dimensions represent the different components of color [20]. A color model is simply a standard way to represent color in mathematical terms. Most color models use a 3D-coordinate system. Each point within the system's subspace represents a unique color [7].

Color models used today can be classified into two categories: hardware-oriented and user-oriented. Hardware-oriented color models are used for most color devices. For instance, the RGB (Red-Green-Blue) color model is used for color monitors and cameras. The CMY (Cyan-Magenta-Yellow) model is a standard used to describe color in the color printing industry, while YIQ (Y-axis In-phase Quadrature) color model is used in broadcast television [59]. User-oriented color models include those based on the three human perceptions of color (Hue, Saturation, and Intensity) [5].

The selection of color models determines the way to represent color content of images as well as consecutive color features representations and the selection of image retrieval techniques. In order to understand many characteristics of color features that are used in CBIR systems, some of the commonly used color models are discussed next

Alternative color spaces can be generated by transforming the RGB color space. The idea for color space transformation is to develop a model of color space that perceptually similar with human color vision [20]. There are many color spaces in use today but there is no agreement on which one is the best.

1. RGB Color Model

The most popular color space is RGB (Red-Green-Blue). RGB space; as depicted in Figure (2.3) is device-dependent, perceptually non-uniform, and perceptually non-linear. Equal distances in the space do not in general correspond to perceptually equal sensations [27]. It is composed of three color components. These components are called "additive primaries" since a color in RGB space is produced by adding them together. The representation of the colors in the RGB space is difficult to understand intuitively [56].

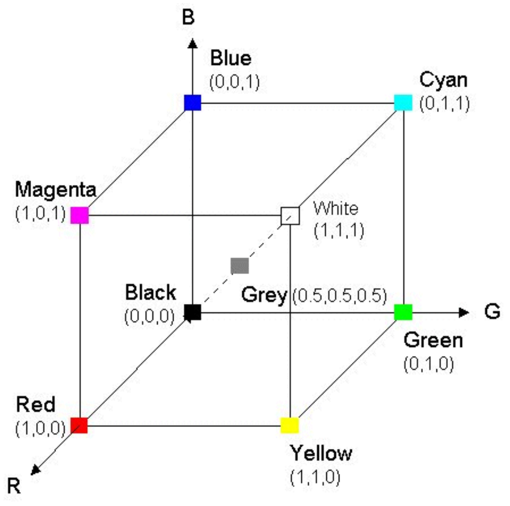


Figure (2.3) The RGB color model [66].

2. Gray Model

The color model GRAY or INTENSITY is calculated from the original RGB tristimulus values. Gray is not perceptually uniform as the measured proximity between two gray-values does not necessarily correspond to the psychological similarity between them. It can be obtained by the following transformation [9]:

$$\text{Gray} = 0.299R + 0.587G + 0.114B \quad , \quad \ldots\ldots\ldots\ldots\ldots (2.1)$$

3. The HSx Color Spaces

The HSI, HSV, HSB, and HLS color spaces (conventionally called ‘HSx’) are more closely related to human color perception than the RGB color space, but are still not perceptually uniform. The axes from the HSx color spaces represent Hue, Saturation, and Lightness (also called as Value, Brightness, or Intensity) color characteristics [91].

These models are based on the color circle mapped on the RGB cube; see Figures (2.4), and (2.5). They provide more intuitive way for perceiving color. The main difference between these models is the definition of the Intensity component. Still these models are perceptually non-linear. For color representation in user interfaces, this group is usually preferred [27].

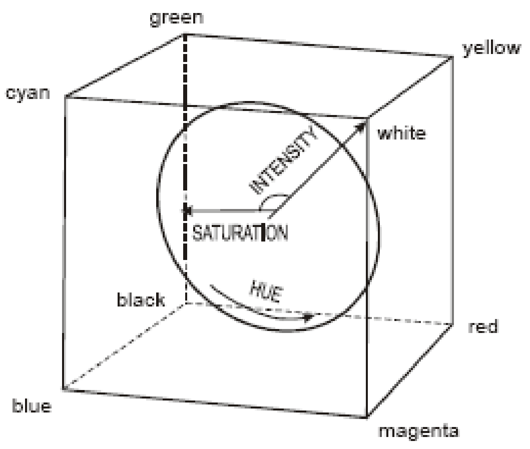


Figure (2.4) Hue-Saturation-Intensity color model within Red-Green-Blue color cube [75].

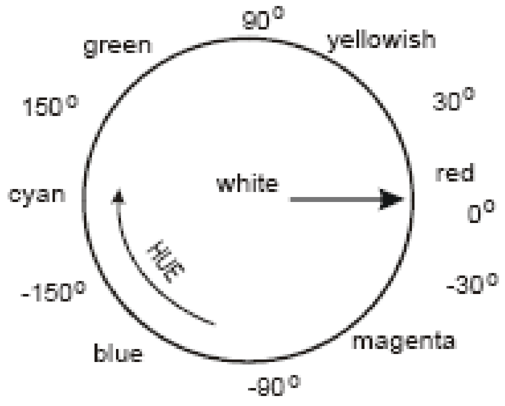


Figure (2.5) The circular color legend (color wheel) [27].

a) HSI Color Model

The HSI (Hue, Saturation, Intensity) color model describes a color in terms of how it is perceived by the human eye. It is an intuitive representation since it corresponds to how a painter mixes colors on a palette. HSI is derived by the following equations [9, 52]:

$$H = arctan\left(\frac{\sqrt{3}(G-B)}{(R-G)+(R-B)}\right) \quad , \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.2)$$

$$S = 1 - 3min(r,g,b) \quad , \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.3)$$

$$I = \frac{1}{3}(R+G+B) \quad , \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.4)$$

where r, g, and b are obtained by dividing R, G, and B by the sum (R+G+B) respectively. While HIS color model is described in Figure (2.6), the HLS model is represented by a double cone similar to it.

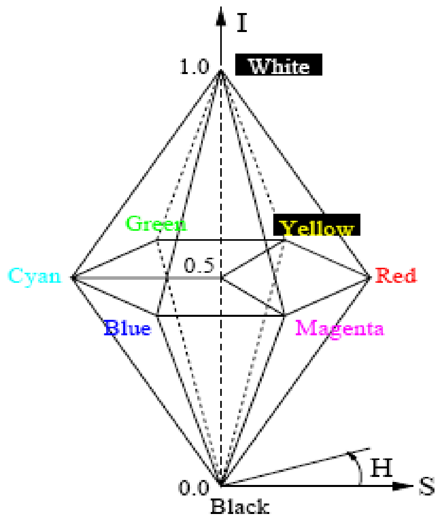


Figure (2.6) The HSI color model [16].

b) The HSV Color Model

In HSV (or HSL, or HSB) space is widely used in computer graphics, and it is a more intuitive way of describing color. V stands for Value (Luminance or Brightness) [3]. Its three components are derived from RGB values as follows [27, 58]:

$$H = \begin{cases} 2\pi - \theta & if\ B > G \\ \theta & otherwise \end{cases} \quad , \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.5)$$

$$S = 1 - \frac{3}{R+G+B}min(R,G,B) \quad , \ldots\ldots\ldots\ldots\ldots\ldots \quad (2.6)$$

$$V = \frac{1}{3}(R+G+B) \quad , \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.7)$$

$$where\ \theta = cos^{-1}\frac{0.5(R-G)+(R-B)}{\sqrt{(R-G)^2+(R-B)(G-B)}}$$

HSV model also uses hue, saturation, and value. It is summarized by the single cone shown in Figure (2.7). Its three components have the same meanings as

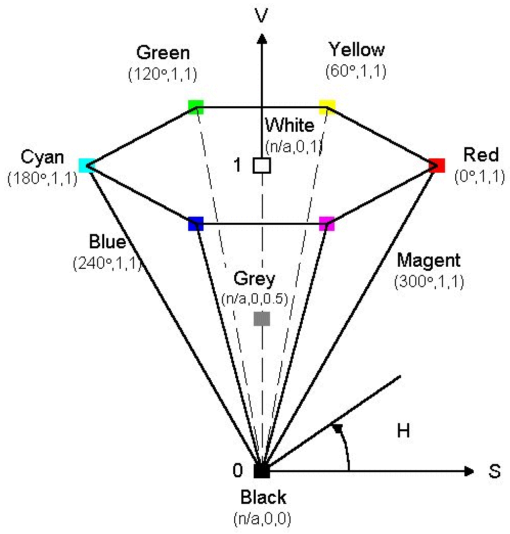


Figure (2.7) The HSV color model [66].

described in subsection (2.3.1). H has the range [0°, 360°). The hue angle in degrees is measured clockwise or counterclockwise. The red, yellow, green, cyan, blue and magenta colors are placed counter-clockwise 60° from each other, starting from red at 0° [39].

HSV color space is a nonlinear transformation of the RGB, but it is easily invertible. The HSV color space is approximately perceptually uniform [20]. The HSV model is very similar to the HIS color model. The main difference between the two is the calculation used to produce the brightness value [7]. Also, RGB coordinates can be easily translated to the HLS coordinates by simple formula similar to equations (2.2), (2.3), and (2.4) [3].

4. CIE L*a*b* and CIE L*u*v*

Also called as LAB (Lab), and LUV (Luv), the Commission Internationale de l'Eclairage (CIE) had recommended these two color spaces that are device independent and considered to be perceptually uniform. They consist of a luminance or lightness component (L) and two chromatic components (a* and b*) or (u* and v*) [3].

CIE LAB is used for surfaces while CIE LUV is for lighting, and video display applications. The perceptual linearity is particularly considered in these color spaces. In the CIE LAB color space, a* and b* are respectively red/green and yellow/blue axes [63]; see Figure (2.8).

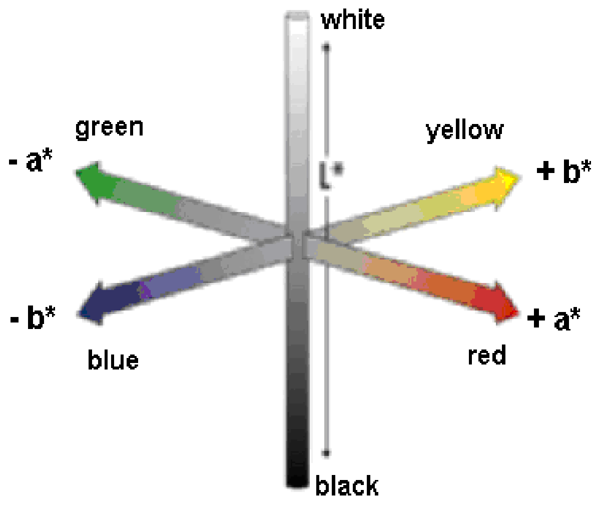


Figure (2.8) CIE LAB color space [10].

Although CIE L*a*b* provides a more uniform color space than previous models, it is still not perfect. CIE LAB values are calculated from CIE XYZ by [9, 56]:

$$L^* = \begin{cases} 116\left(\frac{Y}{Y_n}\right)^{1/3} - 16, & if \ \left(\frac{Y}{Y_n}\right) > 0.008856 \\ 903.3\left(\frac{Y}{Y_n}\right), & if \ \left(\frac{Y}{Y_n}\right) \le 0.008856 \end{cases} \quad , \ \dots\dots\dots\dots\dots\dots \quad (2.8)$$

$$a^* = 500\left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right) \quad , \ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad (2.9)$$

$$b^* = 200\left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right) \quad , \ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad (2.10)$$

$$where \ f(x) = \begin{cases} x^{1/3} & , \ if \ x > 0.008856 \\ 7.787 + (16/116) \ , & if \ x \le 0.008856 \end{cases}$$

the constants X_n , Y_n , and Z_n are the XYZ values for the chosen reference white point. When working with color monitors good choices could be something close to D65's XYZ coordinates [10]. Tristimulus values for the commonly used illuminant D65 are (95.05, 100.0, and 108.88) for (X_n , Y_n , and Z_n) respectively. XYZ color system values could be obtained by the following formula [53]:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.490 & 0.310 & 0.200 \\ 0.177 & 0.813 & 0.011 \\ 0.000 & 0.010 & 0.990 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} , \quad \ldots\ldots\ldots\ldots \quad (2.11)$$

5. Munsell Color Space

The Munsell color order system was developed to provide a notational system for colors. It organizes the colors according to natural attributes. Munsell's book of colors contains 1200 samples of color chips, each assigned a value of H (Hue), V (Value, Intensity), and C (Chroma, Saturation). Each Munsell color chip corresponds to a color generated by some quantization of the RGB color space [29].

The chips are arranged such that unit steps between them are intended to be perceptually equal. The advantage of the Munsell color order system is that it orders a finite set of colors by perceptual similarities over an intuitive three dimensional space. Munsell was designed to be compact (1200 perceptually distinct chips) and complete. There is no simple mapping from color points in RGB color space to Munsell color chips. Rather, Munsell HVC (modified Munsell) which is a good approximate of the original Munsell color space can be produced by the following equations [19, 75]:

$$H = arctan(b^* / a^*) , \quad \ldots\ldots\ldots\ldots \quad (2.12)$$

$$V = L^* , \quad \ldots\ldots\ldots\ldots \quad (2.13)$$

$$C = \sqrt{a^{*2} + b^{*2}} , \quad \ldots\ldots\ldots\ldots \quad (2.14)$$

Munsell HVC color model system is shown in Figure (2.9). However, it does not satisfy the property of uniformity [53].

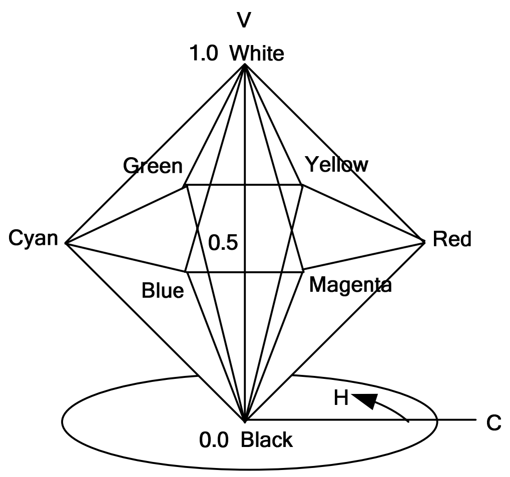


Figure (2.9) The Munsell HVC color model [39].

6. The 9-colors and 11-colors Categories

The categories described in this section are not a color space; rather, they are a human-based segmentation of some standard color spaces. Gong (in 1998) has empirically partitioned the Modified Munsell color space (the same used by QBIC) into eleven color zones defined and validated experimentally by different groups of examiners [12, 25].

The 11-color category is used to segment the HVC color space; as described in Table (2.1), which results in an 11 colors quantization scheme. It is used for color-based image retrieval and compared with other color quantization schemes [24].

Table (2.1) Range of each color zone for 11-colors category [12].

Color name	Hue	Value	Chroma
Red	0 ~ 36 36 ~ 64	4 ~ 9 4 ~ 9	1.5 ~ 30 15 ~ 30
Orange	64 ~ 112	4 ~ 8	9 ~ 30
Yellow	80 ~ 112	9 ~ 10	1.5 ~ 30
Skin color	36 ~ 64 64 ~ 112	4 ~ 9 4 ~ 8	1.5 ~ 15 1.5 ~ 9
Green	112 ~ 196	4 ~ 10	1.5 ~ 30
Cyan	196 ~ 256	6 ~ 8	1.5 ~ 30
Blue	256 ~312	4 ~ 8	1.5 ~ 30
Purple	312 ~ 359	4 ~ 8	1.5 ~ 30
Black	—	< 3	—
Gray	— —	4 ~ 8 3 ~ 4	< 1.5 —
White	—	> 9	< 1.5

In a similar way the 9 colors category is a human-based segmentation of HSV color space, where certain ranges of the three color components are mapped into a single known color label. This category is expressed in reference [59].

2.3.3 Color Space Properties

The desirable characteristics of an appropriate color space for image retrieval are the following [53]:

1. Uniformity: The metric proximity between colors indicates the perceived similarity of colors.
2. Completeness: The color space includes all perceptually distinct colors.
3. Compactness: Each color in the color space is perceptually distinct from the other colors.
4. Naturalness: The color space provides for a natural breakdown of colors into the three basic perceptual attributes of color: hue, brightness, and saturation.

Probably, the most important color space property is its uniformity. Uniformity means that two color pairs that are equal in similarity distance in a color space are perceived as equal by viewers. In other words, the measured proximity among the colors must be directly related to the psychological similarity among them [3].

2.4 Color Quantization

The human eye cannot detect small color differences and may perceive these very similar colors as the same color. This leads to the quantization of color, which means that each color is mapped to some of these pre-specified colors. One obvious consequence of this is that each color space may require different levels of quantized colors, which is nothing but a different quantization scheme [40].

Prior to any processing being performed on a color image, color quantization is a very important step, due to the large number of different colors in the image. Under the RGB color model, there are 256 different color-levels (0-255) for each primary color: red, green, and blue. That means, in a full-color image under the RGB color model, there are $(256 \times 256 \times 256)$ possible colors, in total. Operating with this large color set, storage and processing will both be non trivial [42].

Actually, according to human perception, the difference between two adjacent colors in that large color set is negligible. Therefore, keeping such a big color set is neither practical nor necessary. Color quantization is the procedure used to reduce possible colors to a small number. By using different quantization approaches, such as combining adjacent colors within a predefined range into one single color, the large colors set can be reduced to a small number of possible colors [62]. For example, an image can be quantized from true color with 16777216 possible colors, to only 64 possible colors so that any needed processing on it would be easier [5].

2.4.1 Intensity Slicing

Viewing an image as a 2-D intensity function facilitates the idea of *"Slice"* the intensity (or density) function by one or two slicing planes parallel to the coordinate axes. A certain slicing colors (maximum and minimum) are chosen according to a selected formula [28]. Pixels with gray values above the upper plane are color coded with the maximum color and those below are coded with different colors, while gray values below the lower plane are color coded with the minimum color; as shown in Figure (2.10).

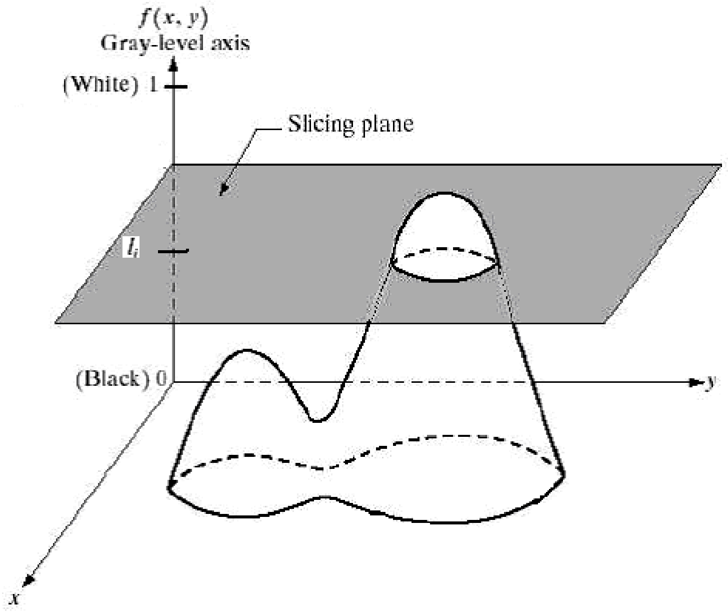


Figure (2.10) Geometric interpretation of the intensity-slicing technique [28].

2.4.2 Quantization Schemes in Different Color Spaces

It has been observed that the fixed color quantization scheme, which is commonly used in computing global and local histograms, has one major drawback. That is, similar colors might be quantized to different bins in the histogram. Therefore, different quatization schemes are proposed according to the selected color space [23]:

1. Uniform Quantization

In uniform quantization, each axis of the color space is uniformly divided into a certain number of bins. It has been shown in the previous section that color distributions are often non uniform, and therefore, a simple uniform quantization scheme is inefficient for some color spaces. The advantage of uniform quantization is that it is a straightforward and natural choice in the absence of a priori information about the color distribution of the image database. Generally, the retrieval performance gets better as the number of quantization bins increases.

2. Standard Vector Quantization (SVQ)

The standard VQ is a method of partitioning the vector space by minimizing the Mean-Squared Error (MSE) with respect to a set of centroid points (i.e., codewords). This procedure can be achieved with a proper initialization and iteration by using the Generalized Lloyd–Max Algorithm (GLA). The retrieval performance of VQ is better than that of uniform quantization as the quantization level increases. It is interesting to point out that the retrieval performance cannot be further improved when the number of quantization bins is larger than a certain threshold.

3. Product Vector Quantization (PVQ)

Although SVQ results in an optimal partitioning of the color space, it is not suitable for large image databases due to its high computational complexity. A simpler but suboptimal quantization method can be adopted to reduce the complexity. That is, we can consider partitioning perpendicular to the axis of the color space. This method is known as Product Vector Quantization (PVQ). PVQ results in a better partitioning of the color space as compared to uniform quantization without a tremendous increase in computational complexity.

The retrieval performance in the RGB space may not increase much by using PVQ due to its correlated color components. In contrast, for the HSV space, the distribution of its color component suggests a PVQ scheme, where H is quantized with a resolution finer than S and V because the human visual system is more sensitive to the hue than to the saturation and value components.

In applying a quantization scheme on a color space, each axis is divided into a number of parts. When the axes are divided in k, ℓ, and m parts, the number of colors (n) used to represent an image will be $n = k \times \ell \times m$.

2.5 Color-Based Features Extractors

Features extraction is the basis of CBIR. Color feature is one of the most widely used visual features. Color is one of the most important low-level features used in VIR [7, 46]. Image abstraction based on color features has been studied extensively in Image Retrieval discipline, and there are multiple ways of representing an image using color features. A few of the most commonly used color measures are the global and local color histograms, dominant colors, and statistical color moments [93].

Average and dominant colors can be used to filter out irrelevant images without too much computational cost. The global color histogram provides a good approach to the retrieval of images that are similar in overall color content. The following subsections discuss some of these color measures [23].

2.5.1 Color Histogram

The most common form of the histogram is obtained by splitting the range of the data into equally sized bins. Then for each bin, the number colors (of the pixels) that fall into each bin are counted and normalized, which gives us the probability of a pixel falling into that bin [10]. The color histogram is easy to compute and effective in characterizing both the global and local distribution of colors in an image. In addition, it is robust to translation and rotation about the view axis and changes only slowly with the scale, occlusion and viewing angle [3, 93]. Given a color image I(x, y) of size X × Y pixels characterized by the color *i* at location (x,y), i.e. *i*=I(x, y). A single bin contains the number of pixels having the color *i.* It is defined by the following equation [19]:

$$h(i) = \frac{1}{XY}\sum_{x=0}^{X-1}\sum_{y=0}^{Y-1}\lambda(I(x,y),i) \quad , \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(2.15)}$$

$$\textit{where} \quad \lambda(w,p) = \begin{cases} 1 & \textit{if } w = \text{p}, \\ 0 & \textit{otherwise.} \end{cases}$$

In the equation above, λ is the unitary impulse function. The *h(i)* values are normalized in order to sum to one. A histogram can be determined for each color component, resulting in three different color histograms for every single image. Another possible method is to have a single color histogram for all of the color channels. In the latter approach, the color histogram is simply a compact combination of three histograms [40].

The more bins a color histogram contains the more discrimination power it has. However, a histogram with a large number of bins will not only increase the computational cost, but will also be inappropriate for building efficient indices for image databases [3]. The computational cost of the histogram-based image classification can be further decreased by decreasing the number of the image gray levels from 256 to 64 or 32 [43]. However, the main drawback of using the color histogram for CBIR is that it only uses color information. Textural and shape properties are not taken into account [91].

2.5.2 Cumulative Histogram

Considering that most Color Histograms are very sparse and thus sensitive to noise, Stricker and Orengo (in 1996) proposed the use of the cumulated Color Histogram [7]. This method is less sensitive to noise. It represents a new kind of histogram G = { g_1, g_2,......, g_n } that could be calculated from the normal color histogram H(I) by applying the following summation [30]:

$$g_j = \sum_{i=0}^{j} h_i \quad , \text{ ....} \qquad (2.16)$$

The result of the cumulative histogram feature extractor is a vector, just like the normal color histogram. In contrast with the normal histogram, this vector is always completely dense, even if only a few colors of the discrete color space appear in the image. A side effect of the robustness of the cumulative histogram technique is the increase of the index size and therefore a lower retrieval speed [25].

2.5.3 Color Moments

A side effect of the technique behind the color histogram and the cumulative color histogram is the quantization of the bins. For storage and retrieval efficiency, colors are grouped together into bins. Similar colors are put together in the same bin. It is very hard to find an optimal quantization of the bins however; e.g., perceptually similar colors may be quantized into different bins, or vice versa. To overcome these effects, the color moments approach was suggested [34].

The mathematical foundation of this approach is that any probability distribution is uniquely characterized by its 12 moments. Thus, if we interpret the color distribution of an image as a probability distribution, then the color distribution can be characterized by its moments [7]. Furthermore, because most of the information is concentrated on the low-order moments, only the first order (mean), the second (variance) and the third order (skewness) color moments have been used. They proved to be efficient and effective in representing color distributions of images [3, 7].

Higher moments, involve more manipulations on the input data, are almost always less robust than lower moments. Therefore, higher moments such as fourth moment or above are rarely used to represent the content of an image [93]. If the value of the ith color channel at the jth image pixel is I_{ij} and the number of image pixels is N, then the three moments related to this color channel are defined as [6]:

$$M_i = \frac{1}{N}\sum_{j=1}^{N} I_{ij} \quad , \quad \text{.........} \quad (2.17)$$

$$\sigma_i = \left(\frac{1}{N}\sum_{j=1}^{N}\left|I_{ij}-\mu_i\right|^2\right)^{1/2} \quad , \quad \text{.........} \quad (2.18)$$

$$S_i = \left(\frac{1}{N}\sum_{j=1}^{N}\left|I_{ij}-\mu_i\right|^3\right)^{1/3} \quad , \quad \text{.........} \quad (2.19)$$

where M_i (mean) is used to estimate the value around which central clustering occurs, σ_i (standard deviation) describes the "width", "dispersion", or "variability" around the mean value, and S_i (skew) characterizes the degree of asymmetry of a distribution around its mean. Skew is non-dimensional and characterizes only the shape of the distribution. With color moments, instead of the complete color distribution of an image, only the major features of the distribution are stored [34, 93].

2.5.4 Color Correlogram

The major drawback of the color histogram is the lack of spatial information in the extracted features [20]. For example, all the patterns shown in Figure (2.11) have the same color proportions, but different spatial distributions.

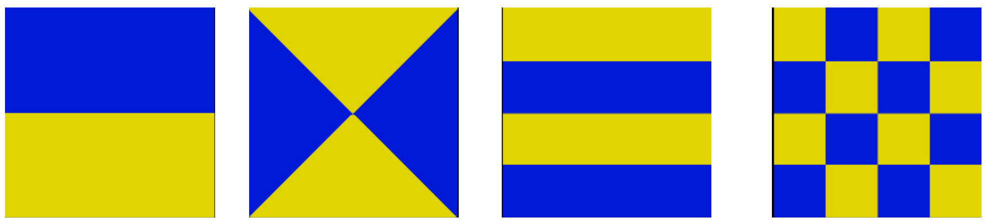


Figure (2.11) Patterns having the same gray proportions, but different spatial distributions [25].

Correlation histogram (correlogram) tries to compensate for this weakness. It takes the spatial correlation aspect of the color distribution into account, and expresses how the spatial correlation of pairs of colors changes with distance. Hence, counting the occurrences of different geometric configurations of colored pixels [21, 24, 45].

A color correlogram is a table indexed by color pairs, where the kth entry for (i, j) specifies the probability of finding a pixel of color j at a distance k from a pixel of color i in the image. Let I represent the entire set of image pixels, and $I_{c(i)}$ represents the set of pixels whose color is c(i). Then, the color correlogram is defined as [3, 25]:

$$C_{i,j}^{k} = Probability_{p_1 \in I_{c(i)}, p_2 \in I} \left[p_2 \in I_{c(j)} \middle| \quad |p_1 - p_2| = k \right] \quad ,.....(2.20)$$

where i, j $\in \{1, 2, \ldots, N\}$, k $\in \{1, 2, \ldots, d\}$, and $| p_1 - p_2 |$ is the distance between pixels p_1 and p_2.

2.6 Color Similarity Measures

If the images to be compared are of different sizes, but have been quantized on a common palette, their histograms can be compared by measuring the similarity between the pre-extracted features. Many similarity (or distance) measures have been proposed. The most commonly used measures are listed:

2.6.1 Histogram Intersection

This is one of the first distance measures in color-based image retrieval. The distance defined is based on the size of the common part of two color histograms. Considering the two histograms *A* and *B*. The similarity between A and B is given by their intersection, defined as [10, 37]:

$$S_{AB} = \sum_{i=1}^{n} \min(A_i, B_i) \; , \; \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.21)$$

In this case the distance between A and B is obtained by:

$$dist_{AB} = 1 - \sum_{i=1}^{n} \min(A_i, B_i) \, , \; \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.22)$$

2.6.2 Minkowski-Form Distance (Lp)

The Minkowski-form distance L_p between two histograms is defined as [10]:

$$L_p = \left(\sum_{i=1}^{n} |A_i - B_i|^p \right)^{1/p} \quad , \text{ .... } \quad (2.23)$$

L_p is called as L_1 metric when p=1, and as L_2 metric when p=2. If the images are of the same size (or the histograms have been scaled to the same size), and quantized on a common palette, their similarity is commonly measured using the sum of the squared differences (L_2 metric), or the sum of the absolute values of differences (L_1 metric). These metrics usually perform poorly, even for the simplest types of query [49]. A slight change in lighting conditions may result in a corresponding shift in the color histogram, causing these metrics to misjudge similarity [53]; as shown in Figure (2.12) between *h and k* color histograms.

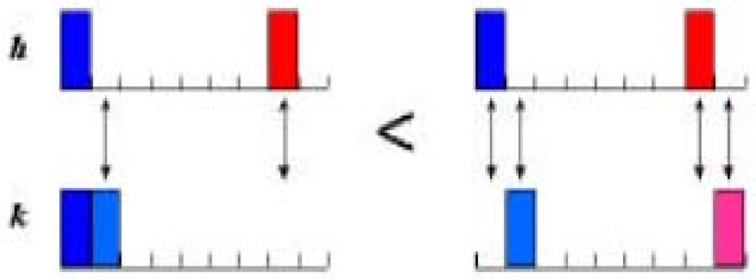


Figure (2.12) Misjudged similarity caused by color shift [25].

2.7 Texture-Based Features

Texture is one of the crucial primitives in human vision and texture features have been used to identify contents of images. One crucial distinction between color and texture features is that color is a point, or pixel property, whereas texture is a local-neighborhood property [32]. As a result, it does not make sense to discuss the texture content at pixel level without considering the neighborhood [60].

It is not easy to give an exact definition of a texture. In literature, textures are defined as visual patterns that have properties of homogeneity that do not result from the presence of only a single color or intensity [21]. Texture is observed in the structural patterns of surfaces of objects such as wood, grain, sand, grass, and cloth. The term texture generally refers to repetition of basic texture elements called texels. A texel contains several pixels whose placement could be periodic, quasi-periodic or random [93].

2.7.1 Texture-Based Features Extractors

Texture-based features extractors focus mainly on the gray level of the pattern. They fall into two main categories; statistical-based and spectral-based. Statistical-based texture extractors employ features extracted from the visual information, which measure coarseness, contrast, directionality and other textural characteristics. Of both categories, the statistical-based texture extractors are most commonly used [9, 21].

1. Statistical Approach

Among the current approaches used in image processing to describe texture is the so called statistical approach. It is widely used because it produces good results with low computational costs. This method considers the distribution of gray levels and their interrelationship [48]. From the statistical point of view, an image is a complicated pattern on which statistics can be obtained to characterize these patterns. The techniques used within the family of statistical approaches make use of the intensity values of each pixel in an image, and apply various statistical formulae to the pixels in order to calculate features descriptors. Texture features descriptors, extracted through the use of statistical methods, can be classified into two categories according to the order of the statistical function that is utilized [40]:

a) First-Order Texture Features:

They are extracted exclusively from the information provided by the intensity histograms, thus yields no information about the locations of the pixels.

b) Second-Order Texture Features:

They take the specific position of a pixel relative to another into account. The most popularly used of second-order features is the *Grey Level Co-occurrence Matrix (GLCM)*. This method roughly consists of constructing matrices by counting the number of occurrences of pixel pairs of given intensities at given displacements and directions. In other words, it counts how often pairs of grey level of pixels, separated by a certain distance and lying along certain direction, occur in an image [20].

2. Spectral Approach

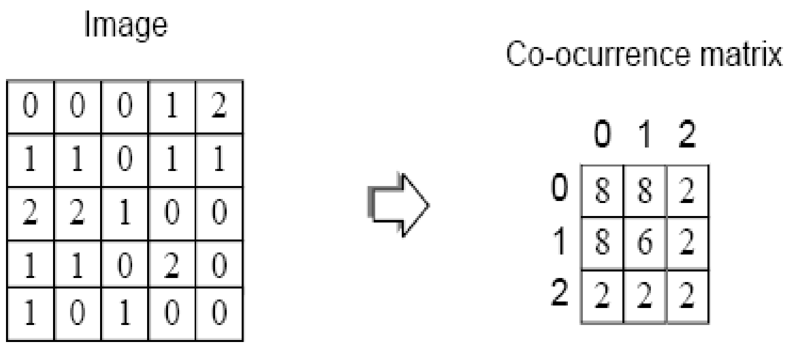
The spectral approach to texture analysis deals with images in the frequency (transform) domain. Spectral-based extractors try to model the pattern into mathematical functions [21]. This approach requires Fourier transform to be carried out on the original images to acquire their corresponding representations in the frequency space [48]. There are several texture-based extractors using spectral domain features, such as discrete Fourier transform (DFT), and discrete wavelet transforms (DWT) [20].

2.7.2 Gray Level Co-occurrence Matrix (GLCM)

In the early 70's, Haralick et al, proposed the Gray Level Co-occurrence Matrix (GLCM), also called Spatial Grey Level Dependency Matrix (SGLDM), as a representation of texture features. This approach explores the gray level spatial dependence of texture. It first constructs a co-occurrence matrix based on the orientation and distance between image pixels and then extracts meaningful statistics from the matrix as the texture representation [7].

Generally, the problem of texture discrimination based on statistical approach implies the analysis of a set of co-occurrence matrices. In each matrix, the indexes of rows and columns represent the given range of the image gray levels, and the value P(i,j) stored at the position (i,j) is the frequency that gray levels i and j occur with, at a given distance d and at a given direction θ [32].

For example, suppose we have the image represented by its gray level matrix that shown left in Figure (2.13). Regarding the 0 direction and the distance 1, the co-occurrence matrix will be like that shown right.



$$\begin{array}{|c|c|c|c|c|}\hline 0&0&0&1&2\\\hline 1&1&0&1&1\\\hline 2&2&1&0&0\\\hline 1&1&0&2&0\\\hline 1&0&1&0&0\\\hline\end{array} \quad \Rightarrow \quad \begin{array}{c|c|c|c|} & 0 & 1 & 2\\\hline 0 & 8 & 8 & 2\\\hline 1 & 8 & 6 & 2\\\hline 2 & 2 & 2 & 2\\\hline\end{array}$$

Figure (2.13) Image quantized in 3 gray levels, sampled in 5x5 pixels, and its co-occurrence matrix for direction = 0^o and distance = 1 [48].

For instance, the value of the co-occurrence P(0,1) = 8 was calculated by scanning the gray level matrix and, for each pixel with value 0, its left and right nearest neighbors were checked and P(0,1) was incremented whenever a value 1 was found.

A co-occurrence matrix, $P_{d,\theta}(i, j)$, is a matrix in which the (i, j)th element describes the frequency of occurrence of two pixels that are separated by distance d in the direction θ with grey levels i and j. Texture variations in a region can be captured through the co-occurrence matrix by various θ and d; see Figure (2.14). That is, the co-occurrence matrix characterizes the spatial interrelationships of the grey levels in a texture pattern and it is invariant under monotonic grey-level transformations [38].

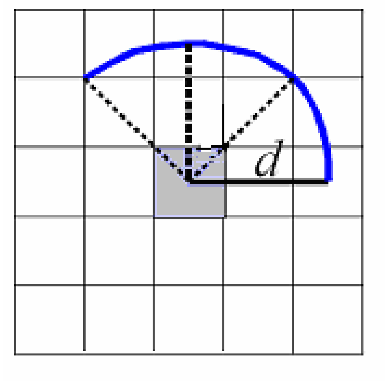


Figure (2.14) Co-occurrence matrix creation using d and θ. [41]

In general, the minimum amount required for the set of co-occurrence matrices is four ($\theta = 0°, 45°, 90°$ and $135°$, respectively; $d = 1$) for the texture, about which we have no prior knowledge [38]. The co-occurrence matrix should be normalized before proceeding with statistical analysis by dividing each entry in the matrix by the summation of all entries of the matrix. Hence, treating the matrix as a probability density function (pdf) [40].

2.7.3 The Formal Definition of a Co-occurrence Matrix

The gray level co-occurrence matrix is determined as follows [13]:
Let $D_x = \{0, 1, \dots, N_{x-1}\}$, and $D_y = \{0, 1, \dots, N_{y-1}\}$ be the spatial domains of row and column dimensions respectively, where N_x and N_y are the number of pixels in axis X and Y respectively. And, $G = \{0, 1, \dots, N_{g-1}\}$ be the domain of gray levels where N_g is the number of gray levels. The Image I can be represented as a 2D function:

$$I: D_x \times D_y \in G$$

For abbreviation, a new domain can be defined, as $D \subset N^2$ (where N is the set of Natural numbers) instead of $D_x \times D_y$. Positions and orientations are shown in Figures (2.15) and (2.16).

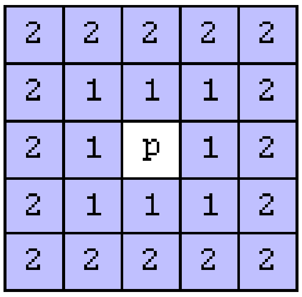


Figure (2.15) Distances of pixel *p* for co-occurrence matrix

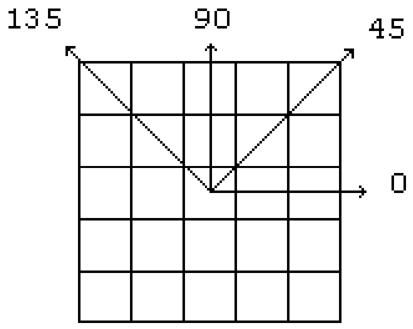


Figure (2.16) Directions for co-occurrence matrix.

In the following definition, the co-occurrence matrix is expressed as $P_{d,\theta}(i,j)$, in distance d and direction θ. For brevity, it may be expressed as *P(i,j)* [60].

$$P_{d,\theta}(i,j) = \#\left\{ ((x,y),(x^*,y^*)) \in D \times D \;\middle|\; d = \left\|(x,y),(x^*,y^*)\right\|, \right. \\ \left. \theta = \angle(x,y),(x^*,y^*),\ i = I(x,y), j = I(x^*,y^*) \right\} \quad , \text{.........} \ (2.24)$$

where, $P_{d,\theta}(i,j)$ is the co-occurrence matrix, # stands for the function "number of", (x,y) and (x^*,y^*) are valid image pixel coordinates, D is discrete gray scale image domain, d is the distance between two pixels, and $\angle$ is the direction of two pixels. In equation (2.24) some of the features of textured images could be obtained. However, two images with the same texture, but different in size may have different features vectors. To accomplish this computation, matrices need to be normalized by the size of images [13].

After these calculations are done, the matrix is reduced by dividing each value by normalization factors rendering a matrix whose sum is equal to 1. One co-occurrence matrix is created for each pair direction-distance considered by the texture analysis. The normalizing factors are given according to direction as [48]:

$$N_\theta = \begin{cases} 2N_y(N_x - 1) & \text{if } \theta = 0^o \\ 2(N_x-1)(N_y-1) & \text{if } \theta = 45^o \,\&\, 135^o \\ 2N_x(N_y-1) & \text{if } \theta = 90^o \end{cases} \quad , \quad \ldots\ldots\ldots\ldots \quad (2.25)$$

By knowing the normalizing factor $N_{\theta,}$ the co-occurrence matrix would be normalized as:

$$P_{d,\theta}(i,j) = \frac{1}{N_\theta} P_{d,\theta}(i,j) \quad , \quad \ldots\ldots\ldots\ldots \quad (2.26)$$

2.7.4 Texture Features Explanations

In this subsection, the intuitive descriptions of the features which describe textures are explored [24, 40]:

1. ***Energy:*** It describes the uniformity of the texture. When all the matrix elements are almost equal, i.e., when gray level intensities are very close to each other, the value of the energy is small. Thus, the higher the value of the energy, the more irregular the GLCM; Figure (2.17).
2. ***Entropy:*** It measures the randomness of the elements in the matrix. When all elements of the matrix are maximally random, entropy has its highest value. So, a homogeneous image has lower entropy than an inhomogeneous image. In fact, when energy gets higher, entropy should get lower. Entropy has its highest peak when the GLCM is uniform; Figure (2.17).
3. ***Inverse Difference Moment:*** It has a relatively high value when the high values of the matrix are near the main diagonal. This is because the squared difference $(i-j)^2$ is smaller near the main diagonal, which increases the value of $1/(1+(i-j)^2)$; Figure (2.18).
4. ***Inertia (contrast):*** It gives the opposite effect of the previous feature. When the high values of the matrix are further away from the main diagonal, the value of inertia becomes higher. So inertia and the inverse difference moment are measures for the distribution of gray-scales in the image; Figure (2.18).

5. ***Cluster shade and cluster prominence:*** They measure the skewness of the matrix, in other words the lack of symmetry. When cluster shade and cluster prominence are high, the image is not symmetric. In addition, when cluster prominence is low, there is a peak in the co-occurrence matrix around the mean values. For the image, this means that there is little variation in gray-scales.

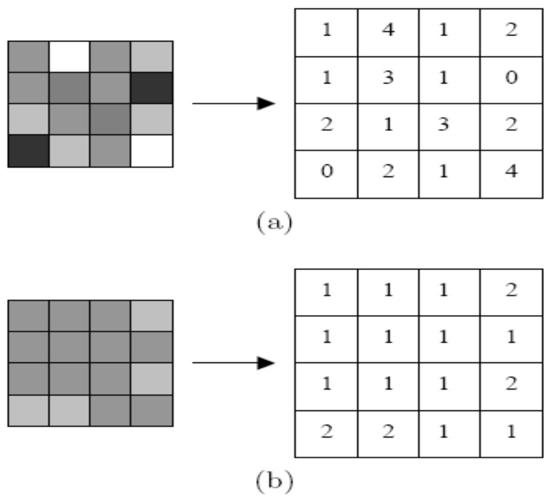


Figure (2.17) Energy and entropy functions; [40] (a) energy low and entropy high, and (b) energy high and entropy low .

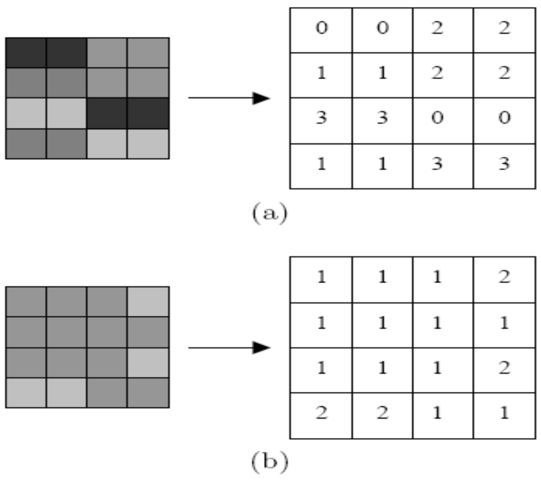


Figure (2.18) Contrast and inverse difference moment functions; [40] (a) contrast high and inverse difference moments low, and (b) contrast low and inverse difference moment high.

6. ***Correlation:*** It measures the correlation between the elements of the matrix. When correlation is high the image will be more complex than when correlation is low. Haralick's correlation is a measure of gray level linear dependence between the pixels at the specified positions relative to each other. Compared to normal correlation, Haralick's correlation reacts stronger to the complexity of an image. A high or a low correlation value leads to no immediate conclusion about the image.

2.7.5 Texture Features Descriptors

Having a co-occurrence matrix, different properties of the pixel distribution can be obtained by applying mathematical operations on the matrix values. These operations are called descriptors. Each descriptor is related to a particular visual feature about texture. Haralick et al., proposed a set of 14 second-order statistical functions [48]. In many applications, an appropriate subset of these 14 descriptors might be enough for an adequate representation of the neighborhood information of the pixels [37].

The mathematical definitions of the mostly used features are as follows [40, 57, 60, 91]:

$$Energy = \sum_{i,j} P(i,j)^2 \quad ,. \ldots\ldots \quad (2.27)$$

$$Entropy = -\sum_{i,j} P(i,j)\log P(i,j) \quad , \ldots\ldots \quad (2.28)$$

$$Inverse\ Difference\ Moment = \sum_{i,j} \frac{1}{1+(i-j)^2} P(i,j) \quad , \ldots\ldots \quad (2.29)$$

$$Inertia = \sum_{i,j} (i-j)^2 P(i,j) \quad , \ldots\ldots \quad (2.30)$$

$$Cluster\ Shade = \sum_{i,j} ((i-\mu_i)+(j-\mu_j))^3 P(i,j) \quad , \ldots\ldots \quad (2.31)$$

$$Cluster\ Prominence = \sum_{i,j} ((i-\mu_i)+(j-\mu_j))^4 P(i,j) \quad ,\ldots\ldots \quad (2.32)$$

$$Correlation = \sum_{i,j} \frac{(i-\mu_i)(j-\mu_j)P(i,j)}{\sigma_i \sigma_j} \quad , \ldots\ldots \quad (2.33)$$

$$Haralick\ Correlation = \frac{\sum_{i,j} (ij)P(i,j) - \mu_x\mu_y}{\sigma_x\sigma_y} \quad , \ldots\ldots\ldots\ldots\ldots \quad (2.34)$$

where:

P(i,j) is the (i,j)th entry in a co-occurence matrix $P_{d,\theta}$,

$\sum_{i}$ means $\sum_{i=0}^{M-1}$ where M is the number of rows,

$\sum_{j}$ means $\sum_{j=0}^{N-1}$ where N is the number of columns,

$\sum_{i,j}$ means $\sum_{i}\sum_{j}$,

μ_i is defined as : $\mu_i = \sum_{i} i \sum_{j} P(i,j)$,

μ_j is defined as : $\mu_j = \sum_{j} j \sum_{i} P(i,j)$,

σ_i is defined as: $\sigma_{\mathrm{i}} = \sum_{i} (i-\mu_i)^2 \sum_{j} P(i,j)$,

σ_j is defined as: $\sigma_{\mathrm{j}} = \sum_{j} (j-\mu_j)^2 \sum_{i} P(i,j)$,

μ_x, μ_y, σ_x , σ_y are the means and standard deviations of :

$C_x(i) = \sum_{\mathrm{j}} P(i,j)$ *and* $C_y(j) = \sum_{i} P(i,j)$ respectively.

Apart from the GLCM, texture contrast could be concluded by the following formula [3]:

$$Con = \frac{\sigma}{\alpha_4^{1/4}} \quad , \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.35)$$

where the kurtosis $\alpha_4 = \mu_4 / \sigma^4$, μ_4 is the fourth moment about the mean, and σ^2 is the variance. This formula can be used for both the entire image and a region of the image.

Chapter Three

Hierarchical Indexing and Image Matching

3.1 Introduction

This chapter is organized as follows: Indexing and retrieval approaches based on different indexing structures are discussed in section (3.2). Different clustering methods are introduced in section (3.3). Hierarchical clustering algorithms are explored in detail in section (3.4). Section (3.5) presents similarity measurements and image searching schemes. Finally; sections (3.6) and (3.7) express Querying and performance evaluation of CBIR systems respectively.

3.2 Indexing

As a result of advances in the Internet and new digital image sensor technologies, the volume of digital images produced by scientific, educational, medical, industrial, and other applications available to users has increased dramatically [41]. However, we cannot access or make use of the information in these huge image collections unless they are organized so as to allow efficient browsing, searching, and retrieval over all textual and image data [10].

To make content-based image retrieval truly scalable to large image databases, efficient indexing techniques need to be explored. Many recent applications such as image databases, medical databases, GIS and CAD/CAM require enhanced indexing for content-based image retrieval [88]. A continuing challenge facing current CBIR technology is that of efficiently retrieving the set of stored images most similar to a given query. Finding index structures which allow efficient searching of an image database is still an unsolved problem [37].

The data domains stored in traditional databases, such as numbers and small character strings, have the total ordering property. This property has led to the development of the current Database Management Systems (DBMS). However, when the data do not have this property, the traditional access methods can not be used [68].

3.2.1 Indexing Structures

Data embedded in multi-dimensional domains are examples of information that can not be directly sorted [70]. Complex data, such as images, video, sound, time series, and DNA strings, among others do not have implicit order property. That is, these data can not be sorted using only their raw information, and there is no direct way to create an access method to improve their retrieval by sequentially scanning all of them [87]. In this thesis the focus will be on images, where a sequential scan of the entire collection in the querying process is going to be avoided in order to speed up the retrieval process.

The goal of indexing is to create a compact summary of the database contents to provide an efficient mechanism for retrieval of the data. The summary data is based on features vectors. In content-based visual databases, all items (images or objects) are represented by pre-computed visual features. Hence, the key attribute for each image will be a features vector, which corresponds to a point in a multi-dimensional features space. Search will be based on similarities between the features vectors. Therefore, to achieve a fast and effective retrieval, an efficient indexing scheme is required [18].

Some CBIR systems provide a kind of an indexing scheme, which is a mechanism for efficiently accessing stored images. Indexing techniques used range from standard methods (such as signature-file access method and inverted-file access method), to multi-dimensional and high-dimensional indexes [92]. In addition to the above approaches, clustering and neural nets are also promising indexing techniques [7].

3.2.2 Multi-dimensional Indexing

The history of multi-dimensional indexing techniques can be traced back to the mid 1970s; when cell methods, quad-tree, and k-d tree were first introduced. However, their performances were far from satisfactory [10]. The majority of the data structures designed to store multi-dimensional data use the approach of dividing space into sections, called buckets or cells. Each bucket may be subdivided when it has too many entries, or the entire space may be redivided. By searching through each cell first, it is possible to determine which cells do, or do not, contain useful data, so that others may be searched thoroughly [12].

Many of the structures employ a hierarchical, or a tree structure, so that many branches of cells may be eliminated from the search very rapidly. Many well-known index mechanisms work well in low dimensionality (2-d, 3-d), but they cannot scale up to high dimensional space. In the context of high dimensional indexing, this phenomenon is called *'the curse of dimensionality'* [83].

The Balanced Tree (B-Tree) is such a structure, but is designed to only store one dimensional data. The multi-dimensional version of the B-Tree is R-tree and its variants R^+ tree and R^* tree [44]. Other multi-dimensional indexing methods include bucketing algorithm, priority k-d tree, K-D-B tree, linear quad tree, hB tree, and grid files [10].

Most of these methods have reasonable performance for a small number of dimensions (up to 20), but explore exponentially with the increasing of the dimensionality and eventually reduce to sequential searching [9, 10, 26]. On the other hand, several index structures exist for high dimensional data such as SS tree, TV tree, X tree, SR tree, and Pyramid-Technique. These methods also fall prey to what is called 'curse of dimensionality', although conceptually they can be extended to higher dimensionalities [88].

3.2.3 Dimensions Reduction

It has been shown that the main factor affecting the efficiency of a multi-dimensional access method is the intrinsic dimensionality of the dataset, that is, the number of uncorrelated dimensions. For instance, regarding histograms, indexing could be done more efficiently if the correlations between close bins were used. Many attempts have been made to find more concise representations of histograms using dimensionality reduction techniques. All these attempts lead to histogram representations with a reduced number of dimensions, but with a pre-defined number of dimensions [87].

Because the features vectors of images tend to have high dimensionality and therefore are not well suited to traditional indexing structures, dimensions reduction is usually used before setting up an efficient indexing scheme. This will provide some useful properties (such as the ability to locate the most important sub-space), but the features properties that are important for identifying the pattern similarity may be destroyed during the blind dimensionality reduction [74].

One attempt to solve the indexing problems is to use hierarchical indexing scheme (clustering). In addition to benefiting indexing, it provides users a useful tool to browse images of each cluster. Yet many features are of high dimensionality and structures like the R-Tree tend to break down very rapidly when the dimensionality of data becomes large [41].

Principle Component Analysis (PCA) will reduce the number of dimensions that a feature has, with a little loss of information, and results in a low dimensional feature. PCA must be performed on all data and may not be useful when a database undergoes many creates, updates, or deletions. An alternative approach to indexing is to identify clusters in the features space, or to partition the features space and label the features [44].

3.2.4 Indexing and Queries

CBIR queries are posed in a fuzzy fashion. The user is typically interested in results according to similarity rather than equality. This requirement influences the indexing scheme, the methods of features comparison, and the means by which queries are solicited from the user [18].

In a typical situation, see Figure (3.1), all the images in the database are processed to extract the selected features that represent the contents of the images. This is usually done automatically once when the images are entered into the database. This process assigns to each image a set of identifying descriptors which will be used by the system later in the matching phase to retrieve relevant images. The descriptors are stored and indexed in the database, ideally in a data structure that allows efficient retrieval in the later phase [7].

Next a query is posted in the matching phase. Using the same procedures that were applied to the image database the features for the query image are extracted. Image retrieval is then performed by a matching engine, which compares the features or the descriptors of the query image with those of the images in the database. The matching mechanism implements the retrieval model adopted according to the selected metric, or similarity measure.

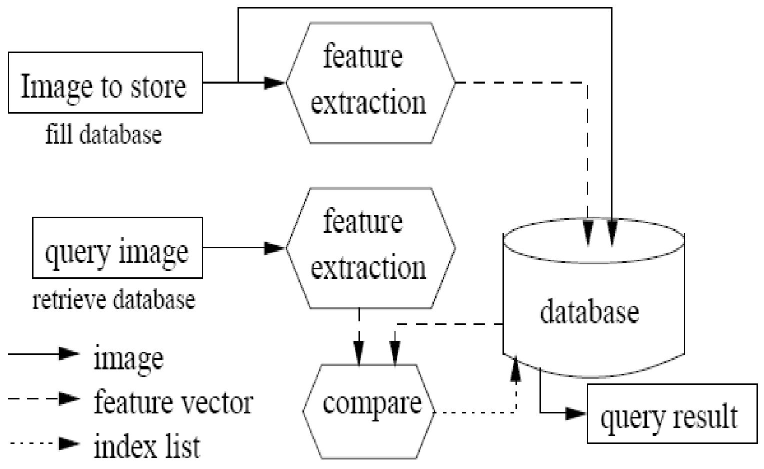


Figure (3.1) General architecture of a CBIR system [50].

The images in the database are then ranked according to their similarity with the query and the highest ranking images are retrieved. Efficiently describing the visual information of images and measuring the similarity between images described by such pre-computed features are the two important steps in content-based image retrieval [10].

3.2.5 Efficient Features Indexing

Whilst individual features in a single media can be very small, and easy to compare, comparing across a whole database of images will take considerable time. Features indexing allows for rapid retrieval of similar features without having to look through all signatures.

Features in a signature are normally of the same form; hence, a signature can simply be a composite representation of them. If this representation is a set of values, the signature can be represented as objects in Euclidean space such as points, vectors or volumes. Signature similarity may be calculated as the distance between two points, or the distance between a point and the center of a volume. Other tests may be required, such as whether a point is inside a volume, or whether a volume is inside a volume [44].

Finding index structures that allow efficient searching of an image database is still an unsolved problem. None of the index structures proposed for text retrieval has proved applicable to the image retrieval problem, since CBIR techniques are based on a fundamentally different model of data [72]. Stored images are typically characterized by fixed-length real-valued multi-component features vectors. Each image has a value for every feature in the database. In this case, searching consists of calculating the similarity between features vectors from query and stored images, a process of numerical computation [30].

So far the most used image indexing approach is multidimensional indexing, but the overheads of using these complex index structures are considerable. An alternative approach, which seems to offer better prospects of success, is the use of similarity clustering of images, allowing hierarchical access for retrieval and providing a way of browsing [17].

3.3 Clustering

Clustering can be defined as a process of grouping data into classes or clusters. The objects in a cluster have high similarity in comparison to each other, but they are dissimilar to the objects in other clusters [43]. In this work, the clustering procedure is developed for the images stored in the image database. The clustering is based on the image content that is described using features presented in chapter two. Hence, the clusters are formed in the features space. The selection of the clustering method is an essential point in the clustering procedure. Several methods and algorithms for clustering have been developed.

In general, there are two kinds of clustering methods. One is a hierarchical method, and the other is a non-hierarchical method. The hierarchical method (which is adopted in this study) is divided into two kinds of methods; agglomerative and divisive. In non-hierarchical clustering methods, the initial value and the number of clusters must be predefined. Non-hierarchical clustering method shows good performance for fixed number of clusters. Nevertheless, a flexible clustering method is needed for information retrieval [90].

Perhaps the most common clustering methods are k-means and k-medoids. The clustering algorithms used for information retrieval can be classified into two categories; static clustering and dynamic clustering. In static clustering, all the objects are first clustered (indexed) before any search can be performed. Whenever a new object is presented, the indexing structure needs a total reorganization to support later searches [43].

In dynamic clustering, search can be interlaced with indexing. When a new object is presented, the indexing structure will grow dynamically, no periodic reorganization is needed. Due to the high frequency of data update in VIR systems, static clustering is unacceptable [46].

3.3.1 Criteria for Choosing a Suitable Clustering Method

Image clustering groups a given set of unlabeled images into meaningful clusters according to the image content without a priori knowledge. Typical clustering techniques include hierarchical clustering algorithms, and partitional algorithms [92].

When choosing a suitable algorithm, the pros and cons of each method are weighted separately. A list of major factors describing a clustering algorithm was made. This is considered to be optimal for this research and can help us in choosing a suitable one. These factors are [69, 70, 84]:

1. *Scalability:* The ability to deal with large data sets. Clustering algorithm should be highly scalable.
2. *Insensitivity to the order of input:* Some algorithms are sensitive to the order of input data. For example when the same data set is presented in a different order, algorithms like these may generate dramatically different clusters. It is important for us to select an algorithm that is insensitive to the order of the input.
3. *High dimensionality:* A database can contain several dimensions or attributes. It is a challenge to cluster data objects in high-dimensional space, because that kind of data can be very sparse and highly skewed.
4. *Reclustering ability:* When new data is added to the data set, clustering algorithm should have the ability to recluster the clusters in a new version.
5. *Requirements for domain knowledge to determine input parameters*: The use of parameters in cluster analysis is important, such as the number of desired clusters. Sometimes it is necessary to be able to decide the final number of our clusters.
6. *Computational complexity:* The computational complexity is required to be as low as possible to save clustering time.

3.3.2 Clustering Techniques

There are two promising techniques towards solving indexing problem; clustering and Neural Nets. These techniques have three advantages. They are: (1) dynamic structure, (2) capability of handling high dimensional data, and (3) the potential to deal with non-Euclidean similarity measures [7].

A dynamic hierarchical clustering technique was developed to be scalable to high dimensionality required by MIS systems. The problem of clustering consists of partitioning N points in a metric space M into k clusters based on some criterion [87]. By storing similar objects together, retrieval can be processed more efficiently by reducing the number of objects to be accessed to return the results [46]. Due to these reasons, hierarchical agglomerative clustering (HAC) are frequently used in information retrieval field [90].

Since clustering is adopted in this work, a brief description of the most common and well-known clustering algorithms is provided in this section. In all proposed methods each sample is described by features vector, in which each feature is represented by a real number.

1. K-means

Given a population of samples and a number K of the desired groups or classes, the final goal of this algorithm is the partitioning of the given population in K different classes. Each class has a center which is the mean position of all the samples in that class, and each sample is in the class whose center is closest to. The inputs of the algorithm are: the number of patterns, the needed number of classes K, and a vector of K random means. One of the main problems of this algorithm is that the classification result depends on the initialization of the algorithm. If the K initial random means are changed, it is possible also that the final result will be different [68].

2. Hierarchical Clustering

Hierarchical clustering methods can be further classified into agglomerative (bottom-up) and divisive (top-down) hierarchical clustering. Agglomerative hierarchical methods start with each object in one cluster and in each step the closest pair of clusters are merged until there is only one cluster left or a certain termination condition is fulfilled [69].

Divisive hierarchical methods start with one big cluster containing all the objects. In each step, a cluster is subdivided until a certain termination condition is satisfied, or the final number of wanted clusters is reached [84]. In this thesis, the opposite, i.e. an agglomerative hierarchical method is proposed; therefore, hierarchical clustering will be explored in detail in a separate section.

3.4 Hierarchical Clustering

Given a similarity matrix, Hierarchical Cluster Analysis (HCA) organizes a set of elements into similar units. This method starts from the elements set to build a tree. Before the procedure begins, all elements are considered as separate clusters. The tree is formed by successively joining the most similar pairs of elements into a new cluster. The way clusters are merged and the similarity matrix is updated depends on the type of the algorithm [2].

Cluster analysis does not use category labels that tag objects with prior identifiers. In other words, we don't have prior information about cluster seeds or representatives. The objective of cluster analysis is simply to find a convenient and valid organization (i.e. grouping) of the data. The main purpose of clustering is to reduce the size and complexity of the data set. Data reduction is accomplished by replacing the coordinates of each point in a cluster with the coordinates of that cluster's reference point (cluster's seed or representative). Clustered data require considerably less storage space and can be manipulated more quickly than the original data [13].

3.4.1 Hierarchical Agglomerative Clustering (HAC)

Hierarchical clustering is a bottom-up clustering method where clusters have sub-clusters, which in turn have sub-clusters, and so on. The classic example of this kind of clustering is species taxonomy. Agglomerative hierarchical clustering starts with every single sample in a single cluster. Then, in each successive iteration, it merges the closest pair of clusters by satisfying some similarity criteria, until all of the data is in one single cluster [70].

The hierarchy within the final cluster has the following properties; clusters generated in early stages are nested in those generated in later stages, and clusters with different sizes in the tree can be valuable for discovery. After the input samples are provided, each pattern is assigned to a separate cluster. Firstly, all pair-wise distances between clusters are evaluated, and then a distance matrix is constructed using the distance values. After that, a loop starts until a termination condition is met. During each iteration of this loop, the pair of clusters with the shortest distance is looked for. The selected pair is removed from the matrix and the two clusters are merged producing one new cluster. The distance among the new cluster and all the others are computed and the matrix is updated [69].

3.4.2 HAC Algorithm

The major steps in agglomerative clustering are contained in the following algorithm [69]:

```
Algorithm: Basic Agglomerative Clustering
          1. Let $k = n$ and $X_i = \{ x_i \}$ , $i = 1, . . . , n$.
   Loop:  2. If $k <= c$ then stop.
          3. Find the nearest pair of distinct clusters, say $X_i$ and $X_j$ ,
          4. Merge $X_i$ and $X_j$ into new cluster, say $X_{n+1}$, delete $X_i$ and $X_j$ ,
             decrement k by one.
          5. Go to Loop.
```

where X_i represents a cluster with a single element x_i, n is the number of items in the whole data set, c is the desired number of clusters.

In the original version of the algorithm, the loops end when the distance matrix is composed of only one element. The most natural representation of hierarchical clustering is a corresponding tree, called dendrogram, which shows how the samples are grouped. Figure (3.2) shows a dendrogram for a simple problem involving six samples. Level 1 shows the six samples as singleton clusters. At level 2 samples x_1 and x_2 have been grouped to form a cluster, and they stay together at all subsequent levels.

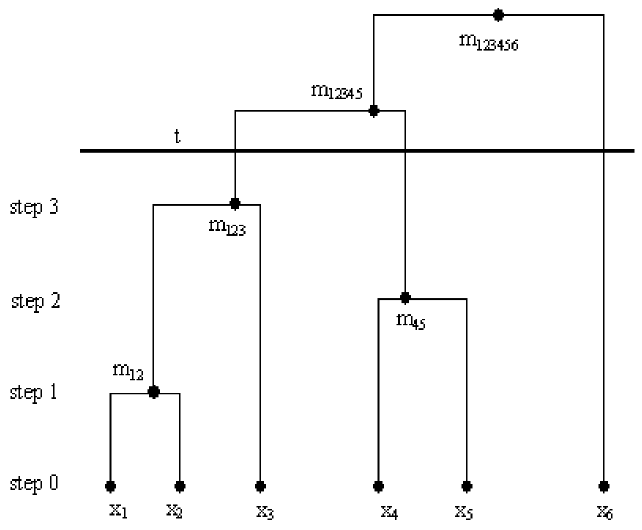


Figure (3.2) A Dendrogram for a HAC of patterns (x_1,…, x_6). [89]

At each step the two nearest patterns are merged and represented by their centroid. After *i* steps, a stop condition *t* is reached and the clustering algorithm terminates. Three clusters are produced: {x_1, x_2, x_3}, {x_4, x_5}, {x_6}. The dendrogram is usually drawn to scale to show the similarity among the grouped clusters. Another representation for hierarchical clustering is based on sets, in which each level is depicted by sets representing clusters that may contain other sub-clusters [89].

3.5 Similarity Measurements and Image Searching

There are different clustering methods, which can be defined according to measurement of distances between clusters. Based on the calculation of similarity between the non-singletons clusters, varieties of hierarchical agglomerative techniques have been proposed. Single-link, complete-link, group average-link, and mean (centroid) clustering are well known in measuring inter-cluster distances [69]. The use of different distance metrics for measuring distances between clusters may generate different results.

3.5.1 Clusters Distance Computation

The more common metrics used for the computation of clusters distance produces different variations. Some details about these algorithms are given below:

1. Single-link

In the single-link (also called as nearest neighbor, or minimum algorithm), the similarity between two clusters is the minimum similarity of all pairs of elements which are in different clusters. Let X_i and X_j, respectively, be the ith and the jth clusters; the distance function used for measuring the distance between those two clusters is [69, 70]:

$$D_{\min}(X_i, X_j) = \min_{x \in X_i, x' \in X_j} \|x - x'\| \quad , \text{ ....} \quad (3.1)$$

When $D_{\min}(.,.)$ is used to measure the distance between subsets, the nearest-neighbor nodes determine the nearest subsets. In other words, the merging of X_i and X_j corresponds to the fusion of two clusters having the two closest elements. This kind of metrics roughly tends to produce “elongate” clusters.

2. Complete-link

In the complete-link (also called as farthest neighbor, or maximum algorithm**)**, the similarity between two clusters is the minimum similarity of all pairs of elements which are in different clusters. In this case the distance function used for measuring the distance between two clusters is [69, 70]:

$$D_{max}(X_i, X_j) = \max_{x \in X_i, x' \in X_j} \|x - x'\| \quad , \text{ ....} \quad (3.2)$$

When $D_{\max}(.,.)$ is used to measure the distance between subsets, the distance is determined by the most distant nodes in the two clusters. This kind of metrics tends to merge the clusters in order to minimize the increment of the diameter of the clusters themselves, thus producing compact and roughly equal in size clusters.

Most of the algorithms discussed above work implicitly or explicitly with the $n \times n$ similarity matrix such that the (i, j) element of the matrix represents the similarity between ith and jth data items [26].

3. Other Metrics

The minimum and maximum measures represent two extremes in measuring the distance between clusters, and in some situations can lead to minor results. The use of averaging is a possible way to improve the quality of clustering. The most commonly used distance functions are the following [89]:

$$D_{avg}(X_i, X_j) = \frac{1}{|X_i| . |X_j|} \sum_{x \in X_i} \sum_{x' \in X_j} \|x - x'\| \quad , \ldots\ldots\ldots\ldots\ldots\ldots \quad (3.3)$$

$$D_{mean}(X_i, X_j) = \|mean(X_i) - mean(X_j)\| \quad , \ldots\ldots\ldots\ldots\ldots\ldots \quad (3.4)$$

In Group-Average-link clustering the similarity between two clusters is the mean similarity of all pairs of singletons which are in different cluster, while in centroid clustering, the similarity is obtained by measuring the centroid distance between any pair of clusters [82]. This last measure (i.e. mean or centroid) is intuitively the most common one; therefore, it is adopted in this thesis.

3.5.2 Exact Match versus Similarity Match

In traditional databases, user often queries on the exact item. However, in multimedia domain, queries for exact match item are rare and impractical. In multimedia applications, comparisons often are not based on exact match, but rather they are based on similarity comparison. In similarity searching, a vast amount of results will be obtained. In order to provide useful solution to users query, filtering and ranking mechanism are applied to the result [85].

There are many applications of nearest neighbor search where an approximate answer is good enough. It is not necessary to insist on the exact answer; instead, determining an approximate one may suffice [86].

3.5.3 Heuristic Search versus Exhaustive Search

As multimedia data is large in size, exhaustive search in multimedia domain is very expensive. Experiments suggest that exhaustive search and systematic search still fail to give optimal performance in some cases. Sub-optimal searching algorithms will be a good alternative to the problem. These algorithms can find nearly optimal solution in a limited time [85, 67]. In this thesis, sub-optimal searching algorithms based on heuristic search in the image retrieval domain are adopted.

3.5.4 Feature-based Similarity

A similarity search problem involves a collection of objects (e.g., documents, images) that are characterized by a collection of relevant features and represented as points in a high-dimensional attribute space. Given queries in the form of points in this space, we need to find the nearest (most similar) object to the query. The particularly interesting and well-studied case is the d-dimensional Euclidean space. The problem is of major importance to a variety of applications; some examples are: data compression, databases / data mining, information retrieval, image / video databases, machine learning, pattern recognition, and statistics / data analysis [54].

Typically, the features of the objects of interest are represented as points in a certain space, and a distance metric is used to measure the similarity of objects. Then, the basic problem is to perform indexing or similarity searching for query objects. The number of features (i.e., the dimensionality) ranges anywhere from tens to thousands. For example, in multimedia applications such as IBM's QBIC (Query by Image Content) the number of features could be several hundreds [86].

Image similarity is usually determined by computing a distance measure between the query and the appropriate features vectors in the index structure. Similar images are ranked according to distance. Thresholding may be used to reduce the number of similar images presented to the user [18].

Feature-based similarity queries employ the notion of distance between two points P and Q in the data space. Several metrics to define distance can be found. The most widely used metrics are the Manhattan metric (also known as L1 metric, rectilinear metric), the Euclidean metric (L2 metric). The mathematical definition of these two metrics is described as [7, 83]:

$$D_p(P,Q) = \left(\sum_{i=0}^{n-1} \left| P_i - Q_i \right|^p \right)^{1/p} \text{, ...} \qquad (3.5)$$

where p is a positive integer, P and Q are data points in high dimensional space (representing images), each with a features vector of n items. This distance metric represents the Manhattan metric (L_1) when p = 1, and Euclidean metric (L_2) when p = 2.

3.6 Query by Example

The most appealing paradigm in many ways is query-by-example. It allows the user to formulate a query by providing an example image. The system converts the example image into an internal representation of features. Images stored in the database with similar features are then searched. Query by example can be further classified into query by external image example (if the query image is not in the database), and query by internal image example (if otherwise) [36, 52].

The main advantage of query by example is that the user is not required to provide an explicit description of the target, which is instead computed by the system. It is suitable for applications where the target is an image of the same object or set of objects under different viewing conditions [41]. Virtually, all current CBIR systems offer query-by-example searching, where users submit a query image and the system retrieves and displays thumbnails of say the 20 closest-matching images in the database. Several alternative query formulation methods have been proposed [37]. The two most frequently used similarity queries are k-Nearest Neighbor (or k-NN query), and range query [87].

3.6.1 K-Nearest Neighbor Query (k-NN)

Searching the nearest neighbor is an important problem in high-dimensional indexing. Given a query point Q, a distance metric M, and a positive integer K; this method will find the K most similar objects in the database with respect to the given query point [92]. Usually a threshold T is also defined indicating the maximum distance error allowed in the final result. Therefore, the k-Nearest Neighbor query becomes: *finding the k most similar objects* within a given distance T to the query point [83].

3.6.2 Range Query

This kind of a query searches for all the objects whose distance to the query object center is less or equal to the query radius [87]. A range query search in an image database is a traversal algorithm of an indexing structure to find the node which contains a set of features vectors that are within fixed similarity bound, say t, from the query image [53]. Since the features vectors form a sparse multidimensional features space, it is natural to assume that there exists an under-lying distribution of these vectors. Usually, the distribution is not uniform. As a result, we may group features vectors together that are generally retrieved together in response to a request query. This leads to clustering of features vectors [45].

3.7 Performance Evaluation of a Retrieval System

Once a content-based image retrieval application has been developed, the next crucial problem is how to evaluate its performance, both retrieval performance and complexity. To evaluate the performance of a retrieval system, two measurements; namely, Precision (P) and Recall (R) are borrowed from traditional information retrieval [10]. Precision and recall metrics are used to evaluate the performance of a retrieval system. The first question is about the accuracy (precision) of the search, while the second is about the completeness (recall) of the search. The precision of the retrieval is defined as the fraction of the retrieved images that are indeed relevant for the query, while the recall is the fraction of relevant images that is returned by the query [94]:

$$\text{Precision} = \frac{\text{The number of relevant retrieved images}}{\text{The number of retrieved images}} \quad , \ldots. \quad (3.6)$$

$$\text{Recall} = \frac{\text{The number of relevant retrieved images}}{\text{The total number of relevant images}} \quad , \ldots\ldots.. \quad (3.7)$$

Recall quantifies the ability of the system to retrieve useful images, while precision measures the ability to reject useless ones [25]. Usually, a tradeoff must be made between these two measures since improving one will sacrifice the other. In typical retrieval systems, recall tends to increase as the number of retrieved items increases; while at the same time the precision is likely to decrease [40].

After collecting images, the first step in evaluating performance of CBIR application is to define a set of queries and their ground truth based on the input image database. A very common way to generate ground truth from a query image is by adding noise, down-scaling, or up-scaling the query image. When the image database is collected and the queries and their ground truth are selected, the query images are presented one by one to the search engine of the CBIR application. The retrieved results are then compared to the ground truth of the corresponding query image [94].

In order to evaluate the retrieval performance of an image database, it is necessary to verify a test set and a ground truth, which means that it is known which images match with the image that is queried in the database. For quality assurance, it is important to have a representative test database, which can be used to measure the performance. This means that the test set will be compared with the complete database. The optimal situation for retrieval is that the relevant images are retrieved, and that there are no missed relevant images [35].

When the number of retrieved images is increased, precision is decreased, and recall is increased. Higher precision indicates less unwanted images; conversely, higher recall indicates less missing images. P & R contingency is shown in Figure (3.3).

	Relevant	Not Relevant
Retrieved	A	B
Not Retrieved	C	D

Relevant = A + C
Retrieved = A + B
Collection size = A + B + C + D

Precision = A/(A+B)
Recall = A/(A+C)

Fallout/false alarm = B/(B+D)
Miss = C/(A+C) = 1 - Recall

Figure (3.3) Precision and recall contingency table

Theoretically, P and R do not depend on each other. Practically, high recall is achieved at the expense of precision. Also, high precision is achieved at the expense of recall. P is equal to 0 when none of the retrieved images is relevant, while P is equal to 1 only when every retrieved image is relevant. Because of this inverse relationship between precision and recall, improving recall almost certainly would lower precision. Depending on the application, one may want a higher precision or a higher recall [10].

Recall measures the ability of the search to find all of the relevant items in the database. Precision evaluates the correlation of the query to the database; it is an indirect measure of the completeness of indexing algorithm. Instead of using one query to derive precision and recall, multiple queries are used to gain averaged values that give a better indication on every retrieval scheme performance [41]. This relation is depicted in Figure (3.4).

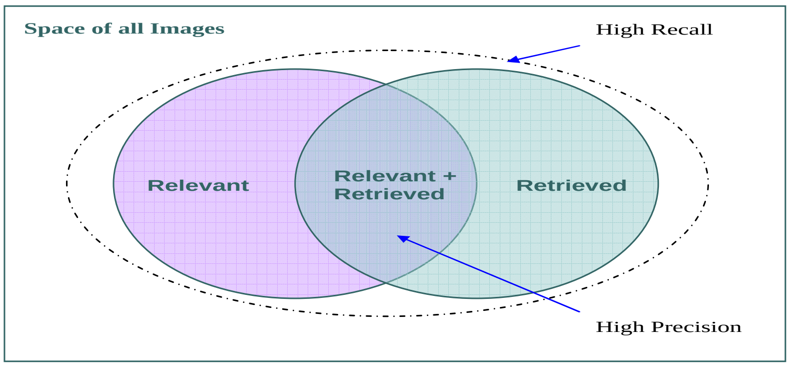


Figure (3.4) Retrieved versus relevant images

Given the relevant images to the following two queries, precision and recall values are shown:

- Relevant: f_3 f_5 f_9 f_{25} f_{39} f_{44} f_{56} f_{71} f_{123} f_{89}
- query1: f_{123}√ f_{84}× f_{56}√
  - Precision: 66% (2/3)
  - Recall: 20% (2/10)
- query2: f_{123}√ f_{84}× f_{56}√ f_6× f_8× f_9√
  - Precision: 50% (3/6)
  - Recall: 30% (3/10)

3.7.1 Mean Average Precision (MAP)

Average precision at selected numbers of retrieved images (usually number of relevant images) can be calculated over many queries. Subsequently the mean of these averages is taken as an excellent concise hint on retrieval performance. This metric is expressed by the following equation [73]:

$$MAP = \frac{1}{N} \sum_{Q_i} \frac{1}{n_i} \sum_{j=1..n_i} \frac{rel_j}{ret_j} \quad , \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.8)$$

where:

N = number of test queries,

Q_i = *Query i* of the N queries,

n_i = number of total *relevant images* for Q_i ,

rel_j=number of *relevant images* at step *j, and*

ret_j=number of *retrieved images* at step *j.*

3.7.2 F-Measure

It is a harmonic weighted mean that combines precision and recall in a single number [74]:

$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R} \quad , \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.9)$$

where β represents the relative importance of precision and recall:

$\beta = 1$, precision & recall have the same importance,

$\beta > 1$, precision is favored, or

$\beta < 1$, recall is favored.

By choosing $\beta = 1$, equation (3.9) would be:

$$F_1 = \frac{2PR}{P+R} \quad , \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.10)$$

This equation is used as an F-measure in the retrieval performance assessment for the developed system. An arithmetic mean of Precision (P) and Recall (R) does not capture the fact that a *(50% P, 50% R)* system is often considered better than an *(80% P, 20% R)* system:

if P = 50% and R = 50% then arithmetic mean= 50%,

if P = 10% and R= 100% then arithmetic mean = 55% (not a good indicator).

On the other hand, F-measure is high only when both P&R are high:

if P = 50% and R = 50% then F-measure = 50%,

if P = 10% and R = 100% then F-measure = 18.2%.

Chapter Four

The Content-Based Image Retrieval System

4.1 Introduction

This chapter is organized as follows: the whole retrieval system is outlined in the next section. Color models conversions are explained in section (4.3). Section (4.4) discusses in detail some of the features extractions methods. The topic features vector normalization is submitted in section (4.5). Hierarchical indexing is described in section (4.6). Section (4.7) introduces query processing techniques. Finally, retrieval methodology based on combined features is summarized in section (4.8).

4.2 The Retrieval System Broad Outline

The fundamental idea of this work is to generate automatically image descriptions directly from the image content by analyzing the content of the image. Low-level features do play an important role, and in some sense are the bottleneck for the development and application of CBIR techniques. A very basic issue in designing a CBIR system is to select the most effective image features to represent image contents.

In order to characterize the color content of an image different color spaces have been used with different quantization schemes. In addition, cumulative histogram and color moments are, also, utilized. Within the statistical approach to the problem of texture analysis the Gray Level Co-occurrence Matrix (GLCM) and correlation histogram (correlogram) were implemented. Different combinations of the above features extraction methods are experimented.

This project focuses on a representation for computing global similarity. That is, the task is to find images that, as a whole, appear visually similar. Query is made upon images of homogeneous color/texture that do not require segmentation.

The similarity between two features vectors can be concluded through using some distance measurement.

To achieve the scalability of search to large databases, it must be ensured that the search time does not increase linearly with the database size. An effective indexing method was designed to accelerate similarity search task. Images are grouped into clusters beforehand based on the content, so that at the time of the query, only the relevant set of clusters needs to be examined.

An overview of the developed system architecture, in this research, is presented in Figure (4.1). The steps of the involved processes are discussed in the following sections. Given a set of images, as a first step, these image objects are represented by their corresponding features vectors through a set of transformations and normalizations. Features extraction is a crucial step in the whole retrieval process. Several extraction methods were adopted in this stage including Gray Level Co-occurrence Matrix (GLCM), correlation histogram (correlogram), cumulative histogram, and moments. In addition to applying each of these methods separately, different combinations of them were also experimented.

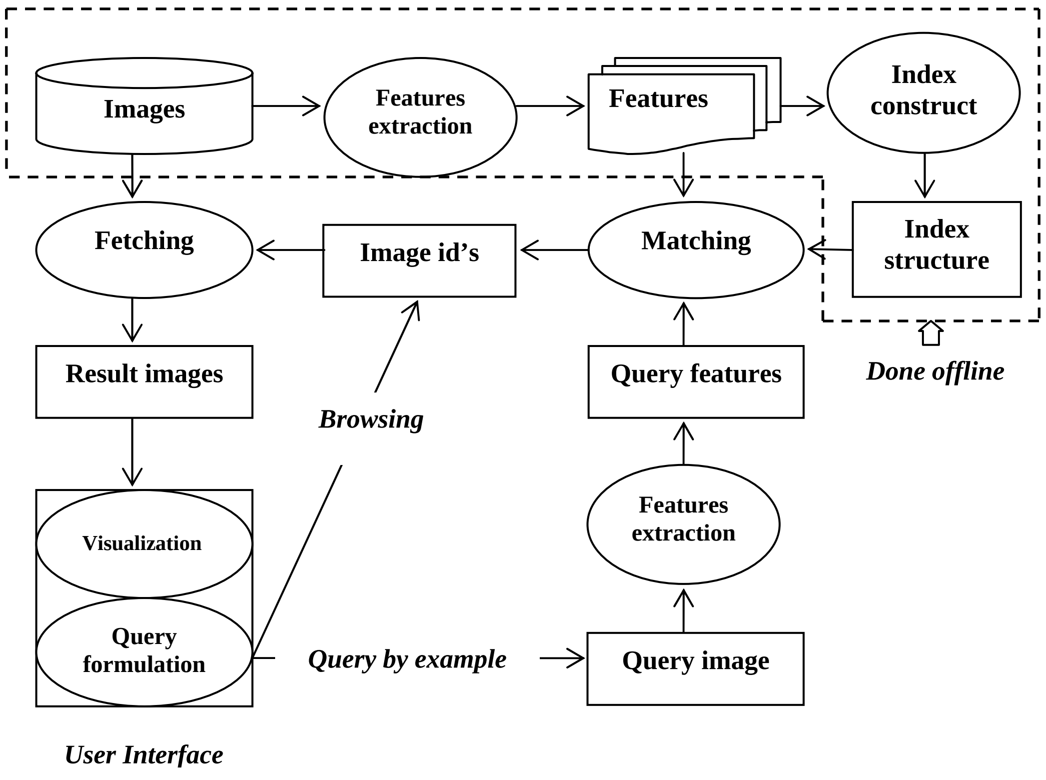


Figure (4.1) The Content-Based Image Retrieval System

In the next step, the features vectors are indexed through a process called index construction using a certain kind of hierarchical indexing methods. Some enhanced searching algorithms for specific query types were, also, designed to search through this index structure. When a query image (which is a user specified image) is used as an input, it is transformed to become a query point and passed to the index.

The index is used to locate similar images to query image by comparing through a distance measurement. The final results of the stage of searching the index structure are visualized to user. Indexed groups of images (clusters) could be explored separately.

4.3 Color Models Conversions

Prior to any analyses to be performed upon image data, it should be known in which color domain (model) the analysis work will proceed. Image is read first from a Bitmap file as an RGB array of records. Each record consists of R, G, and B pixel components. When image is processed in a color model rather than RGB model, it should be converted first to the specified color model. General issues concerning different color models are given in subsection (2.3.2). Munsell HVC, HSV, HLS, L*a*b*, Gray, and RGB color models have been used and compared for different extracted features. In addition, 9-colors and 11-colors categories that are not a color models, rather, they are a human-based segmentation of some standard color spaces are also used. 9-colors is based on HSV model while 11-colors is based on Munsell HVC model. They are described in article 6 under subsection (2.3.2).

Each color model was implemented by coding the transformation equations listed in its relevant article under subsection (2.3.2). HSV and HLS were implemented by using the simplified formula where no trigonometric functions are used. This may improve performance [65]; as described in Pseudo code list (4.1). HLS model is computed in a similar way for Hue component, but Luminance (Value), and Saturation are computed with a little difference.

Pseudo code list (4.1) HSV color model computation routine

```
Input
      RGBarr: Image array in RGB data.
      W, H: Image Width and Height
Output
      HSVarr: Image array in HSV data
Variables
      R,G,B,H,S,V: pixel values
      i,j: image array indices
      min, max: minimum or maximum components values
Procedure
      For all i, j do      {where i = 0..W-1,      j = 0..H-1}
          R← RGBarr(i,j).R
          G← RGBarr(i,j).G
          B← RGBarr(i,j).B
          { Hue computation}
          If min(R,G,B)=B then
                    $H \leftarrow \frac{G-B}{3(R+G-2B)}$
          ElseIf min(R,G,B)=R then
                    $H \leftarrow \frac{B-R}{3(G+B-2R)}+\frac{1}{3}$
          ElseIf min(R,G,B)=G then
                    $H \leftarrow \frac{R-G}{3(R+B-2G)}+\frac{2}{3}$
          { S and V computation }
          min← min(R,G,B)
          max← max(R,G,B)
          V← max
          If max <>0 then
                $S \leftarrow \frac{\max-\min}{\max}$
          Else
                $S \leftarrow 0$
      End For
End
```

4.4 Features Extraction Methods

Features transformation is a mapping from the space of image observations (usually image pixels) to a features space that has better properties for the retrieval operation. The features are usually composed into a multidimensional features vector. If the features are properly chosen, each class of images forms a cluster in the features space. Several features extraction routines were built. They are described according to the utilized color model and the type of the extracted features. Table (4.1) outlines the features extraction routines corresponded to the color models that are applied in this research.

Table (4.1) Color models features extraction routines

COLOR MODEL	FEATURES EXTRACTION ROUTINE
HVC color model	HVC features extraction routine
HSV color model	HSV features extraction routine
HLS color model	HLS features extraction routine
LAB color model	LAB features extraction routine
11 colors category	11 colors features extraction routine
9 colors category	9 colors features extraction routine
RGB color model	RGB features extraction routine
Gray color model	Gray features extraction routine

A choice parameter (representing the selected color model which will be used to extract the selected features) is supplied by the user. Each features extraction routine would result in a features vector array containing the extracted features.

4.4.1 Gray Level Co-occurrence Matrix

In the current subsection, a short description for Gray Level Co-occurrence Matrices (GLCM) is given. A set of well-known statistical textural features are based on GLCM. These matrices contain the information about gray levels (intensities) of pixels and their neighbors at fixed distance and orientation. The idea is to scan the image and keep track of gray levels of each of two pixels separated within a fixed distance d and direction θ. The co-occurrence “matrices and features” routine is presented in Pseudo code list (4.2).

Pseudo code list (4.2) Co-occurrence “matrices and features” routine

```
Input
    LumArr :  Luminance color  channel
Output
    FeatuVec: CLGM Features vector array
Variables
    QuantizedLuminanceArr: Two dimensional array
    Ng: Integer number representing selected quantized levels
Procedure
    OneChannelQuantize(LumArr, Ng, QuantizedLuminanceArr )  {send LumArr to
            the one channel quantization routine to be quantized to Ng levels }
    Co-occurrenceMatrix(QuantizedLuminanceArr, Ng, ar0, ar45,ar90,ar135) {send
            quantized Luminance array to the co-occurrence Matrices computation
            routine}
    Co-occurrenceFeaturesExtraction(ar0, ar45, ar90, ar135, FeatuVec) { send the four
            GLCM to the features extraction routine }
End
```

This approach explores the gray level spatial dependencies of the tested texture. As a first step, the co-occurrence matrix is counted for certain orientation and inter-distance between image pixels, and then some meaningful statistical features are extracted from the co-occurrence matrix as textural features. The features vector for a texture is constructed from many co-occurrence matrices corresponded to different orientations and distances. Using one distance or one direction is generally not enough to describe textural features. So, more than one direction and distance should be taken in the calculations. It is common to use four directions; two are oriented horizontally and vertically, and the other two are for diagonals. Most of the researchers use four directions and one of the four distances {1, 2, 3, or 4}.

In this study, four co-occurrence matrices have been used; one matrix for every direction. Only single value for distance *(d)* is chosen among the four distances. With this *(d),* the four matrices are computed, and each test image was represented by these four matrices.

In the above routine, there are three subroutine calls concerning the full implementation of the GLCM; *one channel quantization, co-occurrence matrices, and co-occurrence features extraction*. They are explained in the following three articles, respectively.

1. Intensity Slicing (One channel quantization)

Each co-occurrence matrix is (256×256) elements in size; assuming that images are in 256 gray levels. Hence, each matrix requires a large memory to store and it is a time consuming task to produce this matrix. So, to handle this problem, the images are quantized to a lower gray-scale; say 18. The co-occurrence matrices are, then, determined for this new image instead of the original one. This gray-scale reduction will reduce the computational complexity of the involved work with GLCM of (18×18) elements. The conducted experiments showed that the conversion of the image from 256 gray-scales to a lower rough scale, does not affect the texture query results (details are given in chapter five). The steps of this method are illustrated in Pseudo code list (4.3) which is based upon intensity slicing that is illustrated in Figure (2.10).

Pseudo code list (4.3) One channel quantization routine

```
Input
    Arr: Image Array to be quantized
    Ng: number of quantization levels
Output
QuanArr: Quantized Image Array
Variables
    i, j: Image array indices
    sum: summation
    mean: mean pixel value
    std: pixel value standard deviation
    stp: step level
    lmin: quantized minimum pixel value
    lmax: quantized maximum pixel value
```

contiued

```
Procedure
    sum ← 0
    For all i,j do        {where i = 0..W-1,     j = 0..H-1}
        sum ← sum+Arr(i, j)
    End For
    mean ← sum / (W * H)
    sum ← 0
    For all i,j do        {where i = 0..W-1,     j = 0..H-1}
        sum ← sum+(Arr(i, j) - mean) ^2
    End For
    std ← Sqr(sum / (W * H) - 1)
    lmin ← mean - 1.5 * std
    lmax ← mean+1.5 * std
    stp ← (lmax - lmin) / Ng
    For all i,j do        {where i = 0..W-1,     j = 0..H-1}
        If Arr(i, j) < lmin Then
                QuanArr(i, j) ← 0
        ElseIf Arr(i, j) > lmax Then
                QuanArr(i, j) ← Ng - 1
        Else
                QuanArr(i, j) ← (Arr(i, j) - lmin) / (lmax - lmin)) * (Ng-1)
        End If
    End For
End
```

2. Co-occurrence Matrix Computation

The proposed method for the determination of image co-occurrence matrix is accomplished in four steps as follows:

***Step 1*:** Determine the normalized co-occurrence matrices of each image. The number of gray levels of the tested image is reduced to less levels (say Ng). For all 0^o, 45^o, 90^o and 135^o directions, and one of the distances 1, 2, 3, and 4, calculate the corresponding co-occurrence matrix. This produces four matrices of (Ng×Ng) integer elements per matrix.

***Step 2*:** Determine the values of the adopted descriptors. For each co-occurrence matrix, the value of each descriptor is calculated. For each image, the resulting descriptors values are stored in a features vector (single dimensional array) where each element is a real valued number.

***Step 3*:** Generate image signatures. The image signatures are calculated from the descriptors (features vector matrix) by normalizing the features vector values that correspond to different directions for the chosen inter-distance for each image.

***Step 4*:** Compare the images through their signatures. The extracted signatures of the images are compared by using certain similarity measure. Euclidean distance function was adopted for this purpose.

Step 1 is illustrated in Pseudo code list (4.4), while step 2 is shown in Pseudo code list (4.5). Steps 3 and 4 are explained in Pseudo code lists (4.13) and (4.17) respectively.

Pseudo code list (4.4) Co-occurrence matrix computation routine

```
Input
    A: Quantized two dimensional array
    d: selected distance.
Output
    ar0, ar45, ar90, ar135: normalized two dimensional arrays      {four GLCM }
Variables
    i, j: Image array indices
    W, H: image array Width and Height
    Nth: Normalizing factor for each GLCM
    Ng: number of quantization levels
Procedure
    For each i, j do
        increment  ar0(A(i, j), A(i, j+d)) by one          { ar0 takes   i= 0..W-1, and
        increment  ar0(A(i, j+d), A(i, j)) by one                j=0..H-1-d.  }

        increment  ar45(A(i, j), A(i+d, j+d)) by one {ar45 takes i= 0..W-1- d,
        increment  ar45(A(i+d, j+d), A(i, j)) by one            and j=0..H-1-d.}

        increment  ar90(A(i, j), A(i+d, j)) by one   {ar90 takes i= 0..W-1-d, and
        increment  ar90(A(i+d, j), A(i, j)) by one       j=0..H-1.}

        increment  ar135(A(i, j), A(i+d, j-d)) by one  {ar135 takes i= 0..W-1-d,
        increment  ar135(A(i+d, j-d), A(i, j)) by one            and  j= d..H-1.}
    End For
```

```
                                                  contiued
    Nth ← 2*W*(H - d)
    NormalizeGLCM(ar0, Nth, Ng) { send GLCM to GLCM Normalization routine to
                                be normalized according to its normalizing factor Nth and
                                number of quantization levels Ng}
    Nth ← 2*(W - d)*(H - d)
    NormalizeGLCM(ar45, Nth, Ng)
    Nth ← 2*(W - d)*H
    NormalizeGLCM(ar90, Nth, Ng)
    Nth ← 2*(W - d)*(H - d)
    NormalizeGLCM(ar135, Nth, Ng)
End
```

Each of the computed GLCM (i.e., ar0, ar45, ar90, and ar135) is normalized by division by the factor Nth which is a total number of occurrences over each matrix. Normalization insures that GLCM is invariant against image size changes.

3. Co-occurrence Features Extraction

The previous subsection shows the involved generation steps of the four normalized GLCM; *ar0, ar45, ar90, and ar135*. However, these matrices are still containing much data. Suppose that (Ng=18), then each matrix will consist of (18×18=324) elements, and they need to be further reduced. What is usually done is analyzing these matrices and computing a few simple numerical values that encapsulate the information held in matrices. Some statistical parameters that are computed from GLCM can be used instead of the whole matrix. The textural features (or descriptors) that are extracted from the co-occurrence matrix are presented in subsection (2.7.5), and are determined by implementing the steps listed in Pseudo code list (4.5).

Usually a subset of these descriptors is adequate for efficient retrieval. The detailed implementation of each descriptor function is straightforward and could be directly concluded from the equations listed in subsection (2.7.5), and they weren't mentioned here.

Pseudo code list (4.5) Co-occurrence features extraction routine

```
Input
    ar0, ar45, ar90, ar135:  GLCM  { the four co-occurrence Matrices }
Output
    FeatuVec(0..31): GLCM Features Vector array
Variables
    Ng: number of quantization levels
Procedure
  FeatuVec(0)  ← ASMfeature(Ng, ar0)          { Angular second moment feature}
  FeatuVec(1)  ← ASMfeature(Ng, ar45)
  FeatuVec(2)  ← ASMfeature(Ng, ar90)
  FeatuVec(3)  ← ASMfeature(Ng, ar135)
  FeatuVec(4)  ← IDMfeature(Ng, ar0)          { Inverse Difference Moment }
  FeatuVec(5)  ← IDMfeature(Ng, ar45)
  FeatuVec(6)  ← IDMfeature(Ng, ar90)
  FeatuVec(7)  ← IDMfeature(Ng, ar135)
  FeatuVec(8)  ← CORHarFeature(Ng, ar0)       { Haralick Correlation }
  FeatuVec(9)  ← CORHarFeature(Ng, ar45)
  FeatuVec(10) ← CORHarFeature(Ng, ar90)
  FeatuVec(11) ← CORHarFeature(Ng, ar135)
  FeatuVec(12) ← SHDfeature(Ng, ar0)          {Cluster Shade Feature }
  FeatuVec(13) ← SHDfeature(Ng, ar45)
  FeatuVec(14) ← SHDfeature(Ng, ar90)
  FeatuVec(15) ← SHDfeature(Ng, ar135)
  FeatuVec(16) ← PRMfeature(Ng, ar0)          {Cluster Prominence Feature }
  FeatuVec(17) ← PRMfeature(Ng, ar45)
  FeatuVec(18) ← PRMfeature(Ng, ar90)
  FeatuVec(19) ← PRMfeature(Ng, ar135)
  FeatuVec(20) ← ENTfeature(Ng, ar0)          { Entropy }
  FeatuVec(21) ← ENTfeature(Ng, ar45)
  FeatuVec(22) ← ENTfeature(Ng, ar90)
  FeatuVec(23) ← ENTfeature(Ng, ar135)
  FeatuVec(24) ← HOMfeature(Ng, ar0)          {Homogenity }
  FeatuVec(25) ← HOMfeature(Ng, ar45)
  FeatuVec(26) ← HOMfeature(Ng, ar90)
  FeatuVec(27) ← HOMfeature(Ng, ar135)
  FeatuVec(28) ← CONfeature(Ng, ar0)          {Contrast }
  FeatuVec(29) ← CONfeature(Ng, ar45)
  FeatuVec(30) ← CONfeature(Ng, ar90)
  FeatuVec(31) ← CONfeature(Ng, ar135)
End
```

4.4.2 Correlation histogram (Correlogram)

Correlogram is the extended color version of the co-occurrence matrix. Correlogram determination process is presented in Pseudo code list (4.6). A color correlogram is a table indexed by color pairs, where each entry (i, j) of the table specifies the probability of finding a pixel of color j at d distance from a pixel of color i in the image. Correlogram computation begins by separating the three components of the color array. Each color component is quantized in order to minimize the size of the correlogram array.

Pseudo code list (4.6) Correlogram "matrices and features" routine

```
Input
    RGBarr: Image array
Output
    FeatuVec: CLGM features vector array
Variables
    HueArr, LumArr, SatArr : Two dimensional arrays representing the three color
                             components
    HueQuanArr, LumQuanArr, SatQuanArr: Two dimensional quantized arrays
    Ng: Integer no. representing selected quantized levels
    NHue, NLum, NSat: quantization levels for the three color components
    SingleQuanArr: Two dimensional quantized array
Procedure
    {Convert RGB model to the selected color model }
    {Branch color model into its three primary components}
    LumArr ← Luminance color channel
    SatArr ← Saturation color channel
    HueArr ← Hue color channel
    QuantizeHue (HueArr, NHue, HueQuanArr)
    OneChannelQuantize(LumArr, NLum, LumQuanArr )
    OneChannelQuantize(SatArr , NSat, SatQuanArr )
    SingleQuanMatrix (SatQuanArr, LumQuanArr, HueQuanArr, NSat, NLum,
                      NHue, SingleQuanArr)
    Co-occurrenceMatrix(SingleQuanArr, Ng, ar0, ar45, ar90, ar135 ) {send
        quantized array to co-occurrence matrices computation routine}
    Co-occurrenceFeaturesExtraction(ar0, ar45, ar90, ar135, FeatuVec) {send the
    four GLCM to the features extraction routine }
End
```

The quantization of Hue component is done in special way; it is explained in Pseudo code list (4.7). The other two components; Saturation and Luminance have flat (or linear) nature, so the one channel quantization process shown in Pseudo code list (4.3) suffices in handling both components. Single quantized matrix process which maps the three color components into single component array is detailed in Pseudo code list (4.8).

Co-occurrence matrix determination and co-occurrence features extraction process follow the same implementation illustrated in Pseudo code lists (4.4) and (4.5) respectively.

1. Hue Quantization

Hue component is quantized in a specific way because of its circular nature. The Hue parameter is measured as the angle around the axes (Luminance), and has the range [0°, 360°) distributed over a color wheel; see Figures (2.2), and (2.4) through (2.9) for Hue representations in different colors models.

Pseudo code list (4.7) Quantization of Hue component

```
Input
    HueArr: Hue component Array
    NHue: Hue quantization levels
Output
    HueQuanArr: Hue Quantized Array
Variables
    i,j: array indices
    Intensity: three levels intensity values to be substituted for the undefined Hue
               values
Procedure
      Intensity ← 2          {maximum quantized intensity value }
      For all i,j do               {where i = 0..W-1,      j = 0..H-1}
          HueQuanArr(i, j) ← HueArr(i, j) * (NHue / 359) {normal Hue value}
          If HueQuanArr(i, j) = NHue Then  { emulate circular Hue nature }
              HueQuanArr(i, j) ← 0
          ElseIf HueQuanArr(i, j) > NHue Then      { Hue undefined }
              HueQuanArr(i, j) ← NHue+(HueArr(i, j)-500) * (Intensity / 255)
          End If
      End For
End
```

In the Hue circle, the primary colors (red, green, and blue) are separated by 120°. The secondary colors (yellow, magenta, and cyan) are also separated by 120° and are 60° away from the two nearest primary colors. For example, when Hue is divided in 18 bins, each primary (or secondary) color is represented by three subdivisions. Hence, the quantization process should preserve this Hue circular order. In addition, Hue is undefined when Saturation is zero. To cope with this situation, the quantized intensity value is used instead. A value of 500 is added beforehand to the intensity pixel value (for which Hue is undefined) in order to differentiate it from other normal Hue values. This added value is removed before quantization. Intensity component is quantized into 0, 1, or 2 and added to the upper quantized Hue value to give final quantized value for the undefined Hue value.

2. Single Quantized Matrix Computation

In order to improve the performance of correlogram computation process, a proper color quantization scheme should be applied to reduce the color resolution. The process of color quantization requires that each axis is divided into a number of parts. Then, the three quantized color components are sent to the single quantized matrix determination stage, whose implementation steps are listed in Pseudo code list (4.8), in order to be mapped into a single component. When each axis is divided into a number of quantization levels, the total number of quantized levels *Ng* used to represent an image would be:

$$Ng = N_1\ N_2\ N_3\ ,\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (4.1)$$

where N_1, N_2, N_3 are the numbers of quantization levels for the three color components. For the undefined Hue value the quantized intensity value stored in Hue array is added to the total Ng. The actual three color components values for each pixel are mapped into a single color value according to the following equation:

$$\text{Color} = \text{Saturation} + N_1\ \text{Luminance} + N_1\ N_2\ \text{Hue}\ ,\ \ldots\ldots \qquad (4.2)$$

where N_1 and N_2 are the numbers of quantization levels for Saturation and Luminance respectively. N_3 is not included in the equation.

Pseudo code list (4.8) Single quantized matrix computation routine

```
Input
    HueQuanArr, LumQuanArr, SatQuanArr: Two dimensional Quantized Arrays
                            representing the three color components
    NHue, NLum, NSat: quantization levels for the three color components
Output
    SingleQuanArr: Two dimensional Quantized Array
Variables
    i,j: arrays indices
Procedure
  For all i, j do   {where i= 0..W-1,  j=0..H-1}
      If HueQuanArr(i, j) >= NHue Then    { Hue is undefined }
          SingleQuanArr(i, j) ← NSat * NLum * NHue + HueQuanArr(i, j) -
                                   (NHue-1 )
      Else
          SingleQuanArr(i, j) ←SatQuanArr(i, j)+NSat * LumQuanArr(i,j) +
                                   (NSat * NLum) * HueQuanArr(i, j)
      End If
  End for
End
```

4.4.3 Color Histogram and Cumulative Histogram

As described in subsection (2.5.1), color histogram is easy to compute and effective in characterizing the distribution of colors in an image. The single quantized matrix resulted from the process described in Pseudo code list (4.8) is fed to the histogram computation and normalization routine explained in Pseudo code list (4.9), (i.e., the three color components are represented by a condensed single component histogram).

Color histogram is firstly computed and normalized by using this single quantized matrix. To achieve high storage and retrieval efficiency, the number of histogram bins used is normally much smaller than the total number of colors used to represent images. Therefore, a number of adjacent colors have to be grouped into one bin.

Pseudo code list (4.9) Histogram computation and normalization

```
Input
    SingleQuanArr: Quantized Array
    Ng: number of histogram bins
Output
    HistArr: normalized histogram Array
Variables
    i,j: array indices
Procedure
    For all i,j do                  {where i = 0..W-1,      j = 0..H-1}
          increment HistArr(SingleQuanArr(i, j)) by one
    End For
    For all i do                    {where i = 0..Ng - 1}
          HistArr(i) ← HistArr(i) / (W * H)
    End For
End
```

Color histogram is totally sparse. Hence, any change in lighting conditions may cause a shift in the color histogram. This produces misjudged similarity which is caused by color shift; as illustrated in Figure (2.12). In contrary, cumulative histogram vector is always completely dense and hence less sensitive to noise; see subsection (2.5.2). Pseudo code list (4.10) shows the cumulative histogram determination steps, where the input is a normalized histogram array resulted from the histogram normalization process.

Pseudo code list (4.10) Cumulative histogram routine

```
Input
    HistArr: normalized Histogram Array
    Ng: number of histogram bins
Output
    CumulHistArr: Cumulative Histogram Array
Variables
    i: histogram array index
Procedure
    CumulHistArr(0) ← HistArr(0)
    For all i do                    {where i= 1..Ng-1}
        CumulHistArr(i) ← CumulHistArr(i-1)+HistArr(i)
    End For
End
```

4.4.4 Moments

Only 9 moments (three moments for each of the three color components) have been used to represent the color content of each image. Color moments are very compact representation compared to other color features. Due to this compactness, the discrimination power may be degraded. Usually, color moments can be used as the first pass to narrow down the search space before other sophisticated color features are used for retrieval. The established moments routine is given in Pseudo code list (4.11).

As discussed in subsection (2.5.3), most of the information is concentrated on the low-order moments. Therefore, only the first order (mean), the second (variance) and the third order (skewness) color moments were utilized. They proved to be efficient and effective in representing color distributions of images [3, 7].

Pseudo code list (4.11) Moments determination routine

```
Input
    RGBarr: RGB Image array
Output
    MomArr: Moments features Array
Variables
    HueArr, LumArr, SatArr : Arrays representing the three color components
Procedure
        {Convert RGB model to the selected color model }
        {Branch color model into its three primary components }
            LumArr ← Luminance color channel
            SatArr ← Saturation color channel
            HueArr ← Hue color channel
        Moments( HueArr, 0, MomArr)       {0..3 MomArr values are concluded from
                                          HueArr}
        Moments (LumArr, 4, MomArr)       {4..7 MomArr values are concluded from
                                          LumArr}
        Moments (SatArr , 8, MomArr )     {8..11 MomArr values are concluded from
                                          SatArr}
End
```

The essence of this routine is the moments computation routine presented in Pseudo code list (4.12).

Pseudo code list (4.12) Moments computation routine

```
Input
    ChannelArr: One channel Array
    k: Moment array current index
Output
    MomArr: Moments features Array
Variables
    sum: summation
    i,j: array indices
    Mom4: fourth moment
    kur: kurtosis
Procedure
    sum ← 0
    For all i,j do      {where i = 0..W-1,      j = 0..H-1}
        sum ← sum+ChannelArr(i, j)
    End For
    MomArr(k) ← sum / (W * H)
    sum ← 0
    For all i,j do {where i = 0..W-1,      j = 0..H-1}
        sum ← sum+(ChannelArr(i, j) - MomArr(k))^2
    End For
    MomArr(k+1) ← Sqr(sum / (W * H))
    sum ← 0
    For all i,j do {where i = 0..W-1,      j = 0..H-1}
        sum ← sum+(ChannelArr(i, j) - MomArr(k))^3
    End For
    MomArr(k+2) ← (sum / (W * H))^(1/3)
    sum ← 0
    For all i,j do {where i = 0..W-1,      j = 0..H-1}
        sum ← sum+(ChannelArr(i, j) - MomArr(k))^4
    End For
    Mom4 ← (sum / (W * H))^(1/4)
    kur ← Mom4 / (MomArr(k+1)^4)
    MomArr(k+3) ← MomArr(k+1) / (kur^(1/4))  { Contrast Feature }
End
```

In addition to the first three moments, the contrast feature (which is determined from moments features) is added to the moments features vector resulting in 12 features covering the three color channels. Contrast computation is formulated in equation (2.35).

Each call to the moments routine results in computing four features (the first three moments orders and the contrast) for each color component array. So this routine is invoked three times for every color model (one time for each color component) to set up the moments features vector which consists of 12 features elements.

4.5 Features Vector Normalization

After the extraction of all features vectors that were mentioned in the previous sections, features vectors need to be normalized to specific scale. Normalization prepares features vectors for comparison by using a selected measure. Normalization process is made by finding the minimum and maximum values for each feature. Minimum feature value will be mapped to Zero, while the maximum one will be mapped to 100. Other feature values are scaled between these two extremes. Knowing '*min*' and '*max*' feature values, the *'Normalized'* feature value will be computed as:

$$Normalized = \frac{100}{max - min}(ActualValue - min), \ldots\ldots\ldots\ldots\ldots \qquad (4.3)$$

The implemented steps of the normalization process are shown in Pseudo code list (4.13).

For the same purpose and within the same stage, the features variance routine is called, where the mean and variance of each feature is computed and stored in '*MeanArr' and 'VrnceArr'* respectively. Feature variance is used when computing distance between any corresponding features vectors, such that the individual feature distance is weighted according to the variance of that feature. By this improvement Euclidean distance would be computed as follows:

$$D(P,Q) = \left(\sum_{i=0}^{n-1} \frac{(P_i - Q_i)^2}{VrnceArr(i)} \right)^{1/2}, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad (4.4)$$

P and *Q* are two features vectors (representing images), each is of *n* features. *'VrnceArr(i)'* is the variance of the i^{th} feature.

Pseudo code list (4.13) Features vector normalization routine

```
Input
        CluArr: all clusters features Array        {unnormalized }
Output
      MinFeatuArr, MaxFeatuArr: Arrays of the minimum and maximum features
                                    value
      CluArr: all clusters features Array      { normalized }
Variables
      min, max: minimum and maximum feature value
      FeatuNo:  Total Features number
      NoClu: Total Clusters number
Procedure
    For all k do                          { where k = 0 .. FeatuNo - 1}
        min ← CluArr(0).FeatuVec(k)
        max ← CluArr(0).FeatuVec(k)
        For all i do                      {where i = 1 .. NoClu - 1 }
            If min > CluArr(i).FeatuVec(k) Then
                  min ← CluArr(i).FeatuVec(k)
            If max < CluArr(i).FeatuVec(k) Then
                  max ← CluArr(i).FeatuVec(k)
        End For
        For all i do                      {where i = 0 .. NoClu - 1 }
                CluArr(i).FeatuVec(k)←(100/(max-min))*
                                        (CluArr(i).FeatuVec(k)- min)
        End For
        MinFeatuArr(k) ← min
        MaxFeatuArr(k) ← max
    End For
End
```

4.6 Hierarchical Indexing

Image retrieval systems that compare the query image exhaustively with each individual image in the database are not scalable to large databases. A scalable system should ensure that the search time does not increase linearly with the number of images in the database. In this research project a clustering based indexing technique is introduced, where the images in the database are grouped into clusters of images with similar color/texture content using a hierarchical clustering algorithm.

At search time the query image is not compared with all the images in the database, but only with a small subset. The conducted experiments show that this clustering based approach offers a superior response time with high retrieval accuracy. The clustering must be performed in such a way that the retrieval accuracy is not sacrificed in this process. The clustering technique does not directly depend on the nature of the applied similarity measure used to compare the images, so this technique can be used with most general similarity measures.

4.6.1 Hierarchical Agglomerative Clustering

The Hierarchical Agglomerative Clustering (HAC) algorithm is a commonly employed classical hierarchal clustering algorithm. The result of HAC is a dendrogram representing the nested groups of images. In this work, an improved version of HAC algorithm is developed.

Let n be the number of images in the database, the similarity between all pairs of images is precomputed according to selected features. The hierarchical clustering is performed as follows:

1. The n images in the database are placed within n distinct clusters. These clusters are indexed by $\{C_0, C_1, C_2, ..., C_{NoClu-1}\}$ where $NoClu=n$. They are stored in *CluArr* which is an array of records. Each record represents a cluster with the following fields:

   id: cluster number.

   NoOfImg: number of images in a cluster.

   ImgArr: one dimensional array holding images pathes.

   FeatuVec: one dimensional array holding image features vector.

   Right: holds the right hand cluster number.

   Left: holds the left hand cluster number.

After this step each image is assigned to single cluster. Each cluster has the following information: *NoOfImg*=1, *ImgArr*=images identifiers, *FeatuVec*=extracted features, *Left*= -1, *Right*= -1.

2. Any two clusters (*k* and *l*) whose distance measure value $D_{k,l}$ is the minimum are picked and lumped (merged) into a new cluster $C_{NoClu+1}$. This reduces the total number of unmerged clusters by one. The new cluster will contain all the images that belong to k^{th} and l^{th} clusters. One of the two childs of $C_{NoClu+1}$ is referred to as the right child; *Right=k,* and the other as the left child; *Left=l.* The similarity measures between the new cluster $C_{NoClu+1}$ and all the remaining clusters are computed.
3. *Step 2* is repeated until the number of clusters is reduced to a certain predefined number, or when the largest similarity measure between the tested clusters is dropped to some lower threshold.

General HAC algorithms produce a nested sequence of clusters, with a single all-inclusive cluster at the top and single point clusters at the bottom. The algorithm is modified to stop when the desired number of clusters is obtained or the distance between two closest clusters is above a certain threshold distance. After each merge step, the total number of clusters decreases by one. The above mentioned step (1) is the basis of the whole clustering process. It is illustrated in the Pseudo code list (4.14).

Figure (4.2) shows a sample of hierarchical clustering with 8 images. Step 1 produces eight clusters; each cluster has a single image with its extracted features. Left and right childs indices are assigned the value (-1). This assignment (i.e., -1) will guide the search engine to stop searching down this node when a query image is compared along this indexed structure.

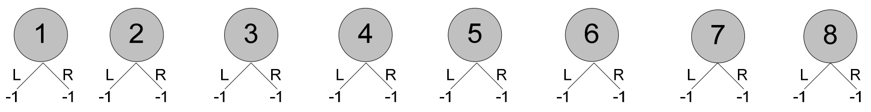


Figure (4.2) Initial clustering

Pseudo code list (4.14) Initial clustering routine {without merging}

```
Input
        n image files.
Output
        n clusters ; each of exactly one element (image).
Variables
        NoClu: no of clusters.
        RGBarr: two dimensional array holding the RGB components of each image
                file.
        CluArr: Clusters Array; one dimensional array of records. Each record
            represents a cluster. Each cluster record has the following fields:
            id: cluster number.
            NoOfImg: number of elements in a cluster.
            ImgArr: one dimensional array which holds the file names of the images
                    belong to that cluster
            FeatuVec: one dimensional array holding the image features vector.
            Right: holds the right hand cluster number.
            Left:  holds the left hand cluster number.
Procedure
        {Convert images folder to File System Object}
        Set FSO ← File System Object
        Set Count ←  number of image files in FSO
        NoClu ← 0                        {image counter }
        For Each ImageFile in FSO do
              NoClu ←  NoClu+1
              read ImageFile data into RGBarr {read image data into an array}
              CluArr(NoClu ).id ←  NoClu
              CluArr(NoClu ).NoOfImg ←  1
              CluArr(NoClu ).ImgArr(0) ←  ImageFile name
              Call features extraction routine  { according to selected color model and
                                                  features type }
              CluArr(NoClu ).FeatuVec ← Extracted features
              CluArr(NoClu ).Right ← -1
              CluArr(NoClu ).Left ←  -1
          End If
        End For
 End
```

1. Index Construction

Given the huge number of the registered images in the database and the high dimensionality of classical features spaces, it is important to avoid exhaustive, linear-time, and direct comparisons. In other words, it is necessary to automatically map features vectors into indices which can provide more rapid access to the relevant images in the database. In this research work, an approach for computing multidimensional indices for an image database was investigated. In addition, the process of organizing such indices in a hierarchical tree structure to allow logarithmic-time binary search was analyzed.

In order to perform efficient content-based retrieval from the image database, it is necessary to construct a hierarchical index representation. Given a query image described by a vector, its index structure should directly point to the classes of images similar to this query image.

2. Hierarchical Index Representation

Clustering algorithm may represent, simply, a convenient method for organizing a large set of data so that the retrieval of information may be made more efficiently. Cluster representatives should provide a very convenient summary of the database. In another say, it forms a narrowing down phase of the whole search space. In this work, cluster representative is its centroid. Index structure is an improved indexing scheme where the calculated centroid of each cluster is the index.

Under the root level there is an index array. Each entry of the index array holds the unique identification number of each cluster followed by the centroid. Centroid may consist of one or more features vectors according to the number of used features classes. In the same level, images count and identifiers of all images contained in that cluster are defined. Left and right child pointers are also labeled according to merging process. Figure (4.3) illustrates the main index structure. Index structure contains mother clusters only. Clusters array *'CluArr'* which was previously mentioned has the same structure, but it contains all clusters introduced along the clustering process including single image clusters. *'CluArr'* is the reference for the index array.

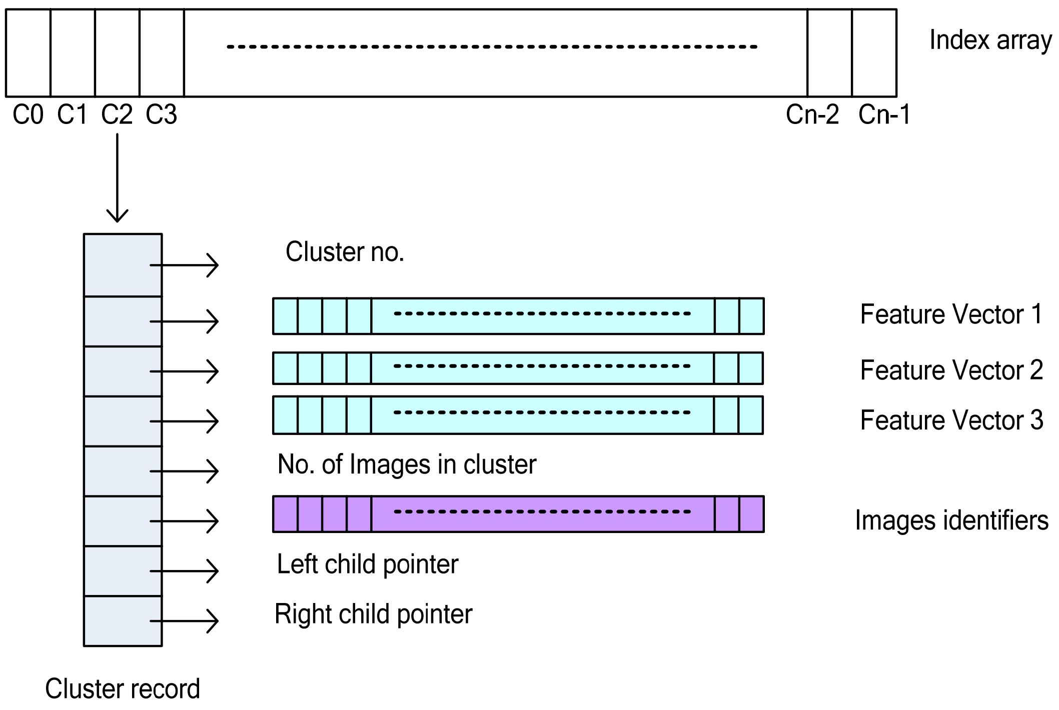


Figure (4.3) Main index structure

Underneath any cluster entry in the main index array, there is an associated binary tree along which an indexed search may be driven; as depicted in Figure (4.4). The same structure holds for all parts of the clusters array *'CluArr'*.

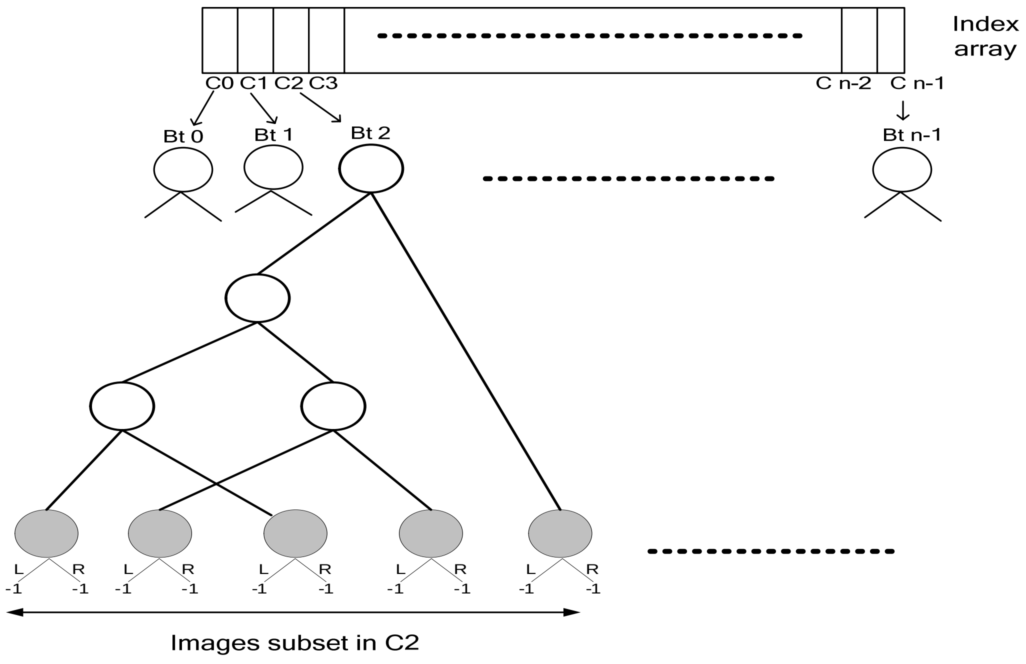


Figure (4.4) Binary tree structure associated with each mother cluster

4.6.2 Distance Matrix

Hierarchical agglomerative algorithm starts with all the data points as separate clusters. A matrix of inter-cluster distance for all singleton clusters is computed. The obtained similarity value between cluster *i* and cluster *j* is denoted by δ_{ij}. By using certain similarity measure the image similarity between each pair of images could be calculated. Then, the Hierarchical Agglomerative Clustering algorithm (HAC) is applied to construct a hierarchy.

Similarity values are normalized usually by mapping and then taking the average of the values of all δ_{ij}, that are obtained from all measurements. Normalization is done in a similar way to that applied on the features, which is shown in Pseudo code list (4.13). Distance matrix is symmetric; therefore, only the upper diagonal part is used to store inter-cluster distances. A clusters pointers array is associated with the distance matrix. Each entry in the clusters pointers is pointed to the cluster number represented by the corresponding row in the distance matrix. Usually the two matrices are updated after each merge process; see Figure (4.6).

The steps of determination of the clusters distance matrix are listed in Pseudo code list (4.15), in which Euclidean distance is used as a distance measure between clusters. Other measures may be used instead, as explored in the following articles, where the Euclidean distance between different features entities is given.

Pseudo code list (4.15) Clusters distance matrix computation

```
Input
    CluArr: Clusters Array; one dimensional array of records. Each record
            represents a cluster
    NoClu: number of clusters
Output
    DistArr: the resulting Distance matrix
Variables
    i,j: array indices
Procedure
    For each i,j do          {where i = 0.. NoClu - 2 ,  j = 0.. NoClu - 1}
        DistArr(i, j) = EuclideanDist(i, j)
    End For
End
```

1. Euclidean Clusters Distance

The implementation of the modified Euclidean distance formula (see equation (4.4)) is illustrated in Pseudo code list (4.16). Cluster features are always read from the clusters array *'CluArr'*, and indexed by the pointers list 'CluPntr'.

Pseudo code list (4.16) Euclidean clusters distance function

```
Input
    id1, id2: clusters identifier
Output
    EuclideanDist: Euclidean Distance between clusters id1 and id2
Variables
    CluArr: Clusters Array; one dimensional array of records (clusters)
    ClurPntr: Cluster Pointer array
    VrnceArr: Features Variance Array
    k: Feature array index
    Sum: squared features difference
Function
   Sum ← 0
   For all k features do        { where k= 0 .. FeatuNo - 1 }
        Sum ← Sum+((CluArr(ClurPntr (id1)).FeaturesVec(k) -
                CluArr(ClurPntr (id2)). FeaturesVec(k)) $^2$)/VrnceArr(k))
   End For
   EuclideanDist ← Sqrt(sum)
   Return EuclideanDist
End
```

2. Euclidean Co-occurrence Clusters Distance

Euclidean distance between the co-occurrence features is computed by using a modified way. In order to minimize the variation that may occur in the computed distance between the features vectors (due to different image variations), an improved method based on choosing a minimum feature group distance is introduced. Given four GLCM; *ar0, ar45, ar90,* and *ar135*, each co-occurrence matrix feature has four feature descriptors participated in the computed features vector. Each four elements are representing a single feature, and they are called a feature group; therefore, the total number of elements contained in a features vector is a multiple of four.

The distance between any two feature groups is measured separately four times with different shifts *(g)* as indicated in Figure (4.5). In each step, the feature corresponding index is given by *'modulo'* variable, while the group indicator is given by *'IntegerPart'* variable; see Pseudo code list (4.17). Out of these four determined distances the minimum one is chosen as the actual distance between the two tested features groups. The feature corresponding shift pointer is assigned to *FeturPntr(k),* where *k* is the feature group starting index.

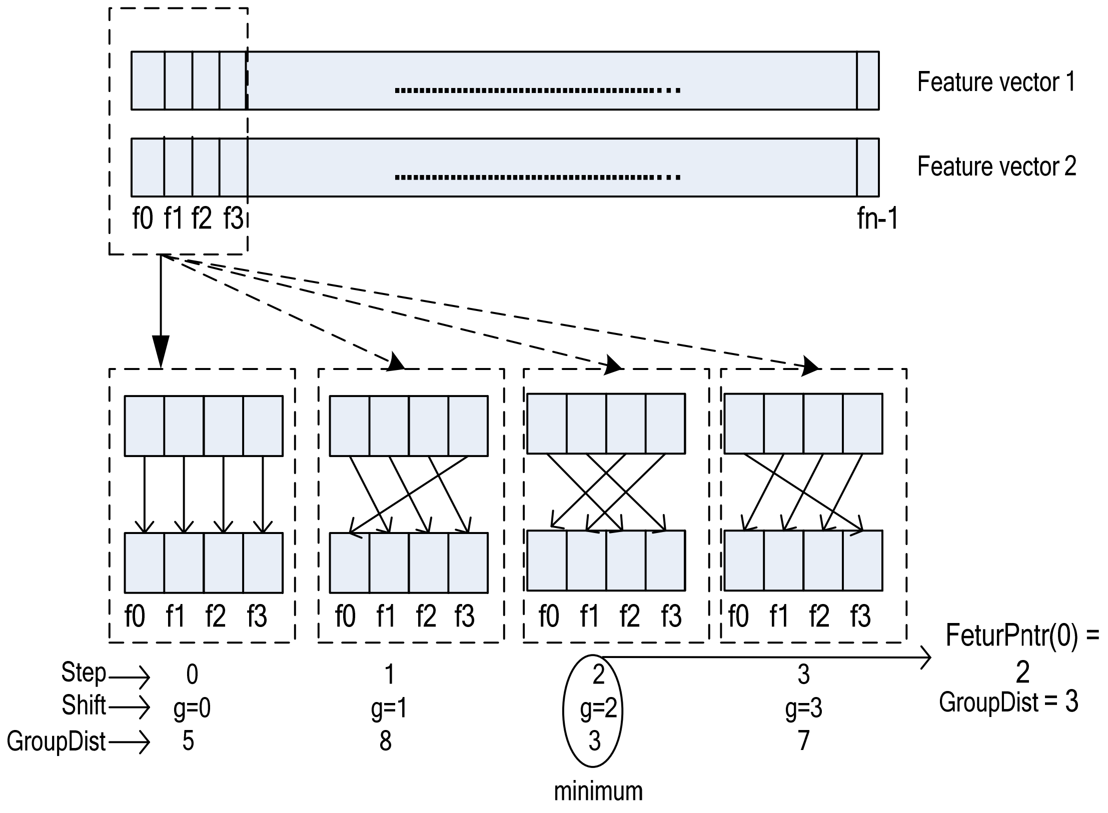


Figure (4.5) Feature group distance computation

This process is repeated for all feature groups. The determined distances *'GroupDist'* for all groups are summed to give the final features vectors distance.

Pseudo code list (4.17) Euclidean co-occurrence clusters distance function

```
Input
        id1, id2: clusters identifier
Output
        OccDist: all features distance
Variables
        i, k: features array index
        g: features group circular index
        SinDist: single feature distance
        Dist, GroupDist: feature group distance
        Modulo: feature corresponded shift
        IntegerPart: feature group indicator
        FeturPntr: feature corresponding shift pointer array
        CluArr: Clusters Array          {one dimensional array of records (clusters)}
        VrnceArr: Features Variance Array
Function
        OccDist ← 0                            { total distance }
        For k = 0 .. FeatuNo - 4  step 4 do    {each feature group has 4 features }
                GroupDist ← 100                { initial large value }
                For g = 0 To 3  do             { feature shift }
                    Dist ← 0
                    For i = k To k+3 do        { group feature index }
                        Modulo ← (i+g) mod (k+4)
                        IntegerPart ← (i+g) div (k+4)            { integer division }
                        SinDist ← CluArr(ClurPntr(id1)).FeatuVec(i) -
                          CluArr(ClurPntr(id2)).FeatuVec(Modulo+IntegerPart*k)
                        Dist ← Dist+($SinDist^2$ / VrnceArr(i))
                    End For
                    If Dist < GroupDist Then
                        GroupDist ← Dist
                        FeturPntr(k) ← g
                    End If
                End For
                OccDist ← OccDist+GroupDist
        End For
        OccDist← $OccDist^{1/2}$
        Return OccDist      { function output }
End
```

3. Histogram Intersection Distance

This distance measure indicates the intersection magnitude between any two histograms. The equivalence is designated with similarity value 1, and the similarity between the two histograms decreases when its value approaches 0. Both of the histograms must be of the same size to have a valid similarity value.

Let $H_1[1..n]$ and $H_2[1..n]$ denote two histograms of size n, and $\delta_{H1,H2}$ denote the similarity value between H_1 and H_2. Then, this similarity can be expressed by the following formula [30, 93]:

$$\delta_{H_1,H_2} = \frac{\sum_i^n \min\left(H_1[i], H_2[i]\right)}{\min\left(|H_1|,|H_2|\right)} \quad , \text{........................} \qquad (4.5)$$

Hence, the distance between histograms H_1 and H_2 could be expressed as:

$$D_{H_1,H_2} = 1 - \delta_{H_1,H_2} \quad , \quad \text{..........................} \qquad (4.6)$$

A special case of the histogram intersection method is the L_1 distance measure introduced in equation (2.23). Its performance is compared to the previous one in chapter five. The implementation of histogram intersection distance method is described in Pseudo code list (4.18).

Pseudo code list (4.18) Histograms intersection distance function

```
Input
    id1, id2: clusters identifier
    Ng: number of histogram bins
Output
    HistDist: Histograms Distance final output
Variables
    CluArr: Clusters Array; one dimensional Array of records (clusters)
    ClurPntr: Distance Matrix Cluster Pointer
    HistVec: Histogram Vector sub array
    i: Histogram bin index
    IntersecSum: Histograms Intersection
    sum1, sum2: Histogram summation
Procedure
    IntersecSum ← 0
```

continued

```
    For all i  'histogram bins' do              { where i = 0 .. Ng - 1 }
        IntersecSum ←IntersecSum+min(CluArr(ClurPntr(id1)).HistVec(i),
                                               CluArr(ClurPntr(id2)).HistVec(i))
    End For
    sum1 ←  0
    For all i  'histogram bins' do              { where i = 0 .. Ng - 1 }
          sum1 ←  sum1+CluArr(ClurPntr(id1)).HistVec(i)
    End For
    sum2 ←  0
    For all i  'histogram bins' do              { where i = 0 .. Ng - 1 }
          sum2 ←  sum2+CluArr(ClurPntr(id2)).HistVec(i)
    End For
    FinalIntersec ←  IntersecSum / min(sum1, sum2)
    HistDist ←  1 - FinalIntersec
    Return HistDist
End
```

4.6.3 Clusters Lumping

Clusters merge is the core step in the whole clustering process. It involves merging of the two clusters which are the most similar pair in the distance matrix. This process is visualized in Figures (4.6) and (4.7). The algorithm starts with searching the proximity matrix to find the smallest element. Smallest element means the most similar images that can be lumped in one cluster. Once they are found, they will be merged and considered as a single object in the next step.

By searching *'DistArr'*, it is found, for example, that clusters *5* and *9* are the closest. Then, a merging process is performed by merging C_5 and C_9 into a new cluster C_{12}. Afterwards, both C_5 and C_9 are deleted from *'DistArr'* and *'ClurPntr'*, but they aren't removed from the clusters array *'CluArr'*. This is done by vertical and horizontal crawling processes, which simply involve moving of the next clusters in places of the deleted clusters in both *'ClurPntr'* and *'DistArr'*. C_{12} is pointed-to by *ClurPntr(10)*(i.e., it is now represented by row *10* in *'DistArr'*).

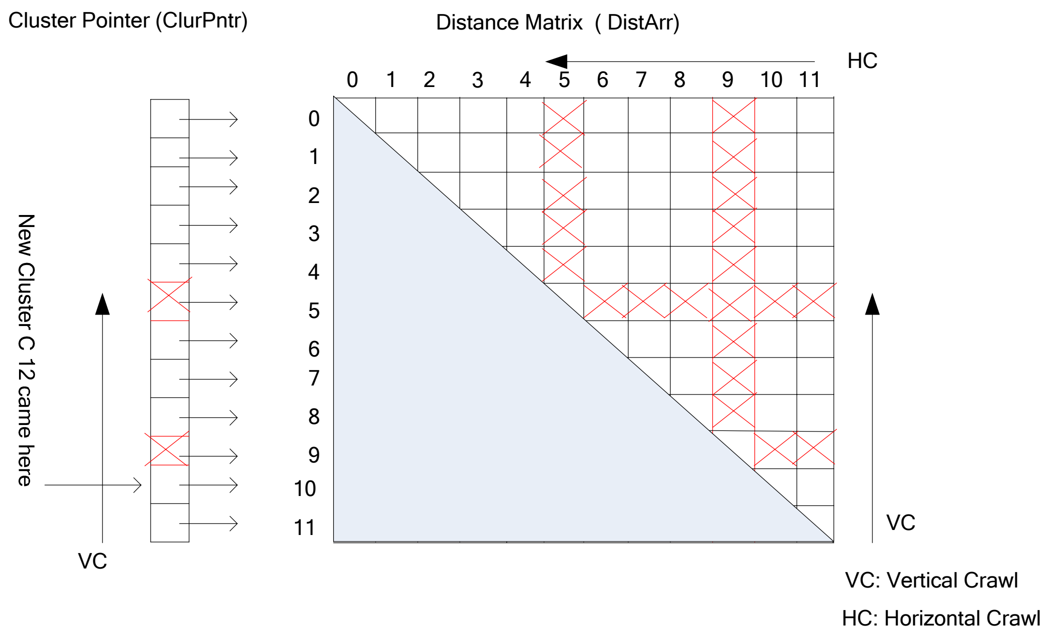
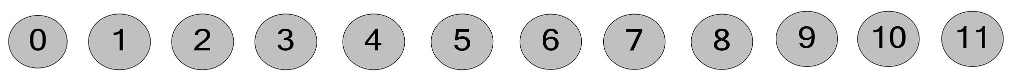
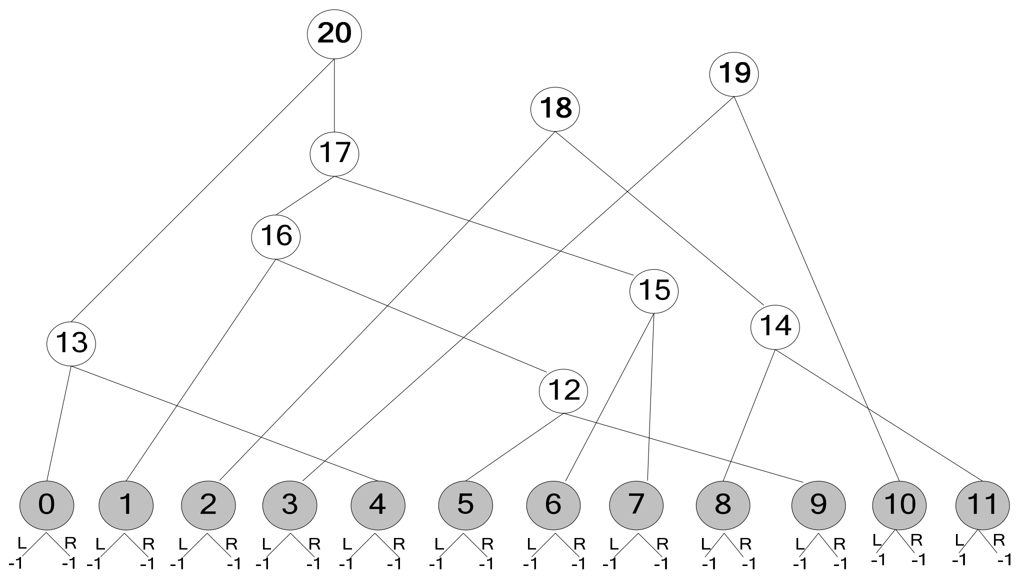


Figure (4.6) Distance matrix update



(a)



(b)

Figure (4.7) Clusters merging process with HAC;
(a) Initial images (clusters), (b) Final clusters.

Clustering process is continued in this way, as depicted in Figure (4.7), merging the next nearest clusters according to their inter-clusters distance in the 'DistArr'. Merging stage is stopped when reaching one of those stopping conditions explained earlier. For this example, each merged cluster has an identification number, and contains a set of images as summarized in Table (4.2). Each mother cluster has an associated binary tree in which left and right childs refer to other trees in the hierarchy. The clustering process results in the mother clusters: C_{18},C_{19},C_{20}. These three clusters are listed in the index array, so that query search will be driven by their associated binary trees.

Table (4.2) A sample of clusters data

Merged cluster no.	**No. of images in cluster**	**Images in cluster**	**Left child**	**Right child**
12	2	{5,9}	5	9
13	2	{0,4}	0	4
14	2	{8,11}	8	11
15	2	{6,7}	6	7
16	3	{1,5,9}	1	12
17	5	{1,5,6,7,9}	16	15
18	3	{2,8,11}	2	14
19	2	{3,10}	3	10
20	7	{0,1,4,5,6,7,9}	13	17

The implementation steps of the clusters merging process are demonstrated in Pseudo code list (4.19). It comprises a number of subroutine calls.

Pseudo code list (4.19) Clusters merge routine

```
Input
    CluArr: Clusters Array; one dimensional array of records (clusters)
Output
    DistArr: Distance Matrix
    CluArr: Clusters Array                    {dynamic array is expanded}
Variables
    id1, id2: the currently merged clusters identifiers
    ClurPntr: Distance Matrix Cluster Pointer
    MinDist: Minimum Distance
    NoClu: current number of clusters
    NxtClu: number of the next cluster to be merged
    NoMothClus: number of desired mother clusters
Procedure
    MinDist ← 0
    While (NoClu > NoMothClus) Or (MinDist < Threshold) do
        FindMinimumDist (DistArr, MinDist, id1, id2)
        Expand  CluArr(NxtClu+1)    {expand dynamic array by one}
        CluArr(NxtClu).NoOfImg ← CluArr(ClurPntr (id1)).NoOfImg+
                                    CluArr(ClurPntr (id2)).NoOfImg
        ComputeClusterCentroid (id1, id2)
        MergeClustersFiles (id1, CluArr(ClurPntr (id1)).NoOfImg, id2,
                             CluArr(ClurPntr (id2)).NoOfImg)
        CluArr(NxtClu).id ← NxtClu
        CluArr(NxtClu).Left ← ClurPntr (id1)
        CluArr(NxtClu).Right ← ClurPntr (id2)
        HorizontaCrawl (DistArr, id1)
        VerticalCrawl (DistArr, id1)
        HorizontaCrawl (DistArr, id2 - 1)
        VerticalCrawl (DistArr, id2 - 1)
        NoClu ← NoClu - 1                  { clusters number reduced }
        ClurPntr (NoClu - 1) ← NxtClu
        For each i  do                      { where i= 0 .. NoClu - 2 }
            DistArr(i, NoClu - 1) ← 100 / (MaxDistArr - MinDistArr) *
                              (EuclideanDist (i, NoClu - 1)-MinDistArr))
        End For
        NxtClu ← NxtClu+1                   {next new cluster that will result from
    End While                                the next merge iteration }
End
```

First step in merging process is finding out the minimum value in the '*DistArr'*. The involved steps of finding minimum distance routine are listed in Pseudo code list (4.20). The two clusters with the minimum distance are lumped in a new cluster each time; hence, the clusters dynamic array *'CluArr'*, is expanded in every iteration. The number of mages in new cluster is the total images in its left and right childs clusters.

Pseudo code list (4.20) Find minimum distance procedure

```
Input
        DistArr: Distance Matrix
Output
        id1, id2: identifiers of the two clusters with MinDist
        MinDist: Minimum Distance
Variables
        i,j: Distance Matrix indices
Procedure
        MinDist ← DistArr(0, 1)              { initial value }
        id1 ← 0
        id2 ← 1
        For each i, j do                {where i = 0.. NoClu-2 , j = 0.. NoClu-1}
                If DistArr(i, j) < MinDist Then
                        MinDist ← DistArr(i, j)
                        id1 ← i
                        id2 ← j
                End If
        End For
End
```

The subroutines *compute cluster centroid* and m*erge clusters files* are listed in the following article. The left and right childs of the new cluster point to the two clusters that produced it. Finally, the new cluster distances to all remaining clusters are computed and appended to the distance matrix.

1. Merged Cluster Data Computation

The centroid-based method calculates the distance by using only the centroid of a cluster (the mean of all the points in the cluster). The new cluster centroid is computed by averaging its two childs centroids. This what is exactly performed in Pseudo code list (4.21).

Pseudo code list (4.21) Compute cluster centroid routine

```
Input
    id1, id2: identifiers of the two clusters to be merged
    NxtClu: identifier of the new cluster
Output
    FeatuVec: one dimensional array holding centroid features vector
Variables
    k: features vector index
Procedure
    For all k do                        { where k= 0.. FeatuNo-1 }
      CluArr(NxtClu).FeatuVec(k) ← CluArr(ClurPntr(id1)).FeatuVec(k)+
                                      CluArr(ClurPntr(id2)).FeatuVec(k)) / 2
    End For
End
```

Images contained in the two childs clusters are collected into their parent new merged cluster as implemented in Pseudo code list (4.22).

Pseudo code list (4.22) Merge clusters files routine

```
Input
    id1, id2: the currently merged clusters identifiers
    n1, n2: nos. of image files to be merged in id1, id2 clusters respectively
    NxtClu: the new cluster number
Output
    ImgArr: image files Array of the new cluster
Variables
    i: image files counter
Procedure
      For all i do            {where i= 0..n1-1 }
            CluArr(NxtClu).ImgArr(i) ←  CluArr(ClurPntr(id1)).ImgArr(i)
      End For
      For all i do            {where i= n1..n1+n2-1 }
            CluArr(NxtClu).ImgArr(i) ←  CluArr(ClurPntr(id2)).ImgArr(i-n1)
      End For
End
```

2. Vertical and Horizontal Crawl Implementation

As depicted in Figure (4.6), the vertical and horizontal crawl is a process of removing childs of each newly merged cluster from the distance matrix '*CluArr'*, and the clusters pointers array '*ClurPntr'*. The new cluster pointer is added to '*ClurPntr'*. Crawling is a shifting of the remaining clusters over the deleted ones in both vertical and horizontal directions; see Pseudo code lists (4.23) and (4.24).

Pseudo code list (4.23) Vertical crawl routine

```
Input
        id: id. of cluster to be deleted
        DistArr: Distance matrix
Output
        DistArr: Distance matrix                {reduced}
        ClurPntr: Clusters Pointers array
Variables
        i,j: Distance matrix indices
Procedure
        For each i,j do                 {where i = id.. NoClu-2 , j = id+1.. NoClu-1}
                DistArr(i, j) ← DistArr(i+1, j)
        End For
        ClurPntr(i) ← ClurPntr(i+1)    { because here row(cluster) is deleted }
End
```

Pseudo code list (4.24) Horizontal crawl routine

```
Input
        id: id. of cluster to be deleted
        DistArr: Distance matrix
Output
        DistArr: Distance matrix                {reduced}
Variables
        i,j: Distance matrix indices
Procedure
        For each i,j do                 {where i = 0.. NoClu-2 , j = id.. NoClu-2}
                DistArr(i, j) ← DistArr(i, j+1)
        End For
End
```

4.7 Query Processing

The query search algorithms in this project use the indexing structure that was built during HAC process. The index array which contains the mother clusters and the binary tree associated with each one is the basis for query processing. Index array and binary trees both reference all clusters array '*CluArr'*. Once the index and trees are constructed for the whole database, they can be used to rapidly extract the images that are most similar to the query image. This is done by passing through the clusters and their sub-clusters that are closest to the query image. Each image is displayed through its image file name *(accompanied with the full path)* that refers to this individual image in the matched cluster.

4.7.1 Query Image Search

There are two ways; rather different, to search for query image using the binary trees resulting from the hierarchical index construct. These two ways are the Wide Search algorithm and the Narrow Search algorithm. The reason behind using these two different search algorithms is explained in the sample clusters shown in Figure (4.8).

In this example, a query corresponding image is located in mother cluster C_3. The Narrow Search algorithm will reach this particular cluster containing that image, the query result will be determined by cluster 3 only. So the retrieval result will be a subset or all of C_3 elements according to the cluster size and the desired number of retrieved images from any query instance. Sometimes, some of images that are closer to the query image are out of C_3; eventually, they are located in C_1, C_2, and C_4. Wide Search algorithm is adopted to handle this image retrieval problem, where more than one mother cluster is chosen according to its distance from query image.

Here, the distance is computed between the query image and the centroids of all mother clusters. A subset of clusters with minimum distance to image query is chosen, and all images in these clusters are compared exhaustively with the query image. Images in these clusters are ranked according to their distance from query. Then, a subset of these images (that are most similar to the query image) is displayed as the query result.

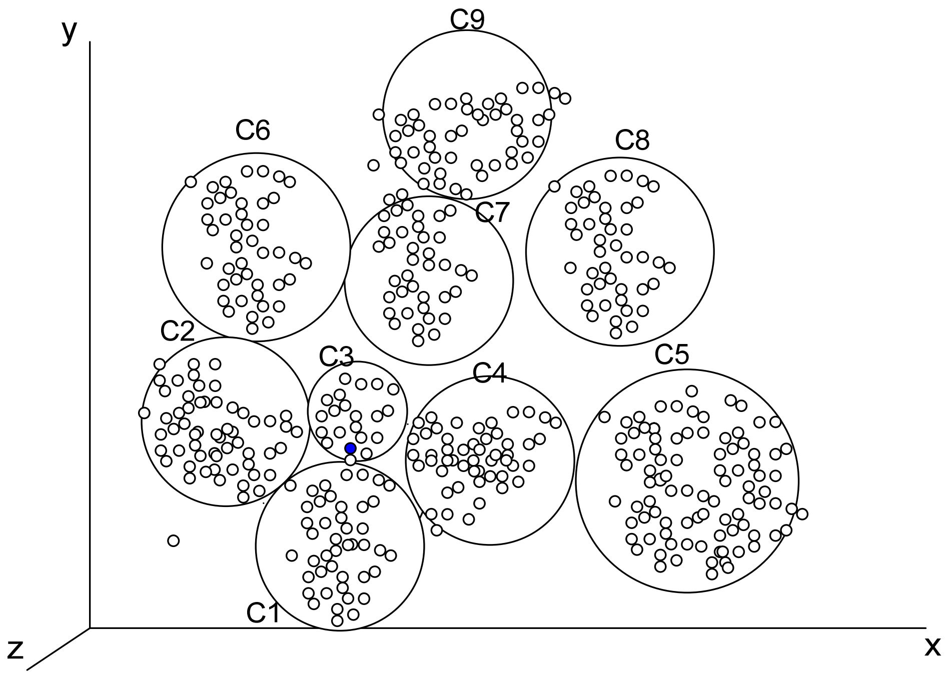


Figure (4.8) A sample of Hierarchical Agglomerative Clustering

1. Narrow Search scheme

The retrieval procedure used in this method is as follows. First, a features vector of the query image is computed. Then, the search starts by comparing the features vector of query image with all mother clusters centroids. The cluster with the closest centroid is selected, and the search continues down to its binary tree childs; as depicted in Figure (4.9).

Beginning with the root node of the selected mother cluster, the algorithm computes the distances between the query image and the centroids of the two clusters represented by the childs nodes; left and right. The closest child is chosen and the search continues recursively underneath its node. The search stops when the calculated distance reaches a predefined threshold or when number of images in the current tested cluster is equal to the predefined number of images that could be retrieved by query. The pointers to the individual images stored in the selected node are then used to access and display the retrieved images to the user.

Find closest mother cluster to query is a simple sequential search over mother clusters centroids. This routine results in finding the mother cluster closest to the query image.

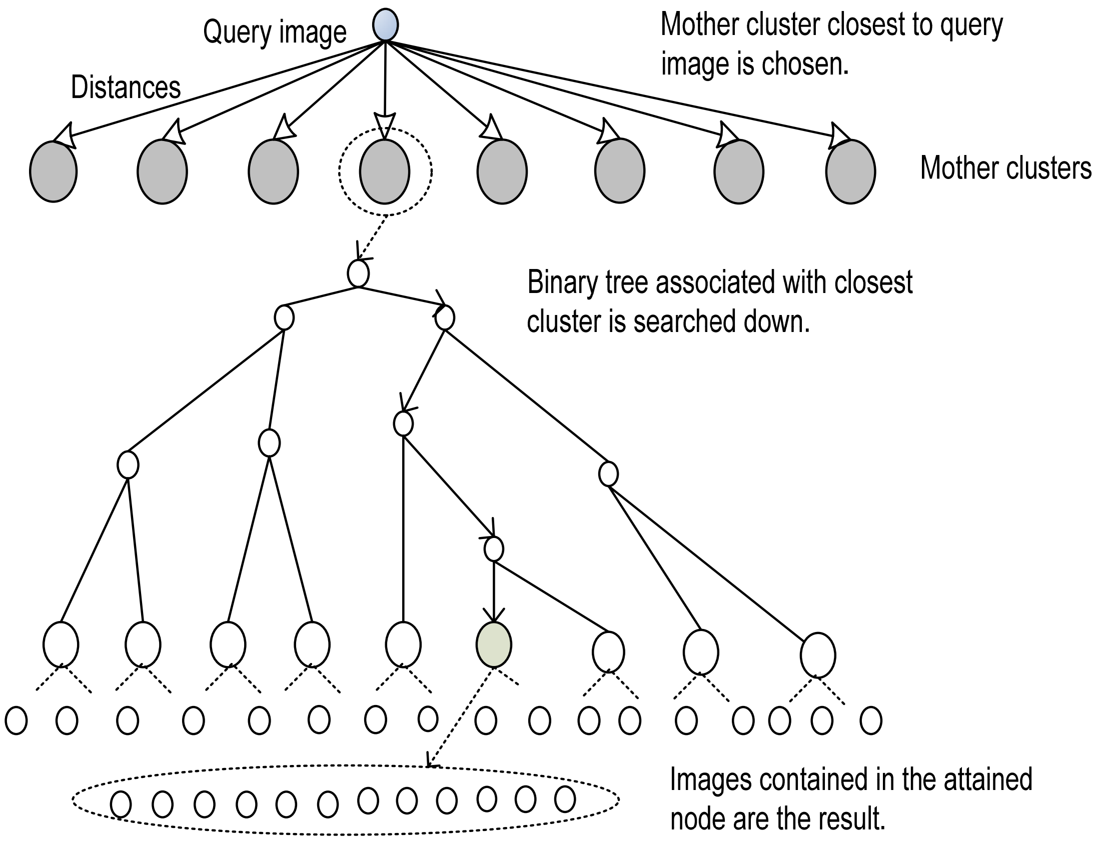

Query image
Distances
Mother cluster closest to query image is chosen.
Mother clusters
Binary tree associated with closest cluster is searched down.
Images contained in the attained node are the result.

Figure (4.9) Narrow search scheme

The mother cluster number is then sent to the recursive *search within cluster* function that is presented in Pseudo code list (4.25), for retrieving the most similar images to query as explained in the previous paragraph.

Pseudo code list (4.25) Recursive search within cluster function

```
Input
    Match: index of mother cluster closest to query
    NoElmInMatchClu: number of element wanted in small cluster
Output
    Match: index of a cluster (underneath the mother cluster) closest to query
Variables
    LefDist: Distance of query image from the Left child
    RigDist: Distance of query image from the Right child
Function
    If CluArr(Match).NoOfEle > NoElmInMatchClu Then
        { it is a large merged cluster; it should be fragmented}
```

continued

```
        LefDist ← EucleadianDist(Query,CluArr(Match).Left)
        RigDist ← EucleadianDist (Query,CluArr(Match).Right)
        If LefDist < RigDist Then                { Left child is closer }
          SearchWithinCluster ← SearchWithinCluster(CluArr(Match).Left)
        Else                                     { Right child is closer }
          SearchWithinCluster ← SearchWithinCluster(CluArr(Match).Right)
        End If
    Else
        Return Match
    End If
End
```

2. Wide Search scheme

All mother cluster centroids within the index array are compared and prioritized according to their distance from the query image descriptors. The cluster containing the centroid with the shortest distance to the query descriptor is the first one to be examined. All images in this cluster are compared with query image. Then the cluster with the second shortest distance to image query is taken next and so on; as depicted in Figure (4.10).

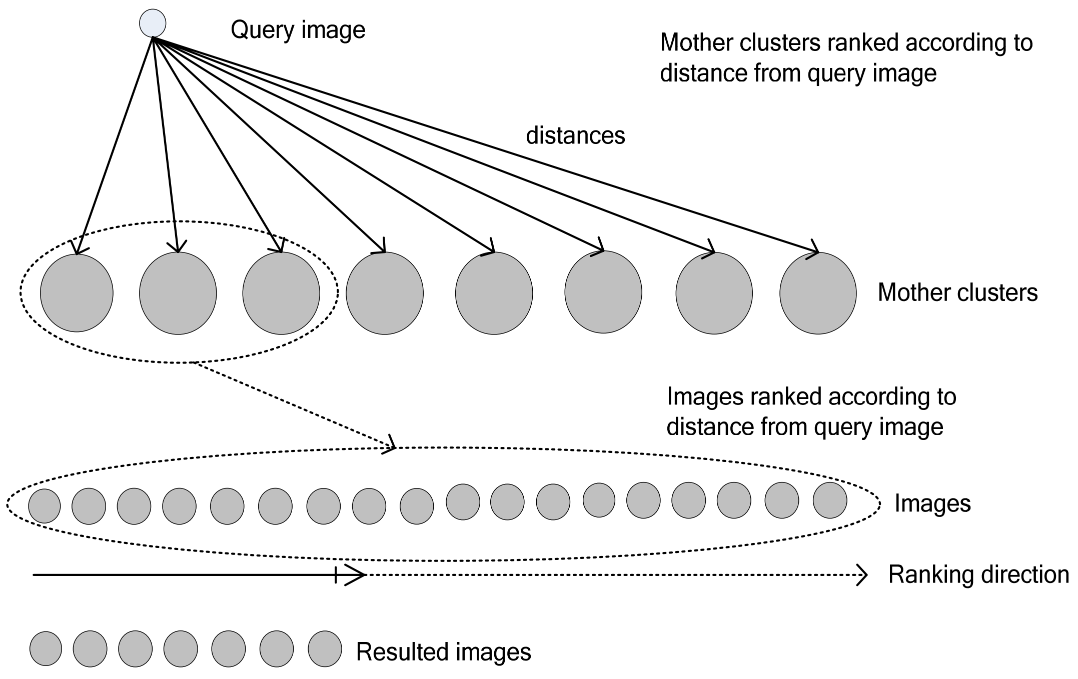

Query image
Mother clusters ranked according to distance from query image
distances
Mother clusters
Images ranked according to distance from query image
Images
Ranking direction
Resulted images

Figure (4.10) Wide search scheme

During the process of comparison between nearest neighbor clusters and query image, an array of size *n* (where *n* is the number of required images from query) is built and ranked according to distance from query. This list is displayed, in the final stage, to the user as the query result.

A part of the algorithm concerning finding the subset of closest mother clusters to query image is shown in Pseudo code list (4.26).

Pseudo code list (4.26) Find closest mother clusters subset to query

```
Input
      ClurPntr: Distance Matrix Cluster Pointer
      NoClu: number of clusters yielded from merge
Output
       CluList: Clusters List sorted according to distance from Query
Variables
      i: clusters index
Procedure
      For all i  do                    { where i= 0 .. NoClu - 1 }
            CluList(i).Dist ←  EuclideanDist(Query, ClurPntr (i))
            CluList(i).Match ←  ClurPntr (i)
      End For
      InSort (CluList, NoClu)   {sort clusters according to distance from Query}
End
```

According to their distance from query, clusters and images sorting is always done by the insertion sort. It is the fastest sorting method for number of elements<=25 [80]. The second part of the algorithm that is concerned with the searching within closest clusters subset is described in Pseudo code list (4.27).

```
Pseudo code list (4.27) Search within closest clusters subset
Input
    CluList: Clusters List sorted according to distance from Query:
    CluArr: Clusters Array; one dimensional array of records (clusters)
Output
    ImgList: Images List sorted according to distance from Query
Variables
    i: Clusters index
    j: index for images in on cluster
    Match: Cluster identifier matching given image
    ImgCount: Count of images closest to Query.
Procedure
      ImgCount ← 0
      For all i, j do      {where i= 0 .. DesNoClu  (desired number of clusters) and j= 0 ..
                           CluArr(CluList(i).Match).NoOfImg - 1 ;
                           j  is number of images within cluster }
            Match ← FindMatch(CluArr(CluList(i).Match).ImgArr(j))
            ImgList(ImgCount).Dist ← EuclideanDist(Query, Match)
            ImgList(ImgCount).Match ← Match
            ImgCount ← ImgCount + 1
    End For
    InSort (ImgList, ImgCount){sort images according to distance from Query}
End
```

Find Match function finds every image indicator in the clusters array *'CluArr'*, in order to reach the image data; especially its features vector. This is done by searching through singleton clusters only. Features vectors are used in comparison between query image and other images.

4.7.2 Clusters Exploring

Clusters built during HAC process can be explored and the images contained within each cluster are visualized. This is done by the recursive show clusters routine illustrated in Pseudo code list (4.28). A call to this routine will cause an increment in the counter of images displayed yet in a window and a decrement in the number of remaining images in that cluster. Increment and decrement steps are made by one in each call.

Pseudo code list (4.28) Recursive show clusters routine

```
Input
        i: number of cluster to be shown
        Counter: counter of images displayed
        No: number of images in cluster
Output
        Cluster images are displayed
Variables
        No variables used
Procedure
    If (No >= 0) and (Counter <= 18) Then    {18 images in window }
            DisplayImage(CluArr(i).ImgArr(No))
            DisplayName(CluArr(i).ImgArr(No))
            ShowCluster (i, Counter + 1, No - 1)            { recursive call }
    ElseIf (No >= 0) and (Counter > 18) Then
            Prompt 'This cluster has more than one window'
            Clear Images in this window
            ShowCluster (i, 1, No)                          { recursive call }
            ElseIf (No < 0) then                            { do nothing, i.e,
                                                            "stop recursive call"}
    End If
End
```

4.8 Features Combinations Based Retrieval

Several retrieval methods have been adopted and applied separately. These methods imply the use of Co-occurrence Matrix *'CO'*, Correlogram *'CR'*, Cumulative Histogram *'CH'*, and Moments *'Mom'*. Also, different combinations of them were experimented with. The studied features combinations schemes are listed in Table (4.3). The determination of these combined schemes imposes some modifications upon the different routines and functions mentioned above, where these codes assume single features type for simplicity.

Table (4.3) Features combinations schemes

No. of Methods	**CO**	**CR**	**CH**	**Mom**	**Retrieval Method**
1	√				CO
1		√			CR
1			√		CH
1				√	Mom
2	√		√		CO + CH
2		√	√		CR + CH
2	√			√	CO + Mom
2		√		√	CR + Mom
2			√	√	CH + Mom

The modifications needed to implement the combined retrieval schemes could be summarized in the following points:

1. Another features extraction routine should be implemented for the added features type. These added features are stored in another features vector which belongs to the same cluster.
2. Two distance matrices were used; one for each features vector. They will be unified into one matrix by computing the average of the corresponding entries in the two matrices.
3. When a pair of clusters is merged, two centroids should be computed for the new cluster; one for each of the two features vector.
4. The measurement of the new cluster distance, relative to all remaining clusters, should be done by using the corresponding two features vectors.
5. In the query processing stage, the two features vectors should be extracted from the query image and take place in comparison and distance computation along the index structure.

Chapter Five

Retrieval Performance Evaluation

5.1 Introduction

This chapter describes two lines of experiments. First, a test is done on gray images to determine the optimal configuration for the co-occurrence matrix features in a texture analysis task; similar analysis is done to evaluate the correlogram performance on colored textures. Second, an automatic evaluation module is built to assess image retrieval task using different color models with different quantization schemes applied with the features extraction methods.

As a whole, a total of 164 different configurations were evaluated to assess their suitability for color/texture based retrieval tasks using Hierarchical Agglomerative Clustering.

5.2 Automatic Evaluation Module

Once a content-based image retrieval application has been developed, the next crucial step is how to evaluate its performance, in terms of both retrieval effectiveness and complexity. In order to measure the effectiveness of the algorithms, precision and recall metrics have been used which provide a clear, overall indicator to their performance.

Several features extraction schemes were investigated separately. In addition, different combinations of them could also be experimented. More than single distance metrics was applied according to selected features, while the index construction task by using HAC algorithm is the same for all retrieval schemes.

Two searching algorithms for specific query type were compared to search through this index structure. They are the *Wide* search and *Narrow* search algorithms. In conclusion, multiple retrieval schemes were experimented and compared along the evaluation process.

Taking the retrieved images resulting from each retrieval scheme; an automatic evaluation module was built to assess its retrieval performance. In this module (see Figure 5.1) a set of 12 images are used as queries in each test. The Precision (P) and Recall (R) metrics from each query result have been averaged across the whole test. Samples of P&R values resulting from a single query are given in Appendix A (Table A.1). A user enters a query image describing the desired information. The system returns a list of images– begins with exact match, followed by a ranked list according to relevance. Ranking is based on the similarity of the query to the images set.

By using binary relevance judgments, any image is either relevant or irrelevant to the query image (i.e., relevant=1, irrelevant=0). In order to improve the evaluation, the retrieval process is performed several times for different randomly picked query images, since different queries may lead to different precision and recall values.

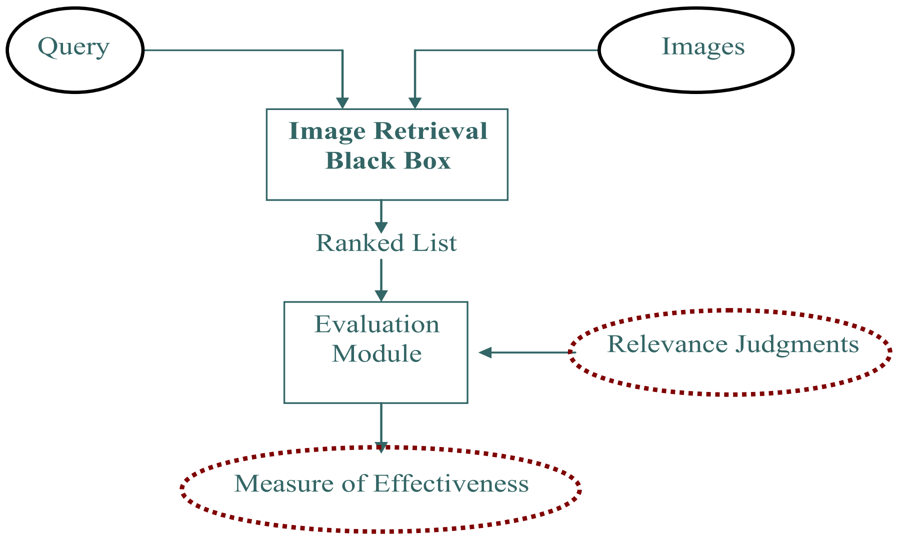


Figure (5.1) Automatic evaluation module

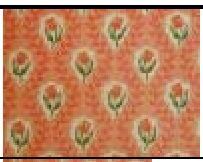
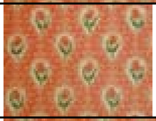
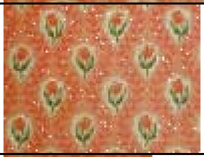
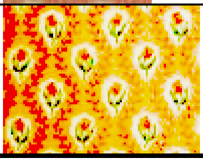
In the evaluation experiments, a set of 26 different images was selected. From each image in this set, a new set of 6 images was generated by adding noise, up-scaling, down-scaling, rotation, and exposure variation. This process establishes a set of 156 tested images. Each original image will have six variations (versions) considered as a single cluster. The parameters that control the generated images are repeated for the whole set. These parameters include the percentage of up/down scaling, the range of the uniform noise, the rotation angle, and the changes in exposure components (Hue, Saturation, and Value). One of the base images and its variations are shown in Table (5.1).

All sample images have been clustered by the system. Single cluster and one query result are shown in Figure (5.2). More results are listed in Appendix B.

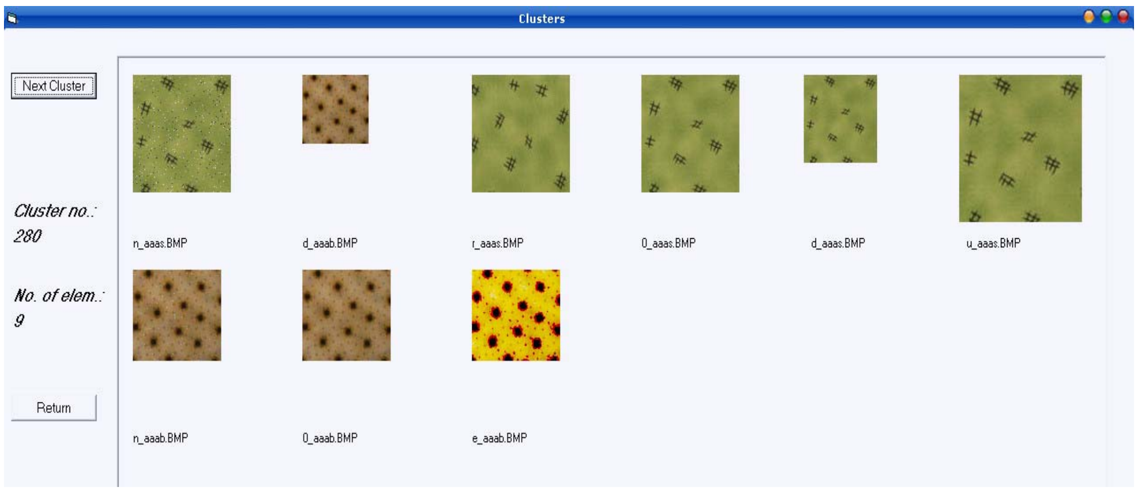
5.3 Retrieval Efficiency and Effectiveness

The two major aspects of retrieval evaluation are efficiency and effectiveness. Efficiency is expressed in terms of system speed. Speed is rather technical and relatively easier to evaluate. Effectiveness is much more difficult to judge. It evaluates "how good is the result". It is related to the relevancy of retrieved items to the query image. Focus will be on effectiveness evaluation.

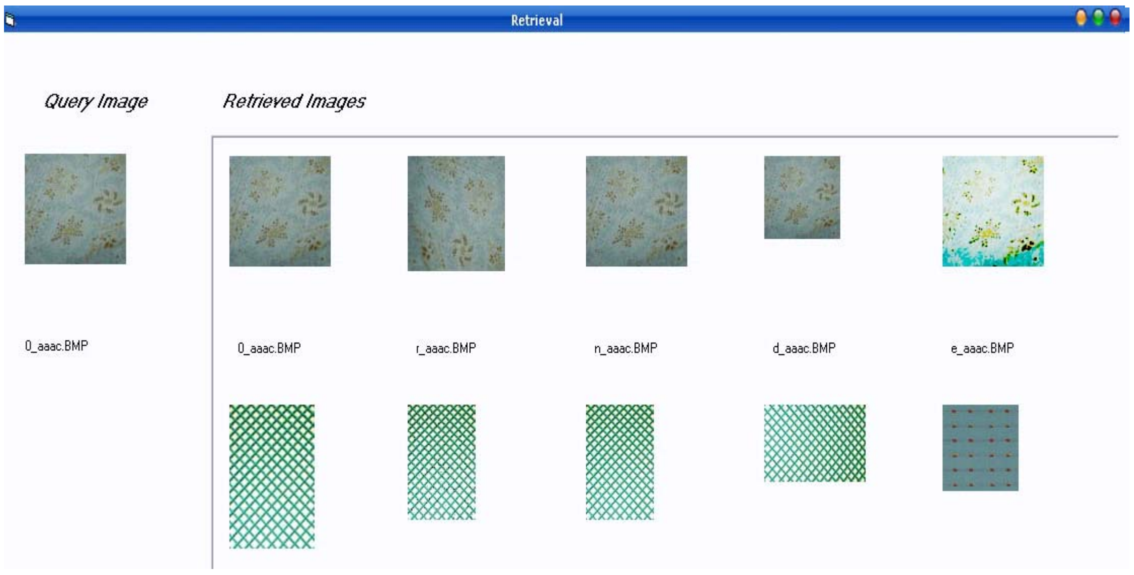
Table (5.1) Generated images for a single cluster

Image	**Image id.**	**Description**
	0_aaaq	**o**riginal image
	d_aaaq	**d**own-scaled
	u_aaaq	**up**-scaled
	n_aaaq	**n**oise-added
	r_aaaq	**r**otated
	e_aaaq	**e**xposure-varied

An important notion in CBIR task is the complexity of the retrieval process. This can be interpreted by time, computational, and storage complexity. Time complexity denotes the time needed to conduct the required calculations. Computational complexity is defined as the number of steps or arithmetic operations in which the process is done. Storage complexity refers to the memory space required during the execution of the process.



(a)



(b)

Figure (5.2) Sample retrieval output;

(a) A cluster resulted by applying GLCM− 8 bins scheme based on LAB.

(b) A query result by applying correlogram− 6×3×3 bins & cumulative histogram− 18×3×3 bins scheme based on HVC.

Time complexity depends on both software (i.e., computational complexity) and hardware constraints (e.g., speed and access-time of hard-disks). Concerning the hardware, it is fully dependent on the industrial developments.

For CBIR purposes, the computational complexity mainly depends on the size of the feature representation and on the matching algorithm for these features. In realistic applications, the features for each image listed in the database are calculated offline. Next, only query image retrieval based on these features is performed at operating. As a method of comparison, time complexity may suffice as an indicator for algorithm efficiency.

5.4 Single Value Measures

Sometimes it is convenient to encapsulate precision and recall values into a single value that gives a compact and easy to understand clue about any retrieval scheme performance.

In addition to the Mean Average Precision (MAP) and F-measure that were presented in section (3.7), R-precision is another metric which refers to the Rth position in the ranking of results for a query that has R relevant images. This is indicated in the sample query result shown in Table (5.2).

Table (5.2) Precision at R relevant images

n	image #	relevant
1	58	√
2	89	√
3	76	
4	90	√
5	86	
6	92	√
7	84	
8	88	
9	57	
10	98	
11	10	
12	91	
13	77	√
14	99	

R = no. of relevant images = 6

R-Precision = 4/6 = 0.67

One can fix the number of images retrieved at several levels (for example: top 5 and top 10). Afterward, precision is measured at each of these levels. R-Precision is computed over multiple queries at 5 and 10 retrieved images in Table (5.3). More details about precision and recall computation are found in Appendix A.

Table (5.3) Precision at different relevant images

f_i : image file

✓ : relevant

✗: Irrelevant

	Query 1	Query 2	Query 3
	f1 ✓	f10 ✗	f6 ✗
	f2 ✓	f9 ✗	f1 ✓
	f3 ✓	f8 ✗	f2 ✓
	f4 ✓	f7 ✗	f10 ✗
	f5 ✓	f6 ✗	f9 ✗
	f6 ✗	f1 ✓	f3 ✓
	f7 ✗	f2 ✓	f5 ✓
	f8 ✗	f3 ✓	f4 ✓
	f9 ✗	f4 ✓	f7 ✗
	f10 ✗	f5 ✓	f8 ✗
Precision at 5 →	**1.0**	**0.0**	**0.4**
Precision at 10 →	**0.5**	**0.5**	**0.5**

The MAP and F-measure is adopted in the evaluation phase of this research. The average precision of a single query at 6 retrieved images is presented in Table (5.4).

Table (5.4) Performance measures computed for a single query.

n	Relevant	Recall	Precision	F-Measure	image id.
1	Y	0.17	1.00	0.29	0_aaai.BMP
2	Y	0.33	1.00	0.50	d_aaai.BMP
3	Y	0.50	1.00	0.67	r_aaai.BMP
4	Y	0.67	1.00	0.80	n_aaai.BMP
5	Y	0.83	1.00	0.91	e_aaai.BMP
6	N	0.83	0.83	0.83	u_aaav.BMP
7	N	0.83	0.71	0.77	r_aaat.BMP
8	N	0.83	0.63	0.71	0_aaat.BMP
9	N	0.83	0.56	0.67	n_aaat.BMP
10	N	0.83	0.50	0.63	d_aaat.BMP

Average P at 6 retrieved images = 0.97
Query image is 0_aaai.BMP

Twelve queries were used in each test. The MAP over these queries is computed as a final mark of each scheme performance. This is explained in Table (5.5).

Table (5.5) MAP over multiple queries for different retrieval schemes

Retrieval Shceme	**Average Precision at 6 retrieved images**				**MAP**
	Q1	**Q2**	-------------------	**Q12**	
Scheme 1	0.91	0.97		0.84	**0.94**
Scheme 2	0.82	0.94		0.97	**0.90**
⋮					
Scheme 9	0.88	0.92		0.94	**0.92**

5.5 GLCM and Color Correlogram Features Selection

There are four Gray Level Co-occurrence Matrices (GLCM); *ar0, ar45, ar90,* and *ar135*, for angles 0, 45, 90, and 135 respectively. Therefore, each co-occurrence matrix feature has four feature descriptors in the computed features vector. This is called a feature group. Sometimes, in texture analysis, it is convenient to use the distances between features groups midpoints (groups' averages or centroids) instead of the distances of the features individually.

Co-occurrence matrix features distance can be computed in rather two different ways; *feature average* distance and *feature group* distance as introduced in articles 1 and 2 in subsection (4.6.2) respectively. Feature average scheme gives a compact features vector where each four feature descriptors are encapsulated into a single one (average). The two distance computation schemes were compared along each retrieval test when it was found applicable. The same route was followed for the color correlogram.

In this assessment, a combination of best texture features which have performed best for gray texture analysis is determined. Concerning the GLCM, the intensity values are quantized into 4, 8, 16, 32, or 64 bins using five color spaces. No instance has indicated that the use of a large number of bins gave a better result. This is consistent with the notion of the existence of a limited number of color categories in which humans represent colors. The co-occurrence matrix is calculated with distances 1, 2, 3, and 4. The most suitable distance is 2 according to the selected texture domain.

The intensity-based co-occurrence matrix features were tested over different color spaces with different quantization schemes. The Gray color model quantized into 8 bins performed best with a classification performance of 92% using *feature group* distance (as shown in Table 5.6), and 88% by using *feature average* distance (as shown in Table 5.7). The chosen quantization scheme is important; that is, a lower number of bins leads to better performance.

Table (5.6) Classification performance for the GLCM (with Gray− 8 bins) by using feature group distance

f.	**f. no.**	**correct classification percentage according to features range (r)**							
		one f.	**two fs.**	**three fs.**	**four fs.**	**five fs.**	**six fs.**	**seven fs.**	**eight fs.**
ASM	1	0.69 r=[1]	0.77 r=[1,2]	0.80 r=[1,3]	0.80 r=[1,4]	0.80 r=[1,5]	0.84 r=[1,6]	**0.92** r=[1,7]	0.88 r=[1,8]
IDM	2	0.65 r=[2]	0.73 r=[2,3]	0.80 r=[2,4]	0.84 r=[2,5]	0.88 r=[2,6]	**0.92** r=[2,7]	**0.92** r=[2,8]	
COR	3	0.65 r=[3]	0.77 r=[3,4]	0.77 r=[3,5]	0.84 r=[3,6]	0.88 r=[3,7]	0.88 r=[3,8]		
SHD	4	0.73 r=[4]	0.77 r=[4,5]	0.80 r=[4,6]	0.84 r=[4,7]	0.84 r=[4,8]			
PRM	5	0.61 r=[5]	0.73 r=[5,6]	0.77 r=[5,7]	0.84 r=[5,8]				
ENT	6	0.73 r=[6]	0.69 r=[6,7]	0.77 r=[6,8]					
HOM	7	0.65 r=[7]	0.73 r=[7,8]						
CON	8	0.61 r=[8]							

f=feature fs=features r=features range

Table (5.7) Classification performance for the GLCM (with Gray− 8 bins) by using feature average distance

f.	**f. no.**	**correct classification percentage according to features range (r)**				
		four fs.	**five fs.**	**six fs.**	**seven fs.**	**eight fs.**
ASM	1	0.73 r=[1,4]	0.77 r=[1,5]	0.80 r=[1,6]	**0.88** r=[1,7]	0.84 r=[1,8]
IDM	2	0.77 r=[2,5]	0.84 r=[2,6]	**0.88** r=[2,7]	0.84 r=[2,8]	
COR	3	0.73 r=[3,6]	0.80 r=[3,7]	0.84 r=[3,8]		
SHD	4	0.73 r=[4,7]	0.77 r=[4,8]			
PRM	5	0.69 r=[5,8]				
ENT	6					
HOM	7					
CON	8					

Regarding the color correlogram, several quantization schemes were applied to the five color spaces, and the efficient human-based 9 colors and 11 colors categories. The HVC color model quantized into 18×3×3 bins and 6×3×3 bins gave best classification result: 92% when using *feature group* distance (as shown in Table 5.8), and 88% when using *feature average* distance (as shown in Table 5.9).

Table (5.8) Classification performance for the color correlogram with (HVC− 18×3×3 bins) using feature group distance

f.	f. no.	correct classification percentage according to features range (r)							
		one f.	**two fs.**	**three fs.**	**four fs.**	**five fs.**	**six fs.**	**seven fs.**	**eight fs.**
ASM	1	0.38 r=[1]	0.69 r=[1,2]	0.77 r=[1,3]	0.80 r=[1,4]	0.84 r=[1,5]	**0.92** r=[1,6]	0.88 r=[1,7]	0.88 r=[1,8]
IDM	2	0.54 r=[2]	0.77 r=[2,3]	0.84 r=[2,4]	0.84 r=[2,5]	0.88 r=[2,6]	**0.92** r=[2,7]	**0.92** r=[2,8]	
COR	3	0.38 r=[3]	0.69 r=[3,4]	0.77 r=[3,5]	0.84 r=[3,6]	0.88 r=[3,7]	0.88 r=[3,8]		
SHD	4	0.54 r=[4]	0.69 r=[4,5]	0.80 r=[4,6]	0.84 r=[4,7]	0.84 r=[4,8]			
PRM	5	0.54 r=[5]	0.73 r=[5,6]	0.77 r=[5,7]	0.77 r=[5,8]				
ENT	6	0.58 r=[6]	0.73 r=[6,7]	0.73 r=[6,8]					
HOM	7	0.38 r=[7]	0.54 r=[7,8]						
CON	8	0.54 r=[8]							

Table (5.9) Classification performance for the color correlogram with (HVC− 6×3×3 bins) by using feature average distance

f.	f. no.	correct classification percentage according to features range (r)				
		four fs.	**five fs.**	**six fs.**	**seven fs.**	**eight fs.**
ASM	1	0.73 r=[1,4]	0.77 r=[1,5]	**0.88** r=[1,6]	**0.88** r=[1,7]	0.84 r=[1,8]
IDM	2	0.77 r=[2,5]	0.80 r=[2,6]	**0.88** r=[2,7]	**0.88** r=[2,8]	
COR	3	0.77 r=[3,6]	0.84 r=[3,7]	0.84 r=[3,8]		
SHD	4	0.73 r=[4,7]	0.80 r=[4,8]			
PRM	5	0.69 r=[5,8]				
ENT	6					
HOM	7					
CON	8					

The classification task was performed by using different possible combinations of GLCM features. Six of the eight used features are found to be the strongest descriptors of texture. They are the Inverse Difference Moment (IDM), Haralick's Correlation (COR), Cluster Shade (SHD), Cluster Prominence (PRM), Entropy (ENT), and Homogeneity (HOM). Using all these features in one time may result in a lower classification percentage.

5.6 Comparison of Image Retrieval with Different Schemes

Several configurations have been investigated; some of the image retrieval experiments are listed in the following subsections. The key factors of each configuration can be described by the color/texture analysis algorithm, the color space, and the used quantization scheme. For all features extraction methods, the color spaces HVC, HSV, HLS, LAB, RGB, and Gray were considered. In addition, both the cumulative histogram and the color correlogram are experimented with the human-based 9 colors and 11 colors categories. It is investigated that the former proved to have a good performance.

The GLCM, color correlogram, cumulative histogram, and color moments as features extraction methods were applied. In addition, different combinations of these methods were experimented with. Since GLCM and color correlogram are computed in two different ways as explained in section (5.5), these two ways (feature group and feature average) are experimented in and compared along the test process. Also, the results of wide and narrow search algorithms were compared for the case of feature group distance.

107 different configurations were tested using singular schemes:

25 for the GLCM,
39 for the color histogram,
38 for the color correlogram, and
5 for the color moments,

while **57** configurations were tested using combined schemes:

12 for GLCM with cumulative histogram scheme,
10 for GLCM with moments scheme,
19 for color correlogram with cumulative histogram scheme,
10 for color correlogram with moments scheme, and
6 for cumulative histogram with moments scheme.

In total, **164** configurations were experimented with along the evaluation process. MAP is used as a performance measure for each configuration. It is shown later that this measure is very close to F-measure as depicted in Figure (5.3). Also, the computation time of every scheme and its query response time are measured as an efficiency grade.

5.6.1 Singular Retrieval Schemes

When the co-occurrence matrix is used, only the intensity of the color is taken into account. Intensity is quantized into 4, 8, 16, 32, or 64 bins. One interesting result confirms the fact that using more bins did not improve performance for a given color model. In this case, coarse color quantization is preferable because it is computationally cheap.

For the cumulative histogram and color correlogram, hue is the most significant characteristic so this component gets the finest quantization (from 2 to 32 bins). While saturation and intensity get coarse quantization (up to 3 bins) to match human color perception, more tolerance to these components deviations is possible. Also, this coarse quantization helps to keep a reasonable computing time. As presented in the tables shown below, the total number of bins augments the multiplication of the three color components by three. This three is added for the undefined hue value where quantized intensity value is used instead and added to the total number of bins (see subsection 4.4.2 and equation 4.2).

No quantization is used when computing the moment's features. As a comprehensive view the wide search algorithm performs better than the narrow search algorithm for a given scheme. Narrow search performs well in few and exceptional cases only.

1. GLCM Scheme

By inspecting Table (5.10), it is clear that Gray color space quantized into 4 bins leads to the best performance with MAP = 92% when using feature group distance and wide search. While when GLCM was used with feature average distance, it gives a poor performance (with MAP = 82%, for Gray− 8 bins), as shown in Table (5.11).

Table (5.10) Retrieval performance for the GLCM using feature group distance

Color Space	Quan. Levels	Comp. Time(Sec)	Wide Search		Narrow Search	
			MAP	Query Time(Sec)	MAP %	Query Time(Sec)
HVC	4	19.7	0.87	0.2188	0.79	0.1875
	8	16.1	0.87	0.2188	0.72	0.1875
	16	16.6	0.87	0.2031	0.67	0.2188
	24	20.8	0.87	0.2188	0.67	0.2188
	32	18.6	0.87	0.2188	0.87	0.2188
	64	28.9	0.87	0.2813	0.67	0.2813
HSV	4	11.5	0.89	0.1250	0.78	0.125
	8	13.9	0.89	0.1250	0.81	0.1094
	16	12.1	0.89	0.1250	0.82	0.125
HLS	4	11.8	0.91	0.1250	0.9	0.125
	32	14.3	0.91	0.1250	0.8	0.2094
LAB	4	13.9	0.87	0.1406	0.79	0.125
	8	14	0.87	0.1406	0.72	0.125
	16	14.3	0.87	0.1406	0.67	0.1406
	32	16.3	0.87	0.1563	0.87	0.1406
	64	26.7	0.87	0.2188	0.67	0.2188
Gray	**4**	**8.22**	**0.92**	**0.0781**	0.84	0.0781
	8	8.3	0.92	0.0781	0.82	0.8215
	16	8.73	0.92	0.0781	0.86	0.0781
	32	10.7	0.91	0.1094	0.86	0.0938

Table (5.11) Retrieval performance for the GLCM using feature average distance

Color Space	Quan. Levels	Comp. Time(Sec)	Wide Search	
			MAP %	Query Time(Sec)
HVC	8	18.98	0.74	0.2031
HSV	8	11.61	0.79	0.1094
HLS	8	11.70	0.78	0.1094
LAB	8	13.64	0.74	0.1250
Gray	**8**	**8.19**	**0.82**	**0.0781**

2. Color Correlogram Scheme

HVC quantized into 6×3×3 bins gives the best result (MAP=0.91%) when using wide search with feature average distance, or narrow search with feature group distance, as indicated in Tables (5.12) and (5.13) respectively. While when HVC is quantized into 32×2×2 bins, it gives the same percent by using wide search for feature group distance. An interesting point is that 9 colors category has the efficient execution time with a tiny MAP difference. This makes it as a good choice for real applications.

Table (5.12) Retrieval performance for the color correlogram using feature group distance

Color Space	**Quan. Levels**			**# of bins**	**Comp. Time**(Sec)	**Wide Search**		**Narrow Search**	
	1st comp.	**2nd comp.**	**3rd comp.**			**MAP %**	**Query Time**(Sec)	**MAP %**	**Query Time**(Sec)
HVC	2	2	2	11	22.63	0.88	0.2500	0.78	0.2344
	3	2	2	15	20.48	0.89	0.2500	0.89	0.2500
	3	3	3	30	22.16	0.90	0.2656	0.82	0.2500
	6	2	2	27	21.70	0.90	0.2500	0.83	0.2656
	6	**3**	**3**	57	29.67	0.90	0.2969	**0.91**	0.3125
	12	2	2	51	27.44	0.90	0.2969	0.90	0.2969
	12	3	3	111	71.64	0.90	0.5625	0.91	0.5781
	18	2	2	75	39.19	0.90	0.3750	0.90	0.3594
	18	3	3	165	164.77	0.91	1.2031	0.90	1.1719
	24	2	2	99	58.44	0.90	0.4844	0.89	0.5000
	24	3	3	219	332.28	0.91	2.2500	0.78	2.2500
	32	**2**	**2**	131	**99.14**	**0.91**	0.7656	0.90	0.7344
	32	3	3	291	705.83	0.91	4.6406	0.90	4.6406
HSV	3	2	2	15	15.86	0.85	0.1563	0.84	0.1563
	6	3	3	57	24.94	0.86	0.2188	0.75	0.2031
	12	2	2	51	22.83	0.87	0.2031	0.87	0.2188
	18	3	3	165	159.86	0.89	1.0938	0.75	1.1094
	32	2	2	131	97.11	0.87	0.6719	0.78	0.6563
HLS	6	2	2	27	17.08	0.86	0.1563	0.77	0.1563
	12	3	3	111	66.92	0.89	0.4844	0.74	0.4844
	18	2	2	75	34.83	0.88	0.2813	0.78	0.2656
LAB	2	2	2	8	15.05	0.88	0.1563	0.89	0.1406
	3	3	3	27	16.53	0.79	0.1563	0.57	0.1406
	4	4	4	64	27.64	0.82	0.2344	0.67	0.2344
	5	5	5	125	83.61	0.84	0.5938	0.65	0.5625
RGB	2	2	2	8	13.97	0.90	0.1406	0.69	0.1406
	3	3	3	27	15.52	0.89	0.1563	0.84	0.1406
	4	4	4	64	26.50	0.89	0.2344	0.88	0.2188
	5	5	5	125	89.53	0.89	0.5781	0.88	0.5625
11Col				11	22.31	0.88	0.2813	0.80	0.2500
9Col				9	9.63	0.89	0.0938	0.82	0.0938

3. Cumulative Histogram Scheme

Firstly, the similarity between histograms is measured by using L_1, L_2 metrics, and the histogram intersection method. They yield 92%, 90%, and 82% correct classification, respectively. L_1 is found to be the best, so it is used in all the following histogram comparison calculations. The results of the performed tests to investigate the image retrieval task on global color histograms with different quantization resolutions in the selected color spaces are listed in Table (5.14).

Table (5.13) Retrieval performance for the color correlogram using feature average distance

Color Space	Quan. Levels			# of bins	Comp. Time(Sec)	Wide Search	
	1st comp.	2nd comp.	3rd comp.			MAP %	Query Time(Sec)
HVC	**6**	**3**	**3**	57	30.16	**0.91**	0.2969
HSV	12	2	2	51	29.34	0.89	0.1885
	18	3	3	165	158.9	0.86	1.0781
HLS	6	3	3	57	24.95	0.84	0.2031
LAB	2	2	2	8	15.17	0.90	0.1406
RGB	2	2	2	8	14.17	0.80	0.1250
11Col				11	22.42	0.84	0.2656
9Col				9	9.672	0.88	0.0938

The best retrieval results are obtained when using HSV color space quantized into 18×2×2 bins, for such case the determined MAP=0.94 for both wide and narrow search. Here, again 9 colors category gives an excellent execution time with a small MAP difference.

4. Moments Scheme

This scheme gives poor performance, where the best obtained retrieval result is with MAP=0.83% for RGB color space by using wide search, as shown in Table (5.15).

5.6.2 Combined Retrieval Schemes

In this evaluation stage, the best retrieval schemes obtained from the previous stage are combined in an integrated manner. A similarity weight is assigned for each features type. It is possible that one type of features is more important than another, in such case, it should get a higher weight. In this research work, it is assumed that all features types are equally important (*weight = 0.5 for all types*). This weight could be changed according to the application field.

1. GLCM with Cumulative Histogram Scheme

HVC and LAB color spaces perform better than the others, they lead to MAP=0.94% when wide search and feature group distance are used, as presented in Tables (5.16) and (5.17). Processing in LAB space is faster than with HVC space.

Table (5.14) Retrieval performance for the cumulative histogram

Color Space	Quan. Levels			# of bins	Comp. Time(Sec)	Wide Search		Narrow Search	
	1st comp.	2nd comp.	3rd comp.			MAP %	Query Time(Sec)	MAP %	Query Time(Sec)
HVC	2	2	2	11	16.53	0.90	0.2188	0.83	0.2031
	3	3	3	30	16.48	0.92	0.2031	0.76	0.2031
	4	4	4	67	16.45	0.91	0.2188	0.83	0.2188
	5	5	5	128	16.52	0.92	0.2188	0.77	0.2031
	6	3	3	57	16.47	0.92	0.2031	0.91	0.2031
	12	3	3	111	18.66	0.92	0.2031	0.92	0.2031
	18	3	3	165	16.53	0.92	0.2031	0.92	0.2031
	24	3	3	219	16.50	0.92	0.2188	0.92	0.2031
	24	4	4	387	16.70	0.92	0.2188	0.92	0.2188
	32	3	3	291	16.53	0.92	0.2031	0.92	0.2031
HSV	2	2	2	11	11.70	0.92	0.1094	0.74	0.1094
	3	3	3	30	11.78	0.91	0.125	0.83	0.1094
	4	4	4	67	11.89	0.91	0.125	0.76	0.1094
	5	5	5	128	11.81	0.91	0.1094	0.91	0.1094
	6	3	3	57	15.53	0.92	0.125	0.92	0.1094
	6	6	6	219	11.83	0.92	0.1094	0.92	0.125
	12	2	2	51	11.88	0.92	0.1094	0.69	0.1094
	12	3	3	111	11.77	0.92	0.125	0.69	0.1094
	18	**2**	**2**	75	**11.64**	**0.94**	0.1094	**0.94**	0.125
	18	3	3	165	13.41	0.94	0.1094	0.93	0.1094
	32	2	2	131	11.73	0.94	0.1094	0.78	0.125
	32	3	3	291	11.70	0.94	0.125	0.78	0.1094
HLS	18	2	2	75	11.61	0.93	0.1094	0.93	0.1094
	18	3	3	165	11.67	0.93	0.1406	0.86	0.1094
	32	**2**	**2**	131	**11.69**	**0.94**	0.125	0.71	0.1094
	32	3	3	291	11.70	0.94	0.1406	0.62	0.1094
LAB	2	2	2	8	13.69	0.86	0.1094	0.86	0.1094
	3	3	3	27	11.19	0.84	0.1094	0.77	0.1094
	4	4	4	64	11.16	0.84	0.1094	0.76	0.1094
	5	5	5	125	11.23	0.84	0.1094	0.74	0.1094
	6	6	6	216	11.19	0.85	0.125	0.85	0.1094
RGB	2	2	2	8	10.36	0.90	0.094	0.87	0.094
	3	3	3	27	10.31	0.90	0.093	0.86	0.094
	4	4	4	64	10.33	0.90	0.094	0.91	0.11
	5	5	5	125	10.27	0.90	0.109	0.91	0.109
	6	6	6	216	10.33	0.90	0.11	0.85	0.093
11Col				11	18.50	0.75	0.1719	0.75	0.1719
9Col				9	5.73	0.93	0.0469	0.92	0.0469

Table (5.15) Retrieval performance for the moments

Color Space	Comp. Time(Sec)	Wide Search		Narrow Search	
		MAP %	Query Time(Sec)	MAP %	Query Time(Sec)
HVC	19.42	0.83	0.1719	0.73	0.1563
HSV	11.72	0.81	0.1094	0.59	0.1094
HLS	14.33	0.83	0.1250	0.50	0.1094
LAB	13.88	0.76	0.1406	0.56	0.1250
RGB	**10.30**	**0.83**	**0.0353**	0.73	0.0313

Table (5.16) Retrieval performance for the GLCM using feature group distance with cumulative histogram

Color Space	CO # of bins	CH Quan. Levels			CH # of bins	Comp. Time (Sec)	Wide Search		Narrow Search	
		1st comp.	2nd comp.	3rd comp.			MAP %	Query Time(Sec)	MAP %	Query Time(Sec)
HVC	**4**	**6**	**3**	**3**	57	26.69	**0.94**	0.2813	0.71	0.2656
	8	12	3	3	111	22.89	0.93	0.2969	0.80	0.2656
HSV	8	18	3	3	165	18.48	0.93	0.1875	0.67	0.1719
	8	6	3	3	57	18.44	0.91	0.2031	0.69	0.1719
HLS	8	18	3	3	165	18.55	0.92	0.2031	0.73	0.1875
LAB	**8**	**2**	**2**	**2**	8	17.88	**0.94**	0.1875	0.83	0.1719
RGB	8	2	2	2	8	17.45	0.93	0.1875	0.89	0.1875

Table (5.17) Retrieval performance for the GLCM using feature average distance with cumulative histogram

Color Space	CO # of bins	CR Quan. Levels			CH # of bins	Comp. Time(Sec)	Wide Search	
		1st comp.	2nd comp.	3rd comp.			MAP %	Query Time(Sec)
HVC	**8**	**12**	**3**	**3**	111	23.38	**0.93**	0.2822
HSV	8	18	3	3	165	18.53	0.91	0.1865
HLS	8	18	3	3	165	18.44	0.91	0.1865
LAB	**8**	**2**	**2**	**2**	8	18.00	**0.93**	0.1719
RGB	8	2	2	2	8	17.52	0.91	0.1885

2. GLCM with Moments Scheme

HVC− 8 bins, HLS− 8 bins, and RGB− 8 bins have the highest MAP values, when using feature group distance and wide search. Among them, processing under RGB color space is the faster with MAP=0.91%; as indicated in Tables (5.18) and (5.19).

Table (5.18) Retrieval performance for the GLCM using feature group distance with moments

Color Space	**Quan. Levels**	**Comp. Time**(Sec)	**Wide Search**	
			MAP %	**Query Time**(Sec)
HVC	8	26.09	0.91	0.2344
HSV	8	18.25	0.90	0.1719
HLS	8	18.33	0.91	0.1875
LAB	8	20.45	0.89	0.2031
RGB	**8**	**15.33**	**0.91**	**0.1563**

Table (5.19) Retrieval performance for the GLCM using feature average distance with moments

Color Space	**Quan. Levels**	**Comp. Time**(Sec)	**Wide Search**	
			MAP %	**Query Time**(Sec)
HVC	8	23.34	0.88	0.2197
HSV	**8**	18.41	**0.90**	0.1719
HLS	8	18.42	0.89	0.1729
LAB	8	20.39	0.87	0.1885
RGB	8	15.39	0.89	0.1406

3. Color Correlogram with Cumulative Histogram Scheme

As indicated in Tables (5.20) and (5.21), the configuration of the HVC color space quantized into 18×3×3 bins has the best performance (with MAP=0.93%) when using the feature group or the feature average distances, and wide search. The 9 colors category performs well with very less computation and query response times.

4. Color Correlogram with Moments Scheme

HVC with 6×3×3 bins and RGB with 2×2×2 bins have better performance (with MAP=0.88%) when using the feature group or the feature average distances and wide search, as indicated in Tables (5.22) and (5.23). Processing under RGB color space needs less computation time.

Table (5.20) Retrieval performance for the color correlogram using feature group distance with the cumulative histogram

Color Space	CR Quan. Levels			CR # of bins	CH Quan. Levels			CH # of bins	Comp. Time (Sec)	Wide Search	
	1st comp.	2nd comp.	3rd comp.		1st comp.	2nd comp.	3rd comp.			MAP %	Query TimeSec
HVC	6	3	3	57	12	3	3	111	36.86	0.92	0.3750
	6	3	3	57	6	3	3	57	39.36	0.92	0.3750
HSV	18	3	3	165	12	2	2	51	169	0.92	1.1563
	18	**3**	**3**	165	**18**	**3**	**3**	165	172	**0.93**	1.1875
HLS	6	3	3	57	18	2	2	75	33.25	0.92	0.2969
	6	2	2	27	18	3	3	165	26.52	0.92	0.2500
LAB	2	2	2	8	2	2	2	8	21.66	0.90	0.1875
RGB	4	4	4	64	4	4	4	64	36.63	0.89	0.3125
11Col				11				11	25.53	0.87	0.2344
9Col				9				9	9.78	0.91	0.1094

Table (5.21) Retrieval performance for the color correlogram using feature average distance with the cumulative histogram

Color Space	CR Levels			CR # of bins	CH Levels			CH # of bins	Comp. Time(Sec)	Wide Search	
	1st comp.	2nd comp.	3rd comp.		1st comp.	2nd comp.	3rd comp.			MAP %	Query Time(Sec)
HVC	6	3	3	57	6	3	3	57	37.08	0.92	0.3750
HSV	**18**	**3**	**3**	165	**18**	**3**	**3**	165	170.30	**0.93**	1.1729
HLS	6	2	2	27	18	3	3	165	23.89	0.92	0.2354
	6	3	3	57	18	2	2	75	31.98	0.92	0.2959
LAB	2	2	2	8	2	2	2	8	19.16	0.91	0.1885
RGB	2	2	2	8	2	2	2	8	22.94	0.91	0.2354
	2	2	2	8	4	4	4	64	22.75	0.89	0.2354
11Col				11				11	24.45	0.85	0.2188
9Col				9				9	9.83	0.91	0.0947

Table (5.22) Retrieval performance for the color correlogram using feature group distance with moments

Color Space	CR Quan. Levels			CR # of bins	Comp. Time(Sec)	Wide Search	
	1st comp.	2nd comp.	3rd comp.			MAP %	Query Time(Sec)
HVC	**6**	**3**	**3**	57	37.77	**0.88**	0.3125
HSV	18	3	3	165	166.69	0.84	1.1563
HLS	6	3	3	57	31.67	0.84	0.2813
LAB	2	2	2	8	21.45	0.84	0.2188
RGB	**2**	**2**	**2**	8	20.41	**0.88**	0.2031

Table (5.23) Retrieval performance for the color correlogram using feature average distance with moments

Color Space	CR Quan. Levels			CR # of bins	Comp. Time(Sec)	Wide Search	
	1st comp.	2nd comp.	3rd comp.			MAP %	Query Time(Sec)
HVC	**6**	**3**	**3**	57	36.97	**0.88**	0.3281
HSV	18	3	3	165	174.45	0.85	1.1572
HLS	6	3	3	57	32.34	0.84	0.2813
LAB	2	2	2	8	21.69	0.86	0.2031
RGB	**2**	**2**	**2**	8	20.61	**0.88**	0.1865

5. Cumulative Histogram with Moments Scheme

LAB color space quantized into 2×2×2 bins shows the best retrieval performance (with MAP=0.88%) when using wide search, as indicated in Table (5.24).

Table (5.24) Retrieval performance for the cumulative histogram with moments

Color Space	CH Quan. Levels			CH # of bins	Comp. Time(Sec)	Wide Search	
	1st comp.	2nd comp.	3rd comp.			MAP %	Query Time(Sec)
HVC	12	3	3	111	22.70	0.86	0.2188
	6	3	3	57	23.31	0.87	0.2344
HSV	18	3	3	165	18.16	0.86	0.1875
HLS	18	3	3	165	18.25	0.86	0.1875
LAB	**2**	**2**	**2**	8	17.63	**0.88**	0.1719
RGB	2	2	2	8	16.52	0.87	0.1563

5.7 Overall Retrieval Assessment

During the implementation and testing, a platform of Pentium 4 with a mobile Intel processor of 1.86 GHz, 512 M RAM, and 2 M Cache Memory was used. The software components and tools utilized along the developing and evaluation processes were: Visual Basic, Excel, and ACD systems.

5.7.1 Whole Schemes Evaluation

In the previous section, the used quantization schemes, and the color and texture analysis techniques (i.e., the co-occurrence matrix− *'CO'*, the color histogram− *'CH'*, the correlogram− *'CR'*, and color moments− *'Mom'*) were already applied in separated and combined manners.

The best retrieval result was obtained when the cumulative histogram was determined alone for HSV− 18×2×2 bins with Wide− *'W'* or Narrow− *'N'* search are used, where MAP=94% and the computation time=11.64 sec. The retrieval results for colorful textures prove that the use of color can improve retrieval performance significantly. The subsequent results are attained by using the following retrieval schemes:

GLCM & Cumulative histogram, with MAP=94%,
Color correlogram & Cumulative histogram, with MAP=93%,
GLCM alone, with MAP=92%,
GLCM & color moments, with MAP=91%, and
Color correlogram alone, with MAP=91%.

The worst retrieval result was obtained when GLCM was used alone with feature average distance scheme, where MAP was 82% for Gray− 8 bins. In addition to the best and worst results, the mid results are tabulated in Table (5.25).

Table (5.25) Overall performance results

Scheme Description	**Color Space**	**Feature Distance**	**Quantization Details**	**Total # of bins**	**Comp. Time** (Sec)	**MAP %**	**Query Time** (Sec)	**Search**
Cum. Hist.	HSV	---	18×2×2	75	11.64	**0.94**	0.1094	W&N
GLCM & Cum. Hist.	LAB	group	8 for CO 2×2×2 for CH	8 8	17.88	**0.94**	0.1875	W
GLCM & Cum. Hist.	HVC	group	4 for CO 6×3×3 for CH	4 57	26.69	**0.94**	0.2813	W
GLCM & Cum. Hist.	LAB	average	8 for CO 2×2×2 for CH	8 8	18.00	**0.93**	0.1719	W
GLCM & Cum. Hist.	HVC	average	8 for CO 12×3×3 for CH	8 111	23.38	**0.93**	0.2822	W
Correlogram& Cum. Hist.	HSV	group& average	18×3×3 for CR 18×3×3 for CH	165 165	170.3	**0.93**	1.1729	W
GLCM	Gray	group	4 for CO	4	8.22	**0.92**	0.0781	W
GLCM& Moments	RGB	group	8 for CO	8	15.33	**0.91**	0.1563	W
Correlogram	HVC	group	6×3×3	57	29.69	**0.91**	0.3125	N
Correlogram	HVC	average	6×3×3	57	30.16	**0.91**	0.2969	W
Correlogram	HVC	group	32×2×2	131	99.14	**0.91**	0.7656	W
GLCM& Moments	HSV	average	8 for CO	8	18.41	**0.90**	0.1719	W
Cum. Hist.& Moments	LAB	---	2×2×2 for CH	8	17.63	**0.88**	0.1719	W
Correlogram& Moments	RGB	group& average	2×2×2 for CR	8	20.41	**0.88**	0.2031	W
Moments	RGB	---	---	---	10.30	**0.83**	0.0353	W
GLCM	Gray	average	8 for CO	8	8.19	**0.82**	0.0781	W

Very similar results could be obtained by inspecting the F-measure chart, for the best six retrieval schemes depicted in Figure (5.3). However, as mentioned in section (5.3), the computational complexity is an important issue in CBIR. As indicated in Tables (5.12) and (5.14), cumulative histogram is much more efficient than color correlogram. When the number of correlogram bins increases the computation time increases significantly. This behavior wasn't noticed with the cumulative histogram, where the computation time approximately stays stable.

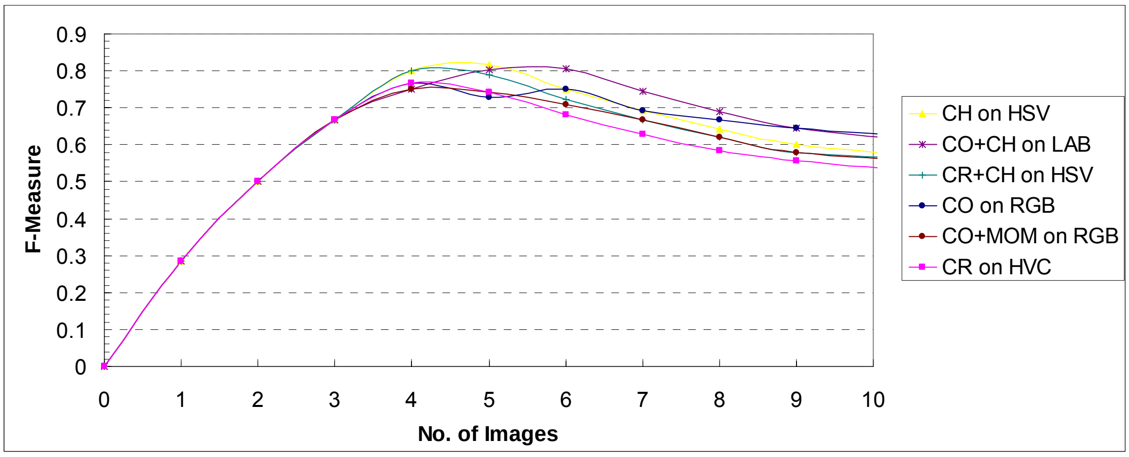


Figure (5.3) F-measure graph of best retrieval schemes

The human-based 9 colors category is computationally much cheaper than (HSV− 18×2×2 bins), also it shows high performance (MAP=93%) as introduced in Table (5.14). 9 colors category method has efficient computation time with good MAP as indicated in Tables (5.12), (5.20), and (5.21).

Wide search performs better than narrow search because it inspects more than one mother cluster. This will increase the probability of finding relevant images for the target query image. In contrary, narrow search is constrained to a single cluster. This cluster may miss the relevant images because of cluster centroid divergence; this problem could be solved by clusters centers optimization. Where, some enhancement to hierarchical clustering can be made by a relocation of images which may have been incorrectly clustered at an early stage. It is necessary to move all such images to their closest clusters.

Feature group proved to be more effective than feature average distances. As depicted in Figure (4.5), all four feature descriptors are utilized by using feature group distance. Also, all descriptors correspondence probabilities over the compared features vectors are taken into account. While by using feature average distance, these four feature descriptors are encapsulated (averaged) into a single number which may blur the single descriptors significance.

5.7.2 Clusters and Query Results Visualized

It is important to mention that these retrieval results reflect the good performance of the Hierarchical Agglomerative Clustering applied to all schemes. However, more convenient results could be attained by broadening the number of images used in the images database. Clustering based on a lower threshold (i.e. <=7%) gives, somehow, clusters with homogenous images as demonstrated in Figure (5.4). As the threshold increases (i.e. >7% and <=15%), clusters will contain more and more inhomogeneous images as illustrated in Figure (5.5). On the other hand, a brief demonstration of queries output is displayed in Figures (5.6) and (5.7). A wider exhibition of different schemes clusters and queries results could be found in Appendix B. These examples are directed by using the cases of *wide search* and *feature group* distance.

An overview on the results of separate queries or clustered images reveals that more success could be attained to each scheme performance (see for example Figures 5.4). This can be accomplished by assuming continuous rather than binary relevance judgments. In other words, instead of assigning binary (relevant=1, and irrelevant=0) values to each retrieved image, in-between values along the period [0, 1] could be used according to the degree of similarity between the query and retrieved images.

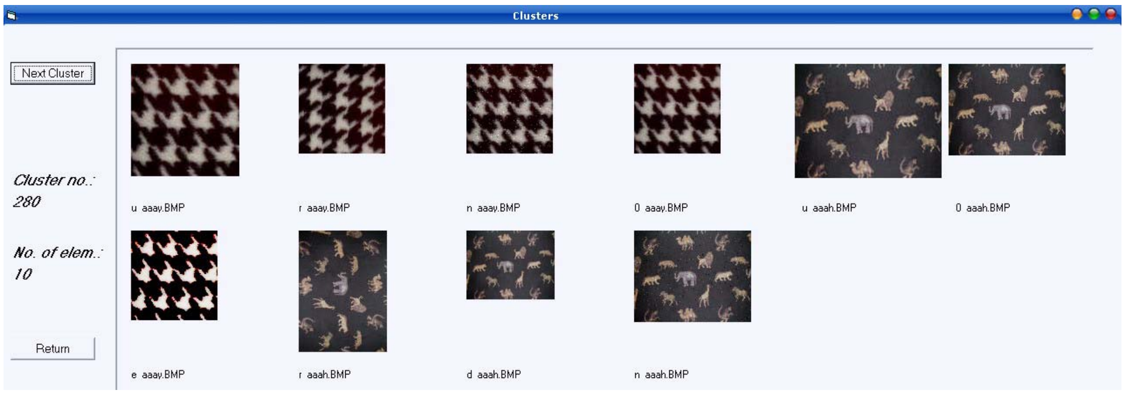


Figure (5.4) A cluster based on a lower threshold resulted from GLCM− 8 bins method applied on HLS color space.

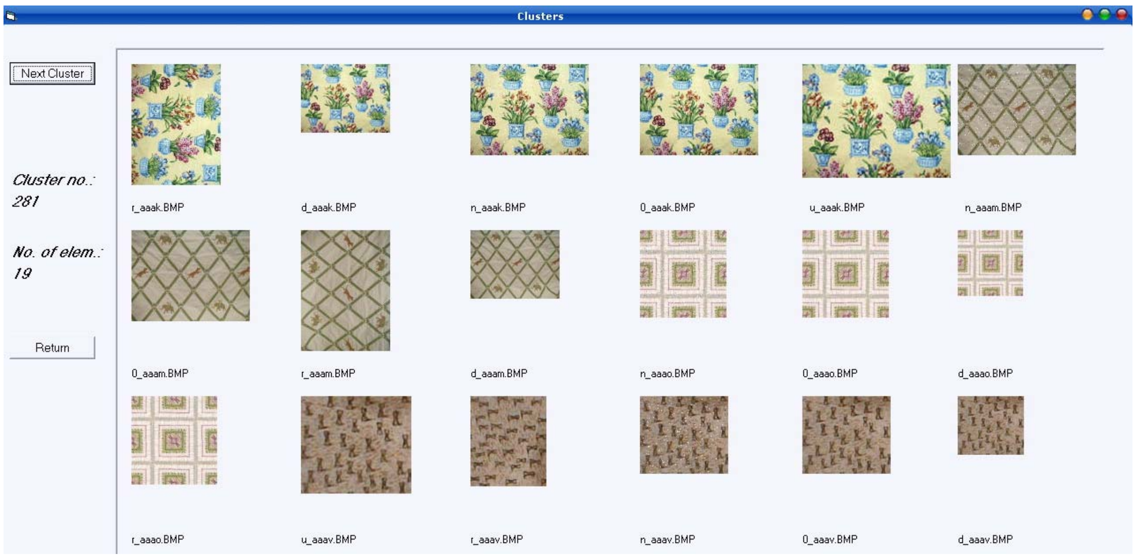


Figure (5.5) A cluster based on a higher threshold resulted from GLCM− 8 bins&CH− 2×2×2 bins method applied on RGB color space.

Concerning the cumulative histogram, retrieval depends on the whole color content of the image with no respect to texture, as indicated in Figure (5.6). On the opposite side, GLCM concentrates on gray-based textures with no attention to color content, as illustrated in Figure (5.7). Color correlogram is a good descriptor for colorful textures, while color moments describe the color contents of the image. In between these extremes, some combined approaches stand midway in sight to color and texture content of the image. An impression about the importance of the studied schemes to discriminate color/texture contents could be perceived by exploring Appendix B.

The configurations based on using coarse quantization (low number of bins) applied to the GLCM outperformed the more precise quantization (high number of bins) for all color spaces, as expressed in Table (5.10). For the color correlogram, the computation time increases significantly when the number of quantization bins increases, accompanied with a small improvement in retrieval accuracy (see Table 5.12). While for the color histogram, the computation time isn't influenced much by increasing the number of quantization bins. Nevertheless, fine color quantization performs a little better than coarse color quantization for a given color space, as presented in Table (5.14). Hence, the use of a coarse quantization for the color correlogram can substantially improve retrieval in terms of speed. In conclusion, for texture analysis, the chosen quantization scheme is the most important key factor; usually a lower number of bins performs better.

In color quantization, hue gets the finest quantization, while saturation and intensity get the coarse one. This matches human color perception which has more tolerance to these two components variations. Also, this quantization helps to keep the computation time at a reasonable level. For the undefined hue value, its corresponding quantized intensity value is used instead.

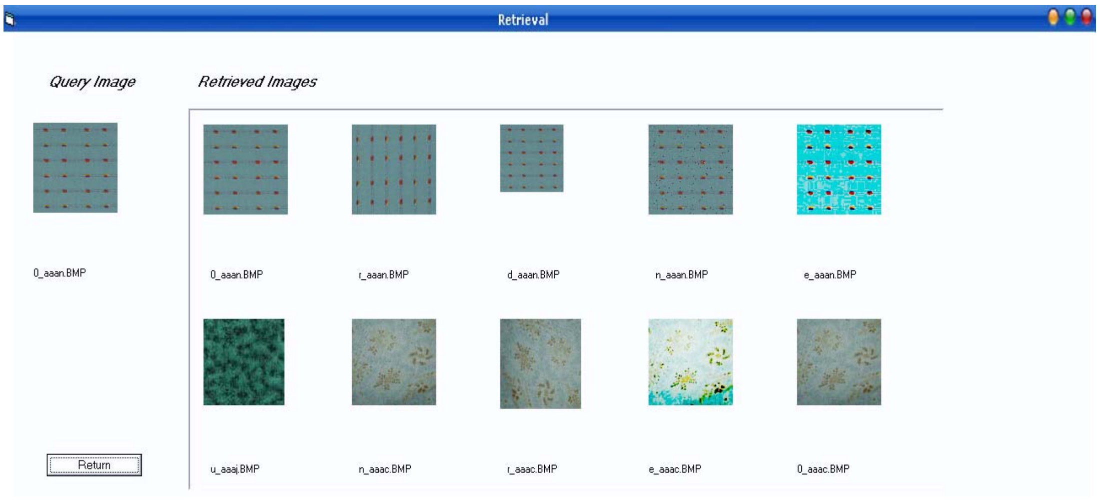


Figure (5.6) Query output resulted from CH− 2×2×2 bins method applied on HSV color space.

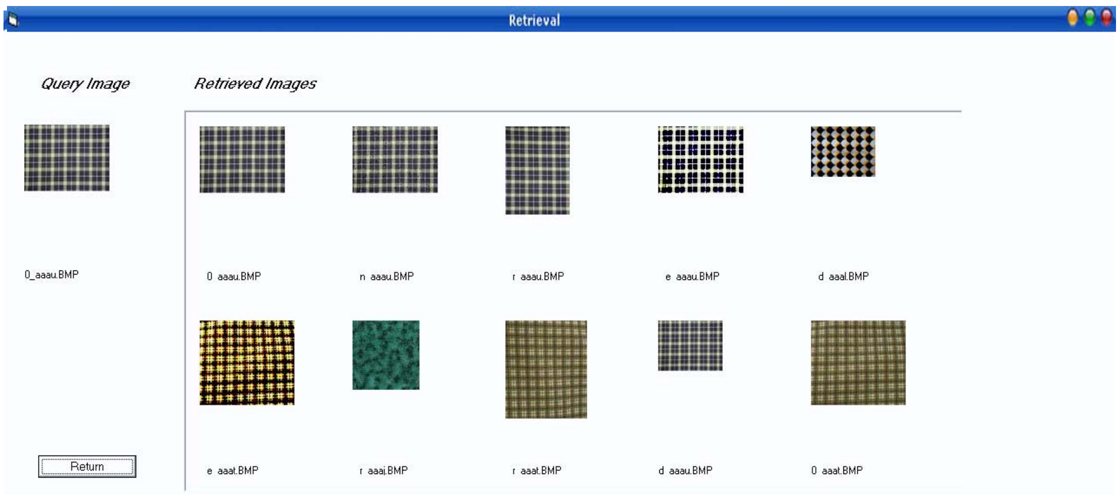


Figure (5.7) Query output resulted from GLCM− 8 bins method applied on HLS color space.

Chapter Six

Conclusions and Suggestions for Future Work

6.1 Conclusions

The results attained by test and evaluation processes have revealed the following articles:

1. Six of the eight used GLCM and correlogram features are found to be strong descriptors of texture. They are the Inverse Difference Moment, Haralick's Correlation, Cluster Shade, Cluster Prominence, Entropy, and Homogeneity. Using all the eight features may lead to low correct retrieval percentage.

2. Feature group distance has proved to be more effective than feature average distance. The four feature descriptors are utilized when using the feature group distance. While by using feature average distance, the four feature descriptors are averaged into a single number which is caused in blurring the single descriptors significance.

3. The co-occurrence matrices and the color correlogram are calculated for the distances 1, 2, 3, and 4. The most suitable distance found is 2.
4. Since RGB space is not quite perceptually uniform, it is transformed to the more perceptual uniform space (such as HVC, HSV, HLS, or LAB) in an early stage of any image retrieval scheme. Eligible color quantization should be applied to the selected color space.

5. The modified combined quantization presented in equation (4.2) has proved to be an accurate compaction of color space data, where the three color components are mapped into a single valued color. This compact value is used as an entry to the correlogram and cumulative histogram for further manipulating.

6. In both the co-occurrence matrix and the color correlogram, the most precise quantization schemes does not give a big improvement on the more coarse

quantization schemes, for a given color space. In this case, coarse quantization is preferred because it has low computational cost.

7. For the cumulative histogram and the color correlogram, hue is the most significant characteristic, so this component is given the finest quantization. 6, 12, and 18 bins are the best quantization for hue component, while saturation and intensity get coarse quantization (3 bins).
8. Among L_1, L_2, and (histogram intersection) distance measures, L_1 is the best for cumulative histogram, while L_2 is the best for GLCM, color correlogram, and color moments.
9. Taking in consideration that most of color histograms are very sparse and thus more sensitive to noise, their cumulative color histograms are used instead. The results demonstrate the advantages of this second approach over the conventional color histogram approach.
10. The human-based 9 colors category is computationally inexpensive than other color spaces, also it has a high performance with a tiny MAP difference. This makes it a good choice for realistic applications.
11. Color moments scheme shows poor performance, where the best reached retrieval result is with MAP=0.83%. Nevertheless, the features vector of the color moments extractor is among the smallest.
12. Hierarchical Agglomerative Clustering is a scalable search method; since at the time of query, only the relevant set of clusters needs to be examined. Also, it is a base to establish efficient and effective searching algorithms that facilitate searching through the constructed index structure.
13. Wide search scheme performs better than narrow search scheme because it inspects more than one mother cluster. This will increase the probability of finding more relevant images for the target query image. In contrary, narrow search is constrained to a single cluster. This cluster may miss the relevant images because of the clusters divergence.
14. More convenient results could be attained by increasing the number of images used in the images database. As the number of images increases, the clustering algorithm can utilize more similar images in the clustering process within a lower threshold.

6.2 Suggestions for Future Work

The work performed along the period of this research could be improved in many directions. Among the important directions are:

1. Greater success in retrieval results could be gained by assuming continuous rather than binary relevance judgments, where a degree of similarity is assigned to each retrieved image according to its resemblance to the query image.
2. When considering all the possible combinations of color pairs, the size of the color correlogram could be very large. Due to the high complexity of the color correlogram scheme, a simplified version of it (which is called the color autocorrelogram) can be used instead. The color autocorrelogram may capture the spatial correlation between identical colors only and thus reduces the dimension of the resulted matrix significantly.
3. In order to make a query upon heterogeneous images (i.e., images that require segmentation), the test image could be divided into a number of regions. The features extracted from each of these regions are compared correspondingly. This approach could be extended to make query at a region level, where only part(s) of the query image may be of interest to the user.
4. Concerning the cluster centroid divergence problem, an improvement on the HAC algorithm could be accomplished by optimizing the clustering process. The centroid updating could be done after each instance of new image addition to the cluster. As a result; after a number of centroids alterations, some images may have a larger similarity measure with other cluster centers than that with their own clusters centers. These images should be reallocated to their new closest clusters.
5. Incorporating users' interactions tools such as relevance feedback in the retrieval process may improve the performance of CBIR systems significantly. By using relevance feedback to the query results, the retrieval operation becomes an iterative process, where the system makes suggestions and the user provides feedback. There is a need for the system to learn from the user feedback. Learning should take place both within a single retrieval session (short-term), and across many retrieval sessions (long-term). This could aid in generating more perceptual and meaningful retrieval results.

References

[1] Chen, C., Wang, J.Z., and Krovetz, R., "Content-Based Image Retrieval by Clustering", Department of Computer Science, University of New Orleans, New Orleans, 2003.

[2] Mojsilović, A., Kovačević, J., Hu, J., Safranek, R.J., and Ganapathy, S.K., "Retrieval of Color Patterns Based on Perceptual Dimensions of Texture and Human Similarity Rules", Bell Laboratories, Lucent Technologies, Murray Hill, New Jersey, 2002.

[3] Long, F., Zhang, H.J., and Feng, D.D., "Fundamentals of Content-Based Image Retrieval", Microsoft Research, Asia, 2003.

[4] Muller, H., Michoux, N., Bandon, D., and Geissbuhler, A., "A Review of Content-Based Image Retrieval Systems in Medical Applications: Clinical Benefits and Future Directions", Division for Medical informatics, University Hospital of Geneva, Switzerland, 2003.

[5] Zhang, Y., "On The Use of CBIR in Image Mosaic Generation", Technical Report, Dept. of Computing Science, University of Alberta, Edmonton, Alberta, Canada, July, 2002.

[6] Tian, Q., Sebe, N., Lew, M.S., Loupias, E., and Huang, T. S. , " Image Retrieval Using Wavelet-Based Salient Points", Journal of Electronic Imaging, Special Issue on Storage and Retrieval of Digital Media, Vol.10(4), pp. 835-849, October, 2001.

[7] Rui, Y., and Huang, T.S., and Chang, S., " Image Retrieval: Past, Present, and Future", Department of Electrical Engineering & New Media Technology Center, Columbia University, New York, 1998.

[8] Veltkamp, R.C., and Tanase, M., "Content-Based Image Retrieval Systems: A Survey", Department of Computing Science, Utrecht University, Netherlands, 2001.

[9] Lew, M.S., ed., "Principles of Visual Information Retrieval", Advances in Pattern Recognition─ APR, Springer-Verlag, London, 2001.

[10] Tran, L.V., "Efficient Image Retrieval with Statistical Color Descriptors", Department of Science and Technology, Linkoping University, Norrkoping, Sweden, May, 2003.

[11] Shah, B.N., and Raghavan, V.V., “An Approach to Content-Based Image Retrieval using Clustering and Space Transformation”, University of Louisiana, Lafayette, USA, 2002.

[12] Gong, Y., “Intelligent Image Databases Towards Advanced Image Retrieval”, Kluwer Academic Publishers, USA, 1998.

[13] Celebi, E., and Alpkocak, A., “Clustering of Texture Features for Content Based Image Retrieval”, Dokuz Eylul University, Department of Computer Engineering, Izmir, Turkey, 2001.

[14] Vasconcelos, N., and Kunt, M., “Content-Based Retrieval from Image Databases: Current Solutions and Future Directions”, Proceedings of International Conference on Image Processing, Thessaloniki, Greece, 2001.

[15] Oracle 9i Documentation, “Oracle Visual Information Retrieval Manual”, Oracle Corporation, 2001.

[16] Johansson, B., “A Survey on: Contents Based Search in Image Databases”, Computer Vision Laboratory, Department of Electrical Engineering, Linkoping University, Sweden, 2000.

[17] Eakins, J.P., and Graham, M.E., “Content-Based Image Retrieval: A Report to the JISC Technology Applications Programme”, Institute for Image Data Research, University of Northumbria, Newcastle, 1999.

[18] Landis, S., “Content-Based Image Retrieval Systems for Interior Design”, Proceedings of SPIE− Storage and Retrieval for Image and Video Databases (3), 1995.

[19] Boujemaa, N., Boughorbel, S., and Vertan, C., “Soft Color Signatures for Image Retrieval by Content”, INRIA Rocquencourt, IMEDIA Project, Cedex, France, 2001.

[20] Aulia, E., ”Hierarchical Indexing for Region Based Image Retrieval”, M.Sc thesis, Louisiana State University, 2005.

[21] Gionis, A., “Similarity Search in High Dimensions via Hashing”, Proc. of the 25th VLDB Conference, pp. 518-529, Edinburgh, Scotland, 1999.

[22] Brandt, S., Laaksonen, J., and Oja, E., “Statistical Shape Features in Content-Based Image Retrieval”, in the Proceedings of ICPR2000, Barcelona, Spain, September, 2000.

[23] Wan, X., and Kuo, C.J, “A New Approach to Image Retrieval with Hierarchical Color Clustering,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 8, No. 5, pp. 628-643, September 1998.

[24] Schouten, T.E., and Rikxoort, E.M., “Colorful Texture Analysis”, Proceedings of Human Vision and Electronic Imaging, SPIE Vol. 5666, California, USA, 2005.

[25] Schettini, R., Ciocca, G., and Zuffi, S., “A Survey of Methods for Color Image Indexing and Retrieval in Image Databases”, Istituto Tecnologie Infomatiche Multimediali, Milano, Italy, 2001.

[26] Karypis, G., Han, E.H., and Kumar, V., “CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling”, Department of Computer Science and Engineering, University of Minnesota, 1999.

[27] Hengl, T., “Visualisation of Uncertainty Using the HSI Colour Model: Computations with Colours”, International Institute for Geo-Information Science & Earth Observation, Enschede, Netherlands, 2004.

[28] Liapis, S., and Tziritas, G., “Colour and Texture Image Retrieval using Chromaticity Histograms and Wavelet Frames”, Department of Computer Science, University of Crete, Heraklion, Greece, 2000.

[29] Yu, L., and Gimel, G., “ Image Retrieval Using Colour Co-occurrence Histograms”, Department of Computer Science, Tamaki Campus, University of Auckland, Palmerston North, New Zealand, November, 2003.

[30] Gentili, S., “Retrieving Visual Concepts in Image Databases”, Ph.D thesis, CS2003 (3), 2003.

[31] Laakso, S., ”Implementation of Content-Based WWW Image Search Engine”, M.Sc. thesis, Department of Electrical and Communications Engineering, Helsinki University of Technology, Finland, 2000.

[32] Hermes, T., Klauck, C., Krey, J., and Zhang, J., “Content-Based Image Retrieval”, Artificial Intelligence Group, Department of Computer Science, University of Bremen, Germany,1994.

[33] Mehrotra, S., “CAREER: Multimedia Analysis and Retrieval System (MARS)”, Department of Information and Computer Science, University of California at Irvine, 1997.

[34] Yu, H., Li, M., Zhang, H.J., and Feng, J., “Color Texture Moments for Content-Based Image Retrieval”, Microsoft Research Asia, and Center for Information Science, Peking University, Beijing, China, 2002.

[35] Geradts, Z., “Content-Based Information Retrieval from Forensic Image Databases”, The Netherlands Forensic Institute, Ministry of Justice, Netherlands, Printed by Ipskamp, Delft, 2002.

[36] Manjunathi, B.S., and Ma, W.Y., “Texture Features for Browsing and Retrieval of Image Data”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.18, No. 8, pp. 837-842, August, 1996.

[37] King, I., and Lau, T.K., “Competitive Learning Clustering for Information Retrieval in Image Databases”, Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, New Territories, Hong Kong, 1999.

[38] Kuo, W.J., Chang, R.F., Lee, C.C., Moon, W.K., and Chen, D.R., “Retrieval Technique for The Diagnosis of Solid Breast Tumors on Sonogram”, Ultrasound in Med. & Biol., Vol. 28, No. 7, pp. 903–909, USA, 2002.

[39] Laihanen, P., ”Colour Science Approach to Digital Image Reproduction”, Graphic Arts in Finland, Vol. 24, No. 1, pp. 3-18, Helsinki University of Technology, 1995.

[40] Konak, E.S., “A Content-Based Image Retrieval System for Texture and Color Queries”, M.Sc. thesis in Computer Engineering, Institute of Engineering and Science, Bilkent university, Turkey, August, 2002.

[41] Fails, J.A., “Image processing with crayons”, M.Sc. thesis, Department of Computer Science, Brigham Young University, August, 2003.

[42] Nastar, C., “Content-Based Image Retrieval: A State of the Art”, LTU technologies, 2003.

[43] Ordonez, J.R., Cazuguel, G., Puentes, J., Solaiman, B., and Roux, C., “Spatial-textural Medical Image Indexing based on Vector Quantization”, Brest-Cedex, France, 2003.

[44] Westmacott, M., “Content Based Image Retrieval Using Indexed Colour Features”, A mini-thesis for transfer from MPhil to Ph.D, Dept. of Electronics and Computer Science, Faculty of Engineering and Applied Science, University of Southampton, September, 2002.

[45] Eakins, J.P., “Retrieval of Still Images by Content”, Institute for Image Data Research, University of Northumbria, Newcastle, U.K., 2000.

[46] ISO/IEC/JTC1/SC29/WG11, “Coding of Moving Pictures and Audio”, International Organization for Standardization, 1997.

[47] Celebi, M.E., Aslandogan, Y.A., “Content-Based Image Retrieval Incorporating Models of Human Perception”, Technical Report CSE-2003-21, Department of Computer Science and Engineering, University of Texas, Arlington, 2003.

[48] Felipe, J.C., “Retrieval by Content of Medical Images Using Texture for Tissue Identification”, Department of Computer Science, Institute of Superior Education, Sao Paulo, Brazil, 2003

[49] Ravela, S., and Manmatha, R., “On Computing Global Similarity in Images”, Computer Science Department, University of Massachusetts, Amherst, USA, 2000.

[50] Glatard, T., Montagnat, J., and Magnin, I.E., “Texture Based Medical Image Indexing and Retrieval: Application to Cardiac Imaging”, Villeurbanne-Cedex, France, 2004.

[51] Park, M., Jin, J.S., and Wilson, L.S., “Fast Content-Based Image Retrieval Using Quasi-Gabor Filter and Reduction of Image Feature Dimension”, School of Information, Technologies University of Sydney, Australia, 2002.

[52] Saykol, E., Gudukbay, U., and Ulusoy, O., “Integrated Querying of Images by Color, Shape, and Texture Content of Salient Objects”, Department of Computer Engineering, Bilkent University, Ankara, Turkey, 2004.

[53] Smith, J.R., “Integrated Spatial and Feature Image Systems: Retrieval, Analysis and Compression”, Ph.D thesis, Graduate School of Arts and Sciences, Columbia University, 1997.

[54] Tian, Q., Sebe, N., Lew, M.S., Loupias, E., and Huang, T.S., “Content-Based Image Retrieval Using Wavelet-based Salient Points”, in the proceedings of SPIE Photonics, Electronic Imaging 2001− Storage and Retrieval for Media Databases, San Jose, California, Jan., 2001.

[55] Breiteneder, C., “Content-Based Image Retrieval of Coats of Arms”, Institute for Applied Computer Science and Information Systems, University of Vienna, Austria, 1999.

[56] Sangwine, S.J., and Horn, R.E.N., eds., “Color Image Processing Handbook”, Champan & Hall, 1998.

[57] Van de Wouwer, G., Scheuders, P., and Van Dyck, D., “Statistical Texture Characterization from Discrete Wavelet Representations”, IEEE Transactions on Image Processing , Vol. 8, No. 4, pp. 592-598, April, 1999.

[58] Bimbo, A.D., ed., “Image Analysis and Processing”, 9th International Conference, ICIAP’97, Florence, Italy, Proceedings, vol. II, Springer-Verlag, Berlin-Heidelberg, 1997.

[59] Martinez, J., and Guillaume, S., “Color Image Retrieval Fitted to Classical Querying”, cited in reference [58].

[60] Randen, T., and Husoy, J.H., “Filtering for Texture Classification: A Comparative Study”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 4, pp. 291-310, April, 1999.

[61] Do, M.N., and Vetterli, M., “Wavelet-Based Texture Retrieval Using Generalized Gaussian Density and Kullback–Leibler Distance”, IEEE Transactions on Image Processing, Vol. 11, No. 2, pp. 146-158, Feb., 2002.

[62] Hsu, C.T., and Shih, M.C., “Content-Based Image Retrieval by Interest Points Matching and Geometric Hashing”, Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, 2002.

[63] Rubner, Y., and Tomasi, C., “Perceptual Metrics for Image database Navigation”, Kluwer Academic Publishers, USA, 2001.

[64] Billmeyer, F.W., and Saltzman, J.M., “Principles of Color Technology”, 2nd edition, John Wiley & Sons, 1981.

[65] Greenberg, D., Marcus, A., and Schmidt, A.H., “The Computer Image: Application of Computer Graphics”, Addison Wesley, 1982.

[66] Kuehni, R.G., “COLOR: An Introduction to Practice and Principles”, John Wiley & Sons, 1997.

[67] Devijver, P.A., and Kittler, J., “Pattern Recognition: A Statistical Approach”, Prentice Hall, USA, 1982.

[68] Mollineda, R.A., and Vidal, E., “A Relative Approach to Hierarchical Clustering”, pp. 19-28, in Torres, M.I., and Sanfeliu, A., eds., “Pattern Recognition and Applications”, IOS Press, 2000.

[69] Duda, R.O., and Hart, P.E., “Pattern Classification and Scene Analysis”, John Wiley & Sons, 1973.

[70] Willett, P., “Similarity and Clustering in Chemical Information Systems”, John Wiley & Sons, and Research Studies Press, 1987.

[71] Bertino, E., Catania, B., and Zarri, G.P., “Intelligent Database Systems”, ACM Press, Addison Wesley, 2001.

[72] Han, J., and Kamper, M., “Data Mining: Concepts and Techniques”, Morgan Kaufmann Publishers, Academic Press, 2001.

[73] Baeza-Yates, R., and Ribeiro-Neto, B., “Modern Information Retrieval”, ACM Press, Addison Wesley, 1999.

[74] Santini, S., “Exploratory Image Databases: Content-Based Retrieval”, University of California, Academic Press, 2001.

[75] Levkowitz, H., “Color Theory and Modeling for Computer Graphics, Visualization, and Multimedia Applications”, University of Massachusetts, USA, Kluwer Academic Publishers, 1997.

[76] Gonzalez, R.C., and Wintz, P., “Digital Image Processing”, Pearson Education Asia, Addison Wesley Longman, Singapore, 2000.

[77] Umbaugh, S.E., “Computer Vision and Image Processing: A Practical Approach Using CVIP tools”, Prentice Hall, USA, 1998.

[78] Rosenfield, A., and Kak, A.C., “Digital Picture Processing”, Academic Press, 1976.

[79] Tremblay, J., and Cheston, G.A., “Data Structures and S/W Development in an Object-Oriented Domain”, fifth edition, Prentice Hall, USA, 2001.

[80] Horowitz, E., and Sahni, S., “Fundamentals of Data Structures in Pascal”, Computer Science Press, 1984.

[81] Ali, A.H., “Clouds Height Classification Using Texture Analysis of Meteosat Images”, Ph.D thesis, Al-Nahrain University, College of Science, 2004.

[82] Khan, L., and Wang, L., “Automatic Ontology Derivation Using Clustering for Image Classification”, Department of Computer Science, University of Texas at Dallas, 2002.

[83] Huang, Z., “Individual Study Option: Scalable Multimedia Database Indexing”, ISO report, 2004.

[84] Valgeirsson, A.G., Erlingsson, B., and Einarsson, I.S., “Using Clustering to Index Image Descriptors: A Performance Evaluation”, Reykjavik University, 2003.

[85] Ching, C.K., Fei, H.D., Wing, L.K., Hin, L.Y., Wing, L.C., and Hin, N.T., "Content-Based Image Retrieval", Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, 1999.

[86] Bekker, S.C.M., "Uniform Integration of Feature Extractors: The Design of an Intermediary for Content-Based Multimedia Retrieval", Department of Computer Science, University of Twente, KPN Research, 2001.

[87] Traina, A.J.M., Jr, C.T., and Bueno, J.M., "The Metric Histogram: A New and Efficient Approach for Content-Based Image Retrieval", Computer Science Department, University of Sao Paulo, Brazil, 2002.

[88] Shin, M.K., Huh, S.Y., Lee, S.H., Yoo, J.S., Jo, K.H., and Lee, J.S., "An Efficient Index Structure for High Dimensional Image Data", International Journal of Information Technology, Vol.6, No.1, Korea, May, 2000.

[89] Kushima, K., Satoh, M., Akama, H., and Yamamuro, M., "Integrating Hierarchical Classification and Content-Based Image Retrieval− Image Compass", NTT Cyber Space Laboratories, Japan, 2000.

[90] Park, G., Baek, Y., and Lee, H.K., "A Ranking Algorithm Using Dynamic Clustering for Content-Based Image Retrieval", Department of Electrical Engineering & Computer Science, Korea Advanced Institute of Science and Technology, Korea, 2002.

[91] Broek, E. L., and Rikxoort, E. M., "Evaluation of Color Representation for Texture Analysis", Proceedings of the Belgian Dutch Artificial Intelligence Conference, BNAIC-2004, pp. 35-42, October, Groningen, Netherlands, 2004.

[92] Zhang, J., Hsu, W., and Lee, M.L., "An Information-Driven Framework for Image Mining", Department of Computer Science, School of Computing, National University of Singapore, Singapore, 2001.

[93] Pua, K.M., "Feature-Based Video Sequence Identification", Department of Electrical Engineering and Computer Science, University of Kansas, 2002.

[94] Muller, H., Muller, W., Squire, D.M., and Pun, T., "Performance Evaluation in Content-Based Image Retrieval: Overview and Proposals", Computer Vision Group, Computing Science Center, University of Geneva, Switzerland, 1999.

Appendix A

Computation of Precision and Recall

Table (A.1) presents a sample of query result, the precision and recall values are computed for each retrieved image.

Table (A.1) Single query result

n	img #	relevant	Recall	Precision
1	58	√	0.2	1.00
2	89	√	0.4	1.00
3	76		0.4	0.67
4	90	√	0.6	0.76
5	86		0.6	0.60
6	59	√	0.8	0.67
7	98		0.8	0.57
8	88		0.8	0.50
9	78		0.8	0.44
10	95		0.8	0.40
11	10		0.8	0.36
12	51		0.8	0.33
13	72	√	1.0	0.38
14	90		1.0	0.36

Total no. of relevant images = 5

R=1/5=0.2; P=1/1=1

R=2/5=0.4; P=2/2=1

R=2/5=0.4; P=2/3=0.67

R=5/5=1; P=5/13=0.38

Precision and recall values listed in Table (A.1) are graphed against the number of retrieved images, and shown in Figure (A.1).

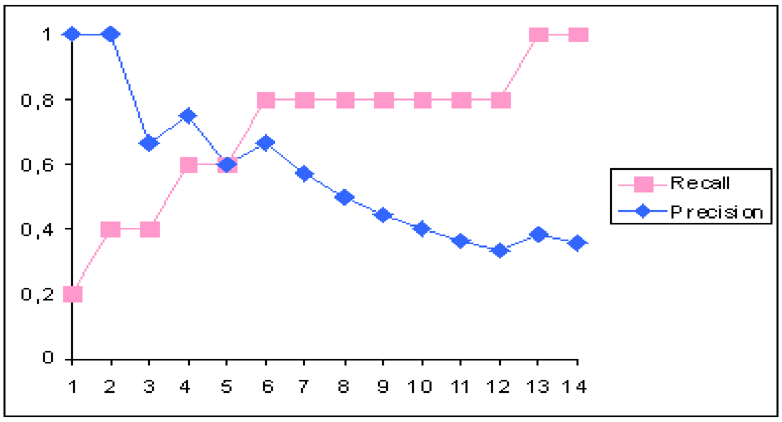


Figure (A.1) Precision & Recall for a single query, 14 images retrieved: 1,2,4,6 and 13 are relevant

Precision and recall values listed in Table (A.1) are graphed against each other in Figure (A.2).

n	Recall	Precision
1	0.2	1.00
2	0.4	1.00
3	0.4	0.67
4	0.6	0.76
5	0.6	0.60
6	0.8	0.67
7	0.8	0.57
8	0.8	0.50
9	0.8	0.44
10	0.8	0.40
11	0.8	0.36
12	0.8	0.33
13	1.0	0.38
14	1.0	0.36

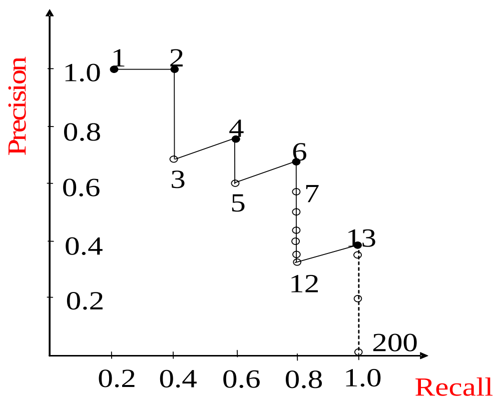


Figure (A.2) Precision & recall relationship

The individual precision values are interpolated to a set of 11 standard recall levels (0, 0.1, 0.2, ..., 1) to ease the computation of average precision and recall values using the following equation:

$$P(r_j) = \max_{r_j \le r \le r_{j+1}} P(r) \quad , \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \text{(A.1)}$$

The interpolated precision values are shown in Table (A.2)

Table (A.2) Interpolated precision values

Recall	Precision
0.0	1.0
0.1	1.0
0.2	1.0
0.3	1.0
0.4	1.0
0.5	0.75
0.6	0.75
0.7	0.67
0.8	0.67
0.9	0.38
1.0	0.38

→ *actual Precision value*

→ *interpolated Precision value (:by convention equals to the next actual data value).*

Recall cutoff graph resulted from choosing the interpolation points is depicted in Figure (A.3).

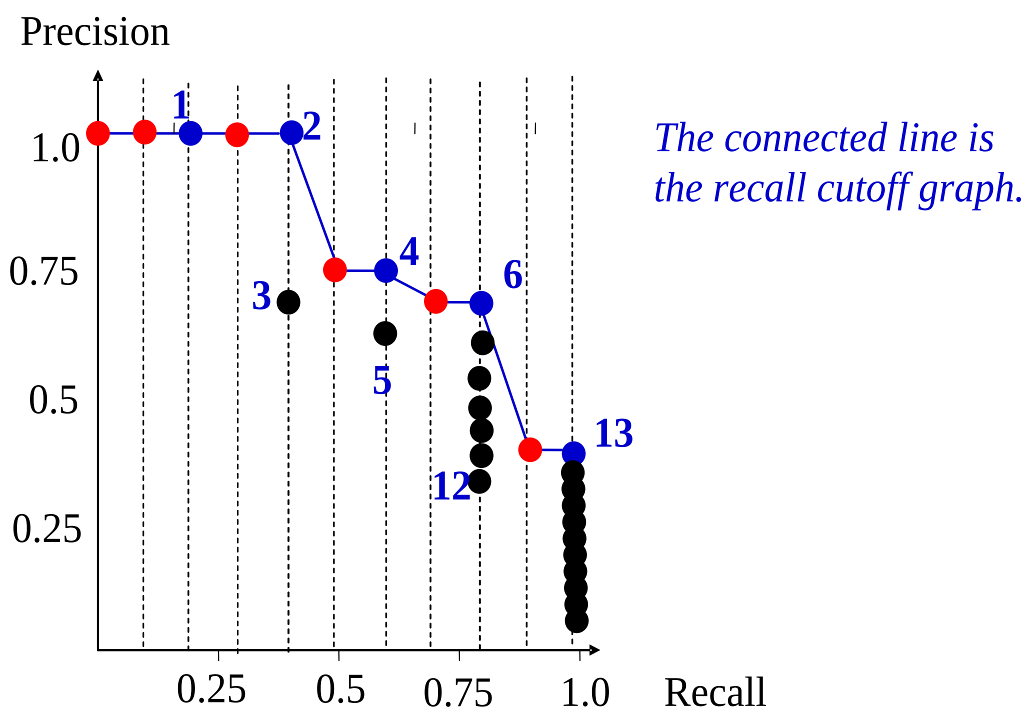


Figure (A.3) Recall cutoff graph

Appendix B

Clustering and Querying Output

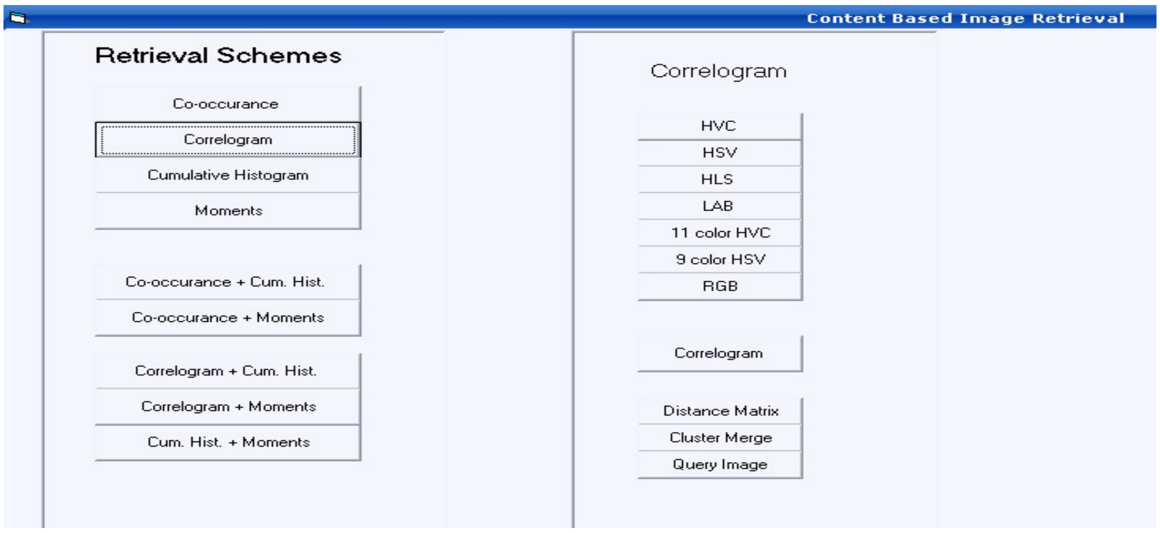


Figure (B.1) The user interface for choosing one of the applied retrieval schemes with the selected color model.

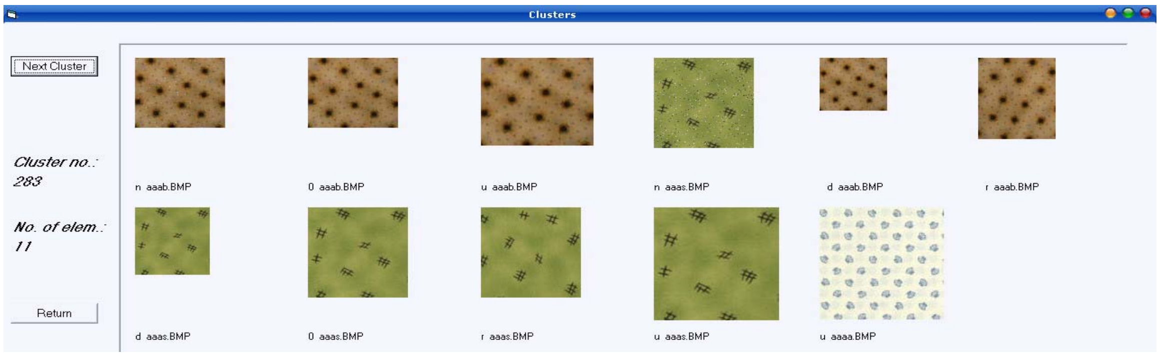


Figure (B.2) A cluster resulted by applying GLCM– 16 bins scheme based on HSV.

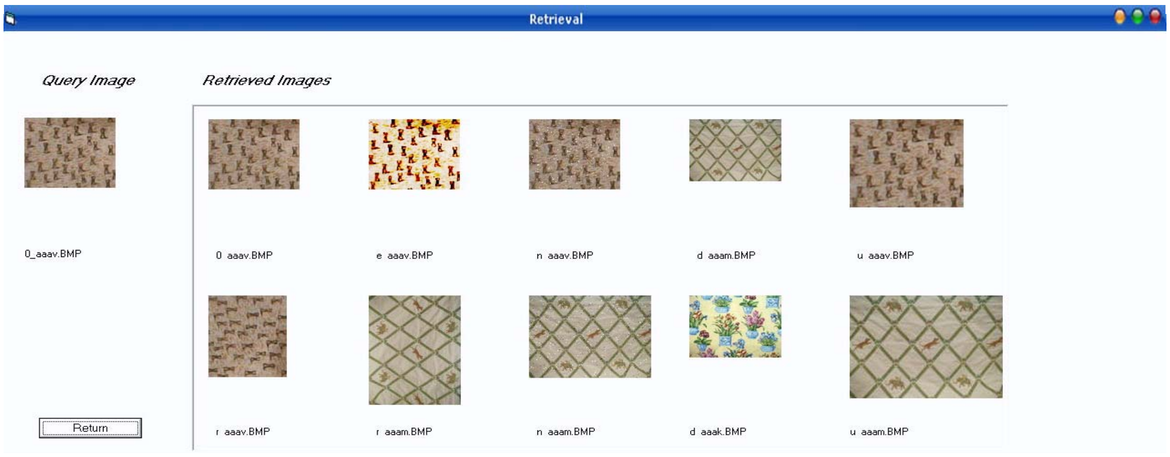
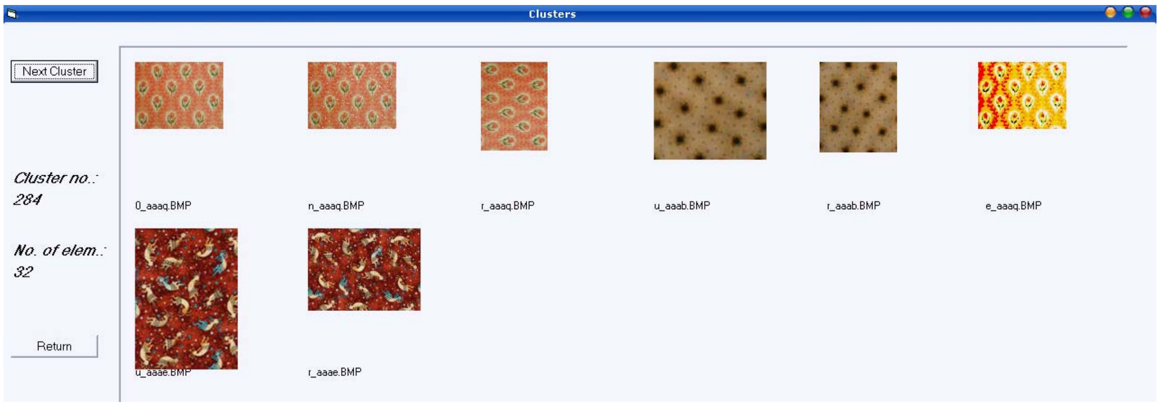


Figure (B.3) A query result by applying GLCM– 16 bins scheme based on HLS.



Cluster no.: 284

No. of elem.: 32

Figure (B.4) A cluster resulted by applying correlogram– 2×2×2 bins scheme based on HVC.

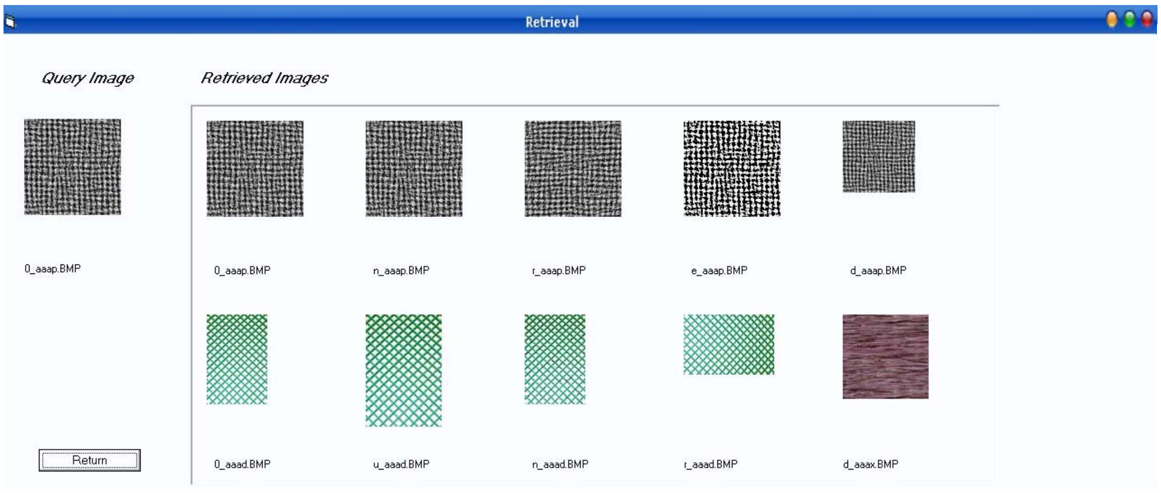
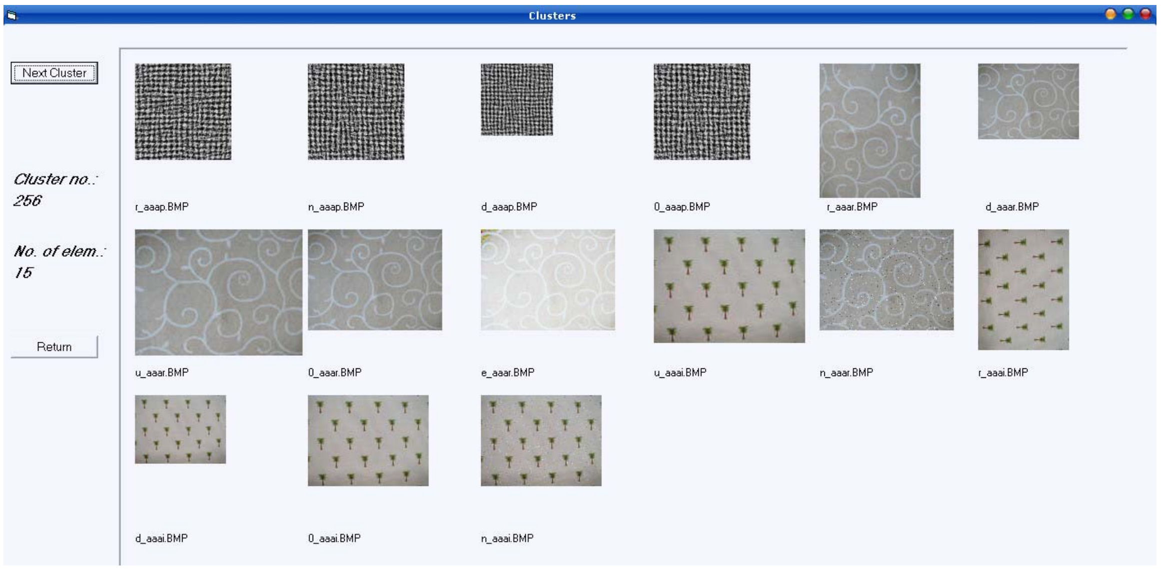


Figure (B.5) A query result by applying correlogram– 2×2×2 bins scheme based on RGB.



Cluster no.: 256

No. of elem.: 15

Figure (B.6) A cluster resulted by applying cumulative histogram– 9 bins scheme based on 9 colors category.

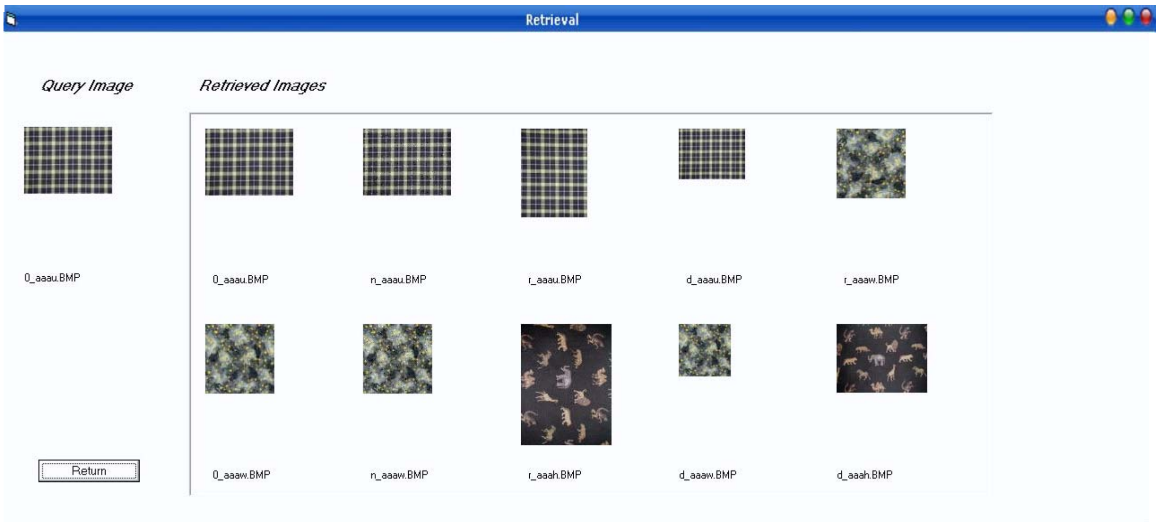


Figure (B.7) A query result by applying cumulative histogram− 6×3×3 bins scheme based on HVC

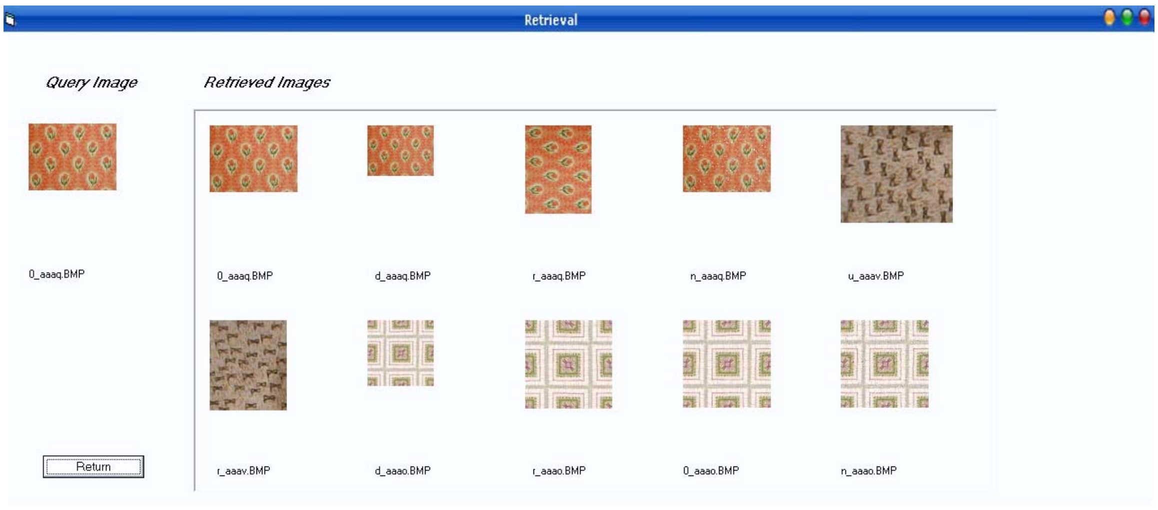


Figure (B.8) A query result by applying cumulative histogram− 18×3×3 bins scheme based on HSV.

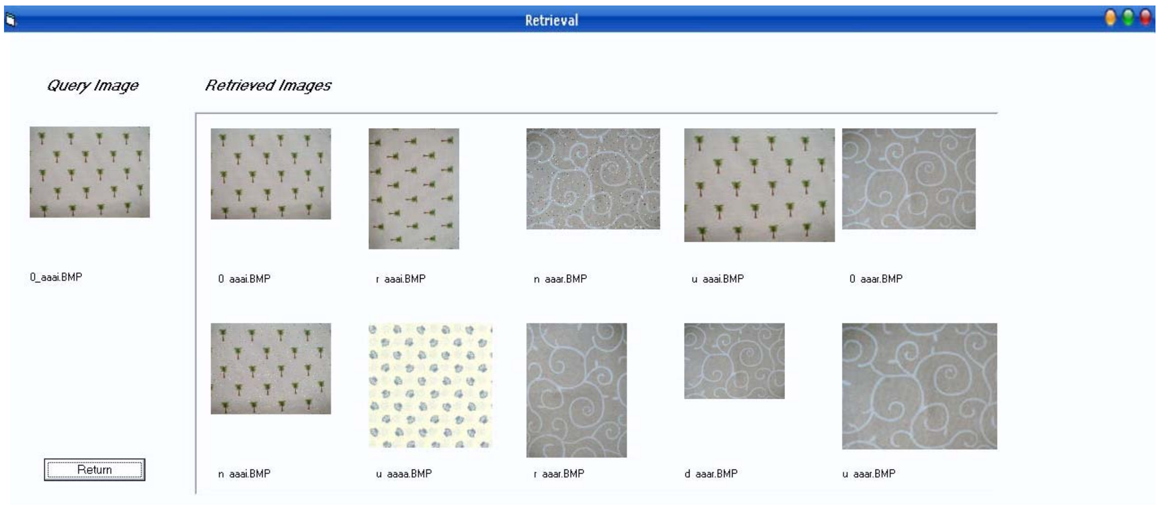


Figure (B.9) A query result by applying the moments scheme based on HVC.

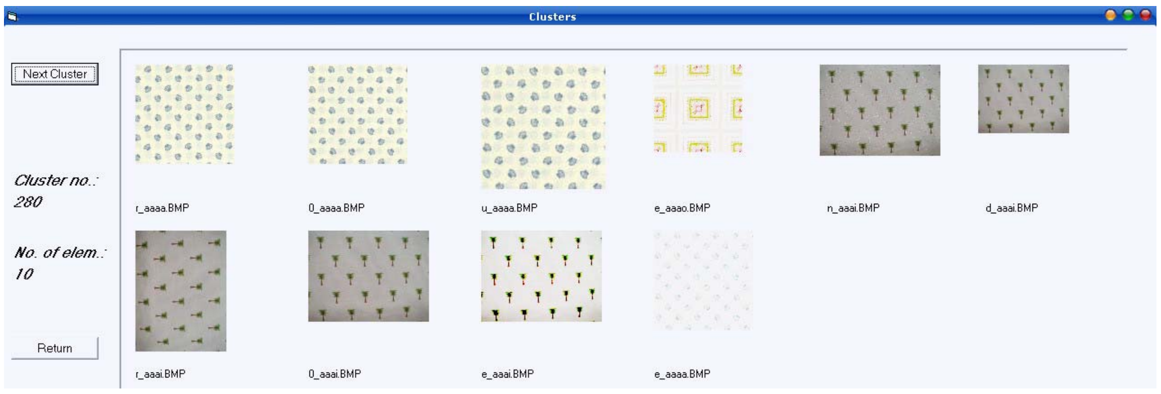


Figure (B.10) A cluster resulted by applying GLCM− 8 bins & cumulative histogram− 6×3×3 bins based on HLS.

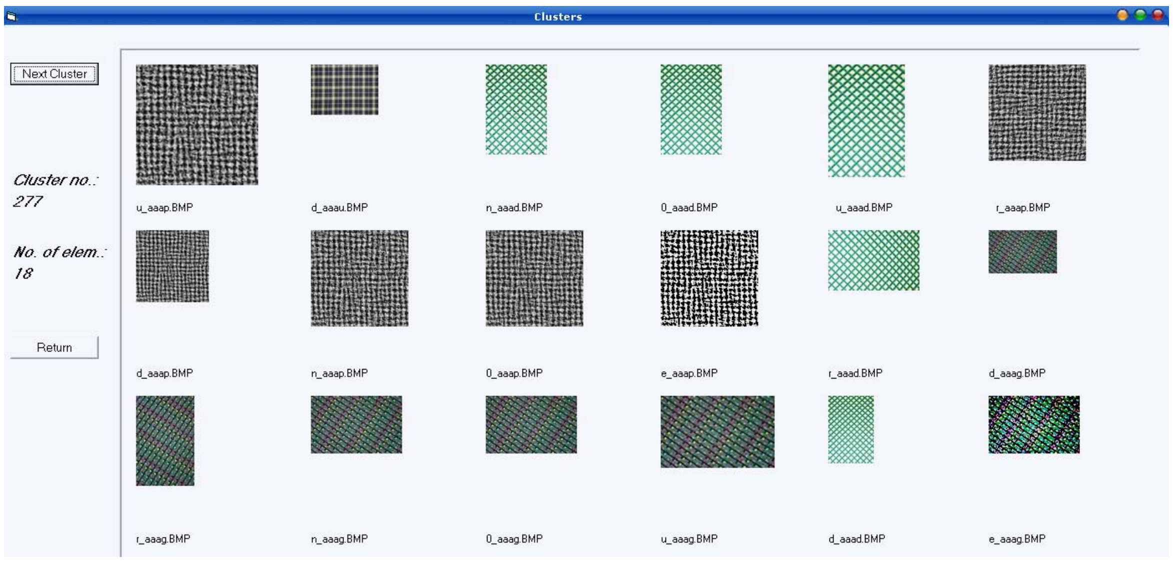


Figure (B.11) A cluster resulted by applying GLCM− 8 bins & cumulative histogram− 2×2×2 bins scheme based on RGB.

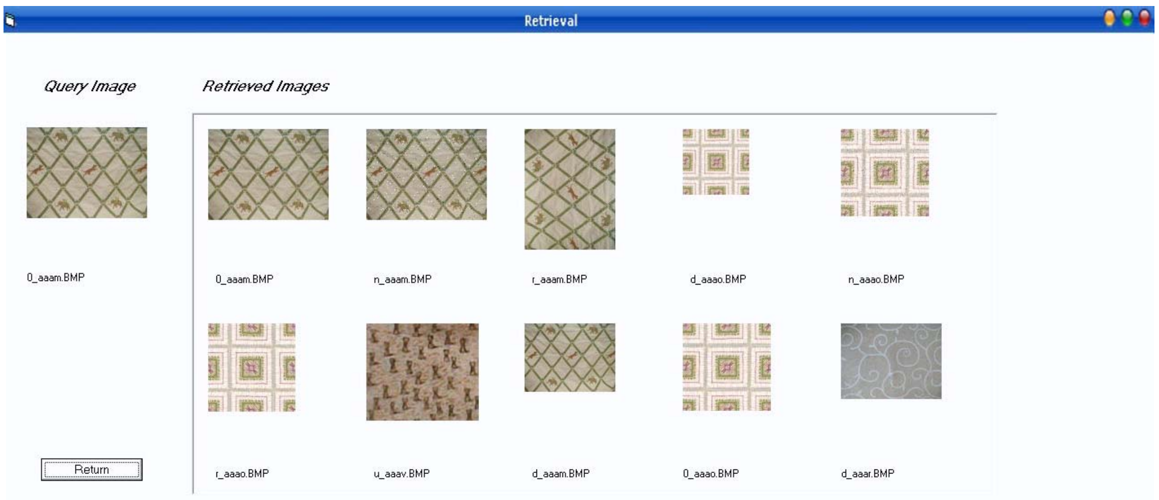


Figure (B.12) A query result by applying GLCM− 8 bins & cumulative histogram− 2×2×2 bins scheme based on LAB.

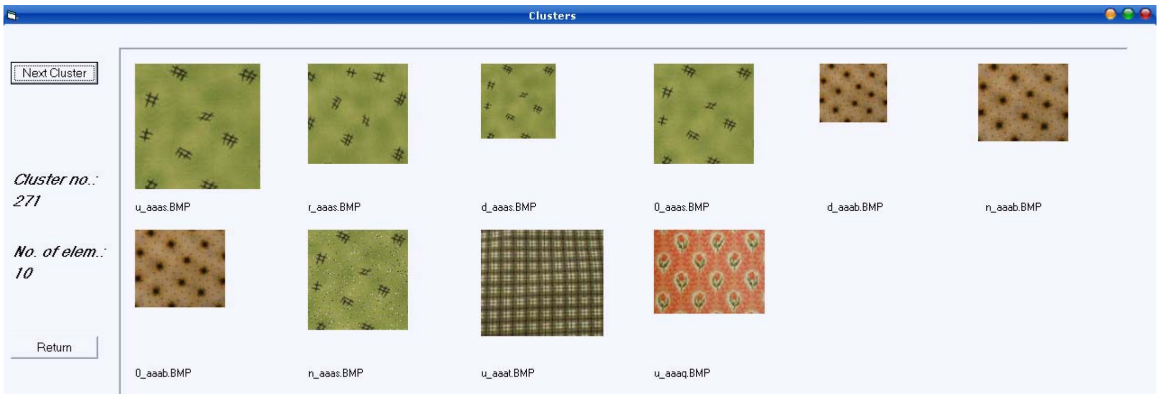


Figure (B.13) A cluster resulted by applying correlogram– 2×2×2 bins & cumulative histogram– 2×2×2 bins scheme based on LAB.

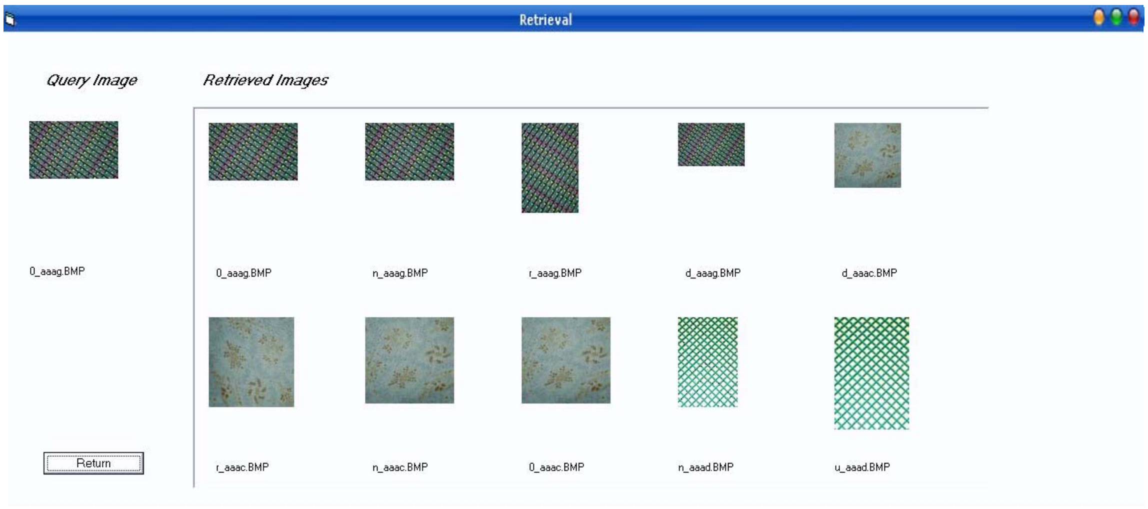


Figure (B.14) A query result by applying correlogram– 9 bins & cumulative histogram– 9 bins scheme based on 9 colors category.

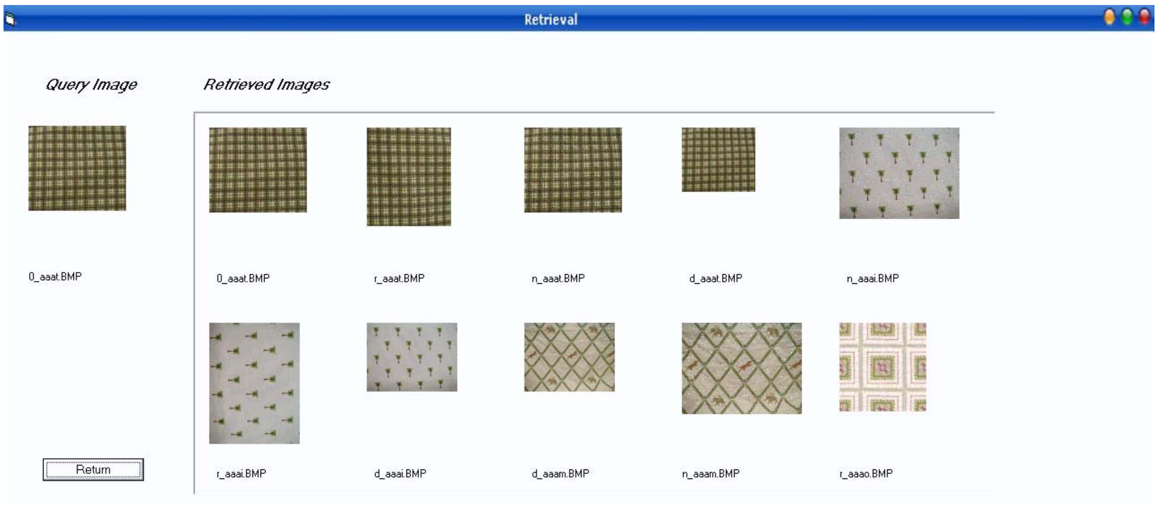


Figure (B.15) A query result by applying correlogram– 6×3×3 bins & cumulative histogram– 6×3×3 bins scheme based on HVC.

الخلاصة

في هذه الرسالة، يتم عرض نظام إسترجاع صورة معتمد على المحتويات يدعم الإسترجاع بالنسبة إلى صفات منخفضة المستوى وهي اللون والتركيب النسيجي للصورة. تتلخّـص الفكرة الأساسية للعمل في إستخلاص ميزات الصورة اؤتوماتيكيا وذلك عن طريق تحليل محتويات الصورة. سيكون التركيز على احتساب التشابه الإجمالي بين الصور. يجري الإستعلام عن صور متجانسة اللون والتركيب النسيجي والتي لاتستدعي التقطيع. إنّ مجال الصور المختار هو تصميم الأزياء والتصميم الداخلي للمباني.

تعتمد التقنيات المضمّـنة على إتّـخاذ مصفوفة الظهورالمتلازم للألوان الرمادية ومعيارالتلازم للألوان كوسائل إحصائية لتحليل التركيب النسيجي للصور. بالإضافة إلى ذلك، تمّ إستغلال المعيار التراكمي و عزوم اللون في تحليل اللون. وقد طبّـقت هذه الأساليب بحالات منفردة و إندماجية.

تمثّـل كل صورة كمتـّجه (أو مجموعة متـّجهات) في فضاء الصفات. وقد تمّـت فهرسة متـّجهات الصفات بإستخدام خوارزمية التجميع التكراري والتي تدعى التجميع التكتّـلي الهرمي وهي توفـّر هياكل بيانات سهلة الفهرسة مع إمكانية تنفيذ إسترجاع سريع. يتم تعريف مقياس التشابه بين الصور من البعد في فضاء الصفات. عند الإستعلام عن صورة، يتم إستخلاص متـّجه الصفات لهذه الصورة ومن ثمّ مقارنته مع متـّجهات البقية حسب التركيب الفهرسي وبإستخدام خوارزميات البحث الواسع أو المحدود. وبذلك يمكن ترتيب مجموعة صور من قاعدة البيانات حسب بعد متـّجهاتها عن متـّجه الصورة المستعلم عنها. إنّ هذه المجموعة المرتـّبة هي ناتج الإستعلام.

تمّ خلال عملية التقييم إجراء دراسة مقارنة لمختلف أساليب الإسترجاع المطبّـقة. وكان المعيار التراكمي هو الأفضل بالنسبة لحقل الصور المختار سواء تمّ تطبيقه منفردا أوعند دمجه مع مصفوفة الظهور المتلازم للألوان الرمادية أو معيار التلازم للألوان، بالتتابع. أوضح التقييم التجريبي إنّ خوارزمية الفهرسة المبنية على التجميع تعطي دقـّة إسترجاع عالية مع إختزال كبير في عدد مقارنات التشابه المطلوبة. إنّ سرعة البحث سوف تتحسّن بسبب أنّ الصورة المستعلم عنها سوف لن يتم مقارنتها مع كل صور قاعدة البيانات.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

تطوير نظام إسترجاع صورة معتمد على المحتويات

رســـالــة
مقدمــة إلى كليــة العلـوم في جــامعــة النهريــن
جــزء من متطلبــات نيــل درجــة دكـتوراه فلســفةوهي
فــي علــم الحــاسوب

من قِبل

مهدي كزار دعيمي

بكلوريوس، علم الحاسوب، جامعة النهرين، 1992
ماجستير، علم الحاسوب، جامعة النهرين، 1995

بإشراف

د. لؤي إدور جورج　　　　د. ليث عبد العزيز العاني

ذو الحجّة 1427 هـ　　　　كانون الأول 2006 م